*Application Note*

# *TPS257xx-Q1 Firmware Update With a Host*

*Haury Zhou, Bing Huang, and Seong Kim*

**ABSTRACT**

The TPS2576x-Q1 and TPS2577x-Q1 are fully-integrated USB Type-C® Power Delivery (PD) management devices targeting automotive USB Type-C charging applications. For some application scenarios, customers might need to update the firmware for the device. This application note covers the TPS2576x-Q1/TPS2577x-Q1 firmware update process using an embedded device such as an MCU. The TPS2576x-Q1/TPS2577x-Q1 stores the boot and application code on its internal ROM, and is able to accept patch bundles (containing application configuration and code patch) from an external host or EEPROM over I2C.

## Table of Contents

## List of Figures

## List of Tables

## Trademarks

USB Type-C® is a registered trademark of USB Implementers Forum.
All trademarks are the property of their respective owners.

## 1 Purpose and Scope

This application note details the TPS2576x-Q1/TPS2577x-Q1 firmware update process. The TPS2576x-Q1/ TPS2577x-Q1 firmware, which includes the boot and application code, is stored in the ROM to execute device functionality. The device can then accept patch bundles (containing code patch and application configuration) from an external host or EEPROM over the I2C bus and is stored in the SRAM. The patch bundles can be generated using the TPS2576x-Q1/TPS2577x-Q1 application customization tool (*TPS257XX-Q1-GUI* on TI's website). To learn more about the GUI, see the *TPS257XX-Q1-GUI Configuration Guide*.

## 2 Firmware Boot Code Brief

The firmware boot code of the device has two primary functions:

1. Device initialization.
2. Patch bundle loading and configuration.

When the device is powered and in Active Mode, LDO_3V3 is enabled and a power-on reset (POR) signal is issued. The digital core receives this reset signal and loads the boot code from the internal ROM, and then begins initializing the device settings. This initialization includes enabling and resetting internal registers, loading initial values and configuring the I2C addresses of the device. The boot code will also measure the resistance on the TVSP pin and decode a TVSP Index value. Depending on the Boot Mode selected, the device will accept patch bundles from either an external EEPROM or host controller such as an MCU. The various TVSP configurations and the corresponding boot sequences of the device are detailed in Appendix A.

## 3 Patch Bundle Brief

The firmware patch bundle is used for two purposes:

1. Provide code patch to replace functions inside the application ROM.
2. Provide configuration for the device.

As described in the previous section, the firmware patch bundle can be loaded from an external EEPROM or from a host after POR by using the device’s I2C host interface. The patch bundle is from a single source. If the configuration and patch data do not load successfully due to communications error, the boot code will use the factory device configurations.

# 4 Firmware Update

## 4.1 Overview

Firmware update refers to the update of the firmware's application configuration and or code patch using a patch bundle through the external EEPROM or external host. This application note covers a set of 4CC ASCII commands (see Appendix B) that enables a host to either program the patch bundle onto an external EEPROM (see Figure 4-1) when the device is in EEPROM Boot Mode or directly download the patch bundle onto the device's SRAM (see Figure 4-2) when it is in HUB/MCU Boot Mode.

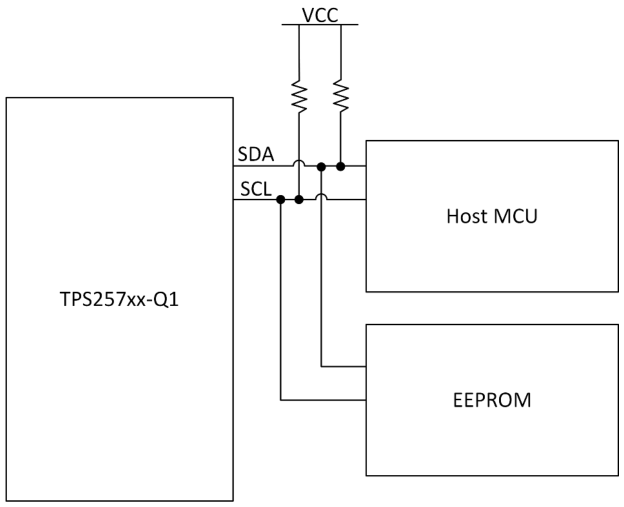


**Figure 4-1. Using a Host to Program the EEPROM for EEPROM Boot Mode**

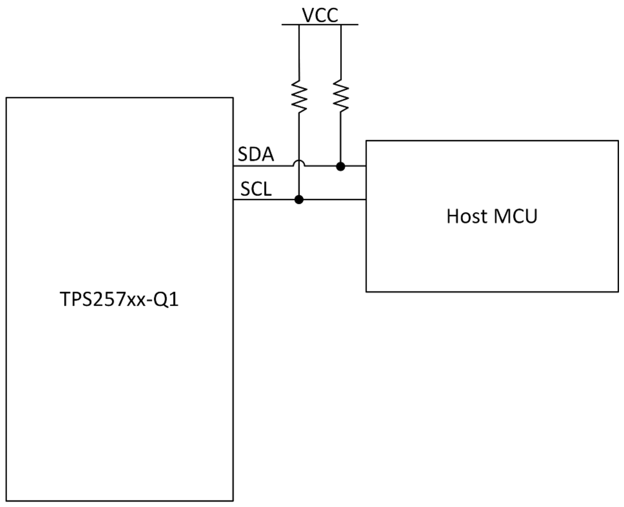


**Figure 4-2. Booting Directly from a Host Using HUB/MCU Boot Mode**

## 4.2 EEPROM Firmware Update

If TVSP is set to external EEPROM patch bundle loading, the device will try to access the address of the external EEPROM (0x50) when powered on. And if an external EEPROM is detected, the boot code firstly attempts to read the patch bundle from the low region of the EEPROM. If any part of the read process yields invalid data, the device aborts the low region read and attempts to read from the high region. If both regions contain an invalid patch bundle, the boot firmware proceeds to enter patch mode. When the device is in patch mode, most features will be limited and only few functions of the device will work. Therefore, the device will enter patch mode when TVSP is set to EEPROM Boot Mode, but is not connected to one.

The device is designed to power the external EEPROM from its internal LDO Regulator LDO_3V3, so pull-up resistors used for the EEPROM memory must be tied to LDO_3V3. Note that the LDO_3V3 can supply external circuits up to 25 mA.

The I2C port supports Standard, Fast Mode, and Fast Mode Plus I2C interfaces. The basic requirements are as below:

- 32 kB (256 kb)
- 7-bit I2C address (0x50)
- Organization: 32 kb × 8 (totals 256 kb)
- Active firmware image is stored in one 16 kb × 8 partition. The previous firmware image is retained in the other 16 kb × 8 partition for reliability.
- Page size buffer needs to be 64 Bytes

**Table 4-1. Suggested EEPROM**

| Manufacturer | Part Number |
|---|---|
| On Semi | CAV24C256 |
| Microchip | 24LC256 |
| ST Micro | M24256 |
| Rohm | BRA24T512FVT-3AM |

### *4.2.1 EEPROM Memory Organization*

The patch bundle of the device can be contained in the memory of the EEPROM NVM IC. The organization of this data in EEPROM is explained in this section using variables and relative locations. Memory organization might differ between the variants of device, and the application developers shall account for these differences when developing the EEPROM-update applications for the device.

In this application note, the memory organization of an EEPROM is explained based on the assumption that the EEPROM is dedicated to the device and not shared with other ICs of the system. For redundancy, the patch bundle is copied into two regions; Region-0 (low region) and Region-1 (high region). If the full memory of the EEPROM is read directly, it will contain region-headers and regions, which will simply be referred to as an *eeprom.bin* file. A depiction of the full *eeprom.bin* EEPROM memory organization for the device is shown in Figure 4-3.

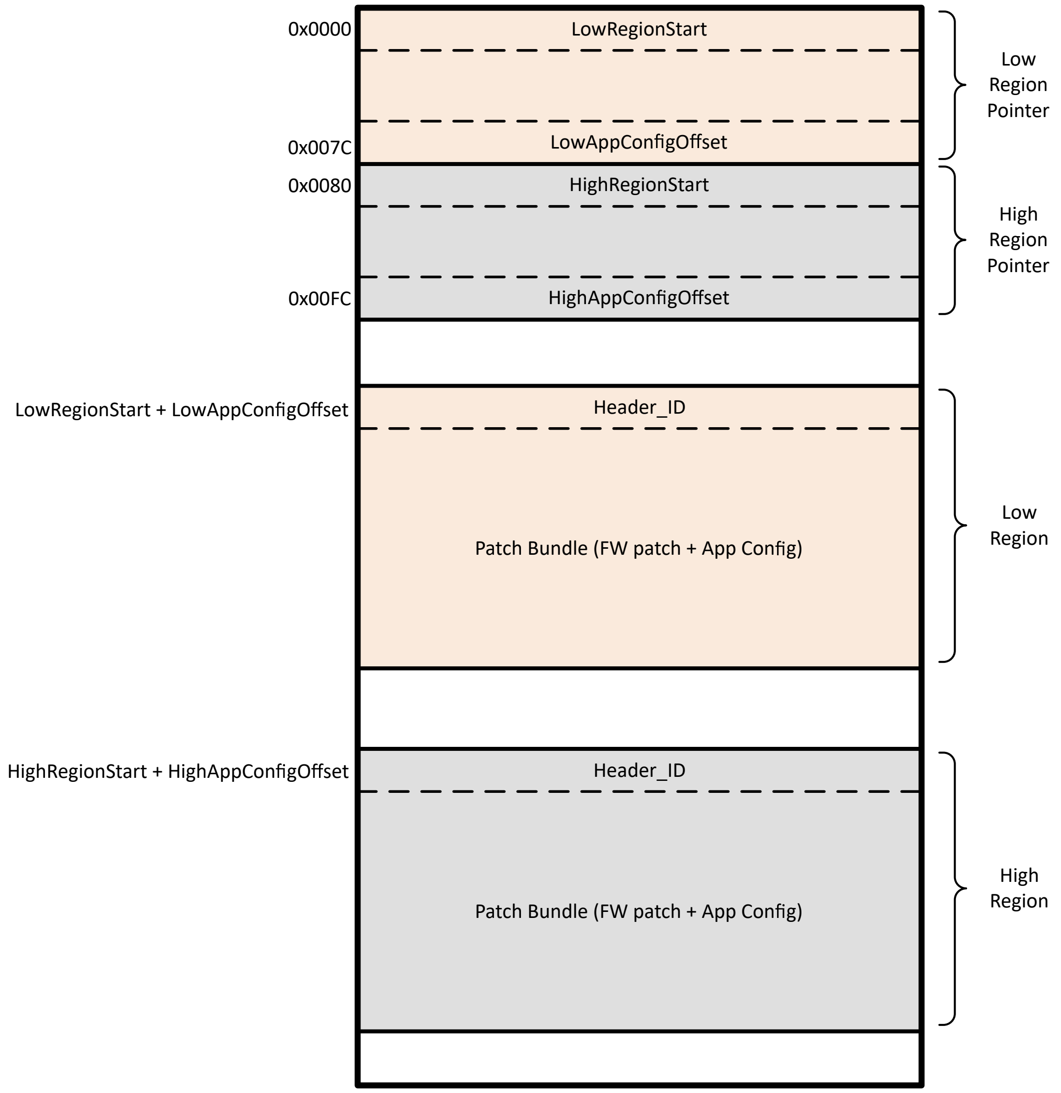


**Figure 4-3. EEPROM Memory Organization of Region-0 (Low Region) and Region-1 (High Region)**

The external EEPROM shall be programmed with a full *eeprom.bin* the first time the platform is powered up so that the region headers are set up correctly. The subsequent EEPROM-update can be executed by the external hosts by following the sequence detailed in subsequent sections. A full *eeprom.bin* can be generated using the device's application customization tool, *TPS257XX-Q1-GUI*. When you generate a full *eeprom.bin* by using the GUI, the region offsets for Region-0 and Region-1 will be set automatically.

By using 4CC commands, we can update the EEPROM from a host controller such as an MCU. In Figure 4-3, the first two blocks are headers for Region-0 and Region-1. Each header needs 128B of space. A header is used to indicate the start address of the patch bundle, and it consists of an address base and an address offset. The first four bytes of the header is the address base, and the last four bytes of the header is the address offset. Thus, the actual start address of the patch bundle is the *address base plus address offset*.

A patch bundle consists of patch bundle header, customized configuration data, and patch codes. The patch bundle for Region-0 and Region-1 are completely the same as previously mentioned.

### 4.2.2 EEPROM Update - 4CC Task Command Set

The 4CC ASCII commands listed in Table 4-2 need to be used when writing the patch bundle to the EEPROM from a host.

**Table 4-2. 4CC Task Command Set – EEPROM Update**

| Name of 4CC Command | ASCII | Input DataX Length (In Bytes) | Output DataX Length (In Bytes) | Description |
|---|---|---|---|---|
| Secure flash update initiate command | SFWi | None | 3 | SFWi prepares the device to receive upcoming data packets. PD controller shall be in FWUP mode when this task is invoked. PD controller shall NOT perform any PD operations while in FWUP mode. |
| Secure flash update data command | SFWd | 64 | 3 | SFWd Task is the primary step in the Firmware Update flow. SFWd provides PD controller with the next 64- bytes to be flashed into the I2C EEPROM. |
| Secure Firmware Update Complete | SFWs | 64 | 3 | The SFWs Task is the final step in the Firmware Update flow provided the PD controller has been provisioned for Secure Flashing using the previous SFWx commands. SFWs passes the image signature information to the PD controller for verification of the data previously received through the SFWd Task. |
| Unsigned Firmware Update Complete | SFWu | None | 3 | The SFWu Task is the final step in the Firmware Update flow provided the PD controller has not been provisioned for Secure Flashing. SFWu informs the PD controller that the Firmware Update process is complete, and causes PD controller to do the verification of the image and changing of the Active Region assuming all checks pass. |

To execute a 4CC Task, the host application shall follow the sequence below:

1. If the 4CC Task requires an input, the application shall first write the input data into the DATAx (0x09 if using I2C1 or 0x11 if using I2C2) register.
2. The application shall then write the 4CC Task characters into the corresponding CMDx (0x08 if using I2C1 or 0x10 if using I2C2) register.
3. The application shall wait until the four byte content of the CMDx register reads the following:

- 0x00 indicating that the command successfully executed.
- or, !CMD indicating that the command's execution failed.

Applications can either poll or set and use the *CMDxComplete* I2C event (for this application note, since the patch bundle has not been downloaded, the host can poll the state of the CMDx register).

If the task is successfully executed, the host can proceed to read the 3 bytes content of the DATAx register that contains the output data if the related task has output values.

### 4.2.3 EEPROM Patch Bundle Update Process

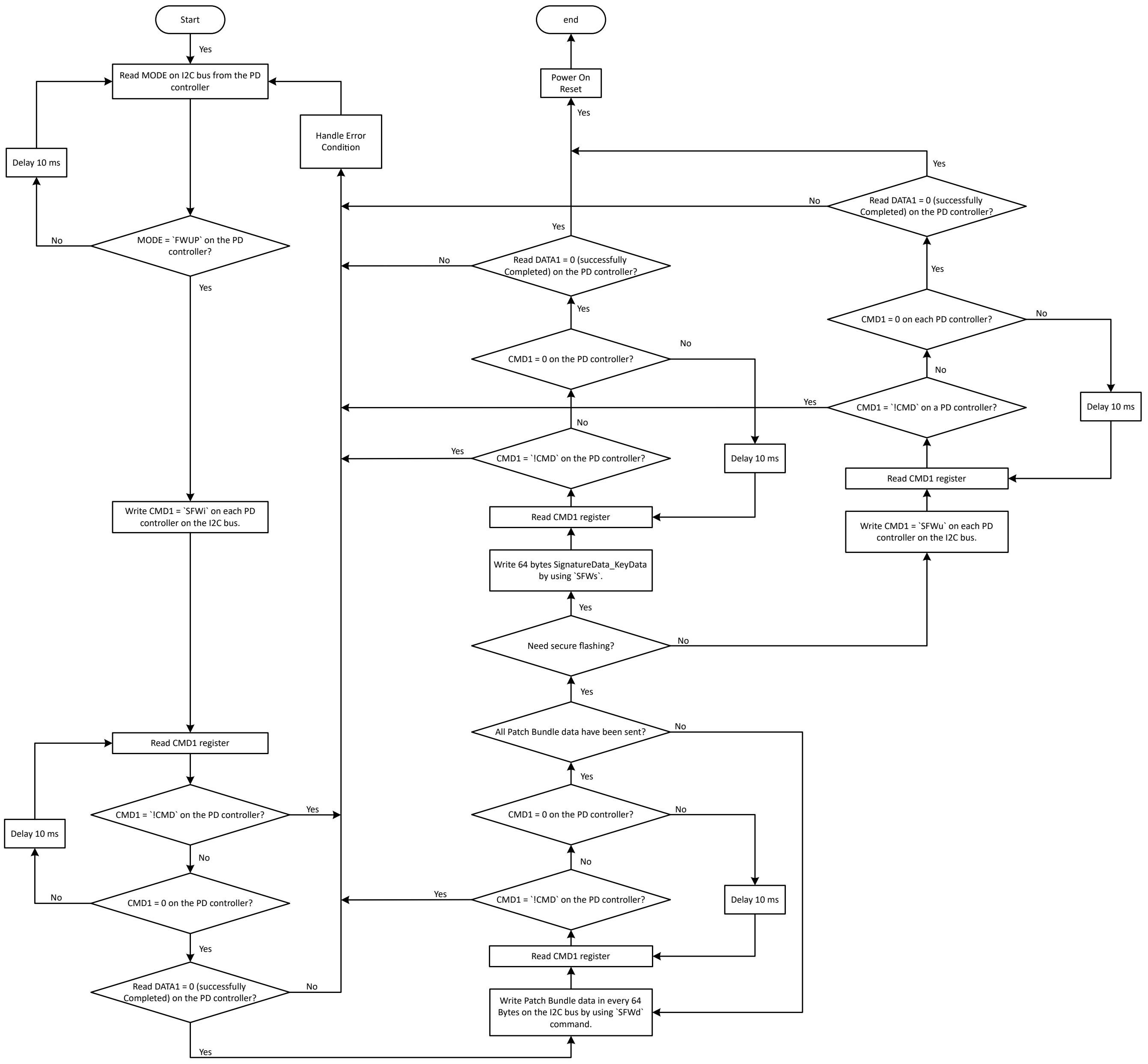

Start
Yes
Read MODE on I2C bus from the PD controller
Delay 10 ms
Handle Error Condition
No
MODE = `FWUP` on the PD controller?
Yes
Write CMD1 = `SFWi` on each PD controller on the I2C bus.
Read CMD1 register
Delay 10 ms
CMD1 = `!CMD` on the PD controller?
Yes
No
CMD1 = 0 on the PD controller?
No
Yes
Read DATA1 = 0 (successfully Completed) on the PD controller?
No
Yes
end
Power On Reset
Yes
Read DATA1 = 0 (successfully Completed) on the PD controller?
No
Yes
CMD1 = 0 on the PD controller?
No
CMD1 = `!CMD` on the PD controller?
Yes
No
Delay 10 ms
Read CMD1 register
Write 64 bytes SignatureData_KeyData by using `SFWs`.
Yes
Need secure flashing?
No
Yes
All Patch Bundle data have been sent?
No
Yes
CMD1 = 0 on the PD controller?
No
No
CMD1 = `!CMD` on the PD controller?
Yes
Delay 10 ms
Read CMD1 register
Write Patch Bundle data in every 64 Bytes on the I2C bus by using `SFWd` command.
Read DATA1 = 0 (successfully Completed) on the PD controller?
No
Yes
Yes
CMD1 = 0 on each PD controller?
No
No
CMD1 = `!CMD` on a PD controller?
Yes
Delay 10 ms
Read CMD1 register
Write CMD1 = `SFWu` on each PD controller on the I2C bus.

**Figure 4-4. Flow Diagram of EEPROM Patch Bundle Update Process**

Figure 4-4 shows the EEPROM update process. For EEPROM firmware update, the whole process shall be executed twice as per the EEPROM memory organization: the first execution is to update one region and the second execution is to update the other region. Firmware will automatically determine which region (Region-0 or Region-1) needs to be updated first. Below is the process for EEPROM patch bundle update from a host.

1. The flash update process can be initiated by the host when the device is fully functioning in the FWUP mode (External EEPROM or USB Endpoint Boot Mode), and the port shall be disabled during the update process.
2. The host will get information on which of the two regions is the current active region – The active region is the region on the external EEPROM from which the device successfully loaded the patch bundle during the current boot, and it could be either Region-0 or Region-1. In FWUP mode, it may set the active region as Region-0 as default.

3. The host shall first attempt to update the contents of the inactive region. Only after the inactive region is successfully updated, the host shall attempt to update the contents of the active region. This process ensures that a redundant and valid copy of the patch bundle is available on the external EEPROM so that the device always successful boots, thereby making the EERPOM update process *fail-safe*.
4. The host shall implement the sequence below to update the patch bundle:
    - Execute *SFWi* command to get information on which region to update.
    - Execute *SFWd* command to flash the patch bundle data to the external EEPROM through the PD controller. The patch bundle data is divided into data pages and each page has 64 bytes of data.
    - Execute *SFWs* or *SFWu* command after the patch bundle data is sent to complete the whole EEPROM update operation. If secure flashing is required, use *SFWs* to write signature data and key data (total 64 bytes). Otherwise, use *SFWu* to end the EEPROM update. Both commands will swap the region header to the other region which has not been updated if this is the first execution of the EEPROM firmware update.

## 4.3 PD Controller Patch Bundle Download

If TVSP is set to HUB/MCU Boot Mode, the device waits approximately 1 second after POR for a host to initiate a patch bundle download. 4CC Task commands can be used to do a patch bundle download such as the burst mode download (*PBMx* command, *x* is the letter corresponding to the specific command).

In burst mode download, the patch bundle is divided into 256 byte sectors and sent to the PD controller in one data transmission phase.

When the PD controller is powered on or reset in External HUB/MCU Boot Mode, the device enters patch mode ('PTCH' state in MODE (0x03) register) after roughly 850 ms and informs the host through the I2C interrupt by pulling GPIO9 low. The I2C interrupt indicates that the device has successfully entered ReadyForPatch status and is waiting for the host to transmit patch bundle data. Therefore, the initial steps for burst mode download are:

1. Power on the PD controller in HUB/MCU Boot Mode.
2. Confirm if the device is ready for a patch update by reading the INT_EVENT1 register (0x14). If any bit in the INT_EVENT1 register is 1, the device is ready for a patch update. An alternative method is to wait until an interrupt is asserted on the I2C bus and GPIO9 is pulled low.
3. Read the mode of the PD controller which is stored in the MODE register. If the mode shows ‘PTCH’, it means that the device has entered patch ready state.
4. Execute 4CC Task commands corresponding to burst mode download.

### *4.3.1 Patch Bundle Download - 4CC Task Command Set*

The 4CC ASCII commands listed in Table 4-3 shall be used when writing a patch bundle directly to the TPS257x2-Q1.

**Table 4-3. 4CC Task Command Set – Burst Mode Download**

| Name of 4CC Command | ASCII | Input DataX Length (In Bytes) | Output DataX Length (In Bytes) | Description |
|---|---|---|---|---|
| Start Patch Burst Download Sequence | PBMs | 6 | 1 | The PBMs task starts the patch loading sequence. The PBMs task initializes the firmware in preparation for a patch bundle load sequence and indicates what the patch bundle contains. |
| Patch Burst Mode Download Complete | PBMc | None | 40 | The PBMc task ends the patch loading sequence. Send the PBMc task after all patch data has been transferred. |
| End Patch Burst Mode Download Sequence | PBMe | None | 1 | The PBMe Task ends the patch loading sequence. The PBMe task instructs the PD controller to complete the patch loading process. |

### *4.3.2 Burst Mode Patch Download Process*

The following is the flow diagram detailing the process for burst mode download.

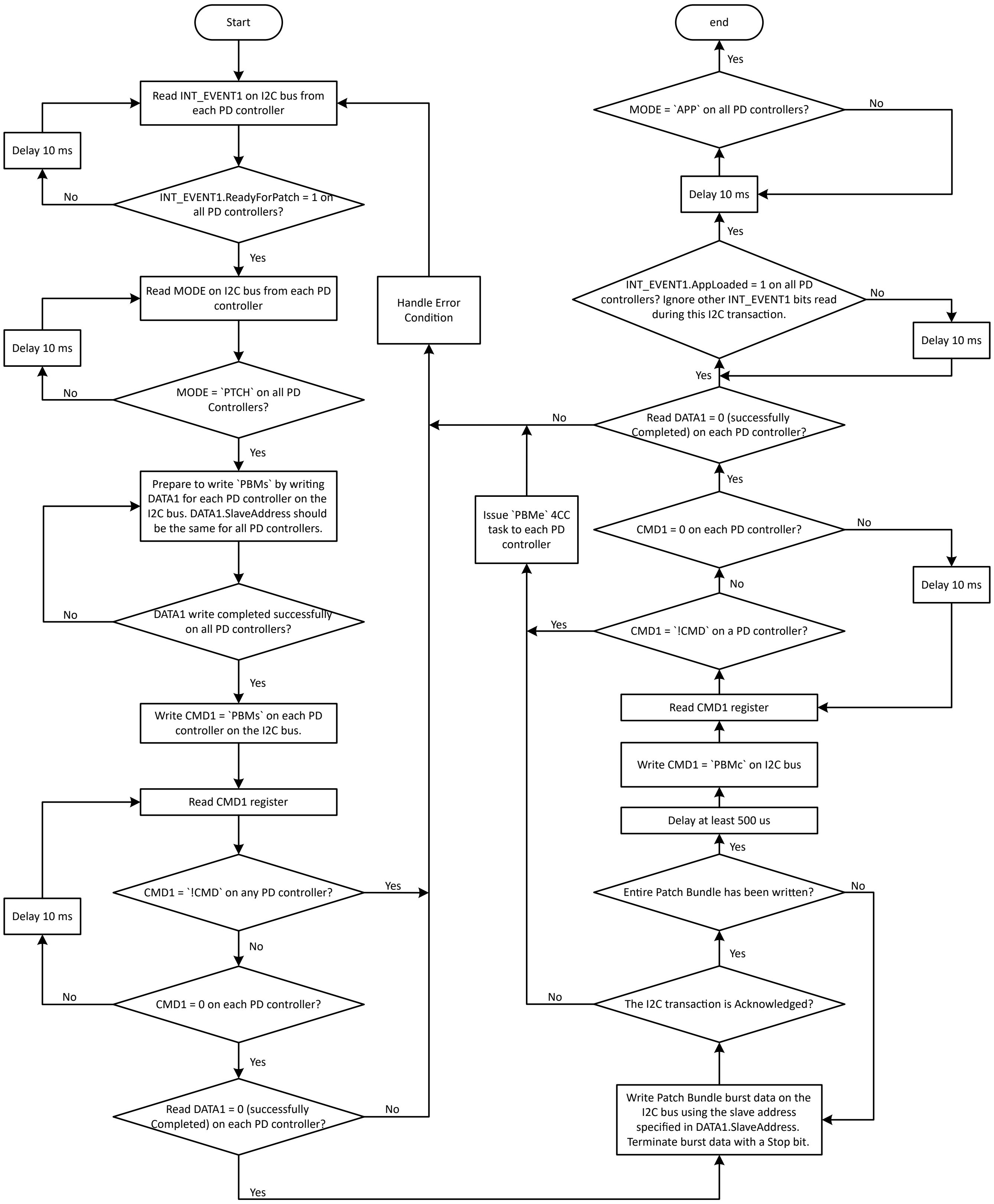

Start
Read INT_EVENT1 on I2C bus from each PD controller
Delay 10 ms
INT_EVENT1.ReadyForPatch = 1 on all PD controllers?
No
Yes
Read MODE on I2C bus from each PD controller
Delay 10 ms
MODE = `PTCH` on all PD Controllers?
No
Yes
Prepare to write `PBMs` by writing DATA1 for each PD controller on the I2C bus. DATA1.SlaveAddress should be the same for all PD controllers.
DATA1 write completed successfully on all PD controllers?
No
Yes
Write CMD1 = `PBMs` on each PD controller on the I2C bus.
Read CMD1 register
Delay 10 ms
CMD1 = `!CMD` on any PD controller?
Yes
No
CMD1 = 0 on each PD controller?
No
Yes
Read DATA1 = 0 (successfully Completed) on each PD controller?
No
Yes
Handle Error Condition
Issue `PBMe` 4CC task to each PD controller
end
Yes
MODE = `APP` on all PD controllers?
No
Delay 10 ms
Yes
INT_EVENT1.AppLoaded = 1 on all PD controllers? Ignore other INT_EVENT1 bits read during this I2C transaction.
No
Delay 10 ms
Yes
Read DATA1 = 0 (successfully Completed) on each PD controller?
No
Yes
CMD1 = 0 on each PD controller?
No
Delay 10 ms
No
CMD1 = `!CMD` on a PD controller?
Yes
Read CMD1 register
Write CMD1 = `PBMc` on I2C bus
Delay at least 500 us
Yes
Entire Patch Bundle has been written?
No
Yes
The I2C transaction is Acknowledged?
No
Write Patch Bundle burst data on the I2C bus using the slave address specified in DATA1.SlaveAddress. Terminate burst data with a Stop bit.

**Figure 4-5. Flow Diagram of Patch Bundle Update – Burst Mode Download**

Figure 4-5 shows the process for burst mode download. After the host confirms that the device has entered *PTCH* mode, the host shall implement the sequence below to download the patch bundle:

1. The host initializes the firmware in preparation for a *PBMx* load sequence and what the patch bundle contains by writing 6 bytes of data to the DATAx (0x09 if using I2C1 or 0x11 if using I2C2) register:
   a. 0x06 is first transmitted to tell the PD controller a 6-byte payload is written.
   b. Bytes 0 to 3 are the patch bundle size. In Figure 4-6, the patch bundle size is 0x3500 or 13568 bytes. Byte 4 is the DATAx.SlaveAddress you want to assign for data transmission. 0x00 or the device's I2C slave addresses (0x22/0x26 or 0x23/0x27) of the PD controller are not valid. Figure 4-6 shows a random address, 0x35, was selected and used.
   c. Byte 5 is the burst mode timeout value (LSB of 100ms). A non-zero value must be used and is recommended to always use 0x32, that gives you a 5 second window to complete the burst mode patch update.

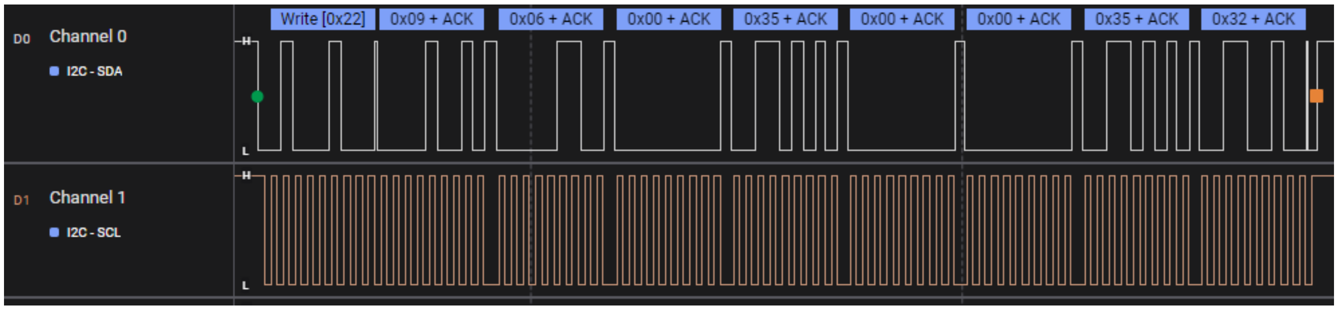


**Figure 4-6. Patch Burst Mode Initialization**

2. The host shall start the burst mode patch download process by sending the 4CC ASCII *PBMs* task command to the CMDx (0x08 if using I2C1 or 0x10 if using I2C2) register. 0x04 is first transmitted to tell the PD controller that a total of 4 bytes will be written.

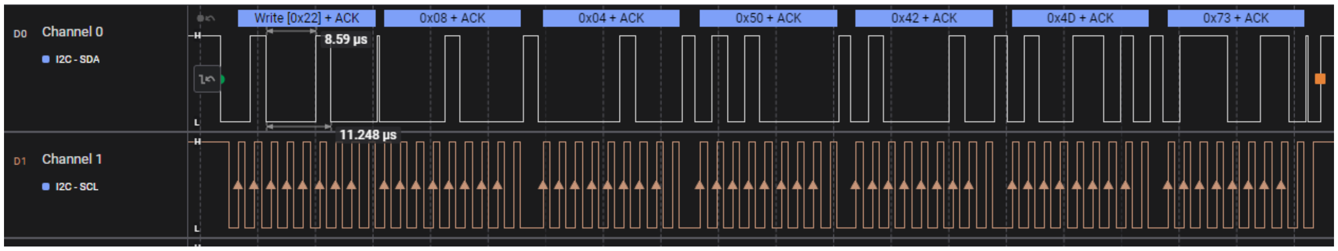


**Figure 4-7. PBMs Task Command**

3. Read and poll the CMDx register until Byte 1 is 0x00, indicating that the task processing is finished.

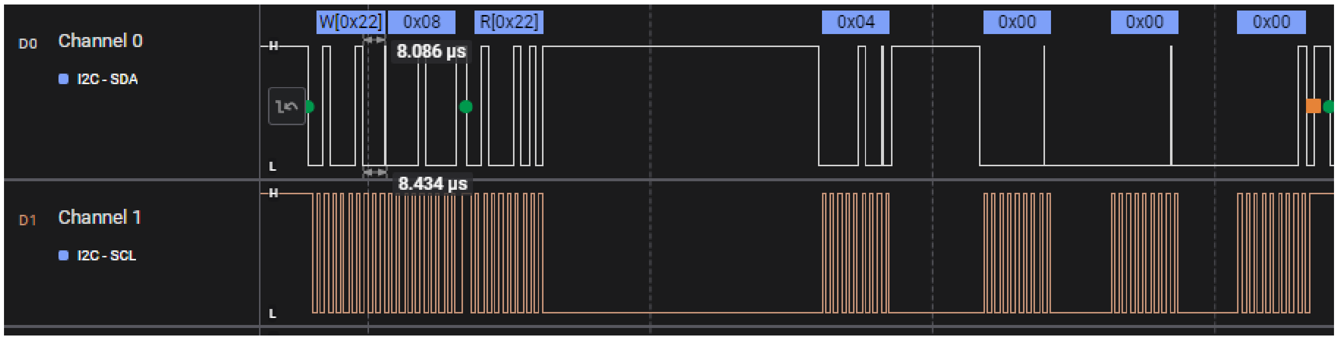


**Figure 4-8. CMDx Output - Task Processing Completed**

4. Read and poll the DATAx register until Byte 1 is 0x00, indicating that the patch initialization from step 1 was successful.

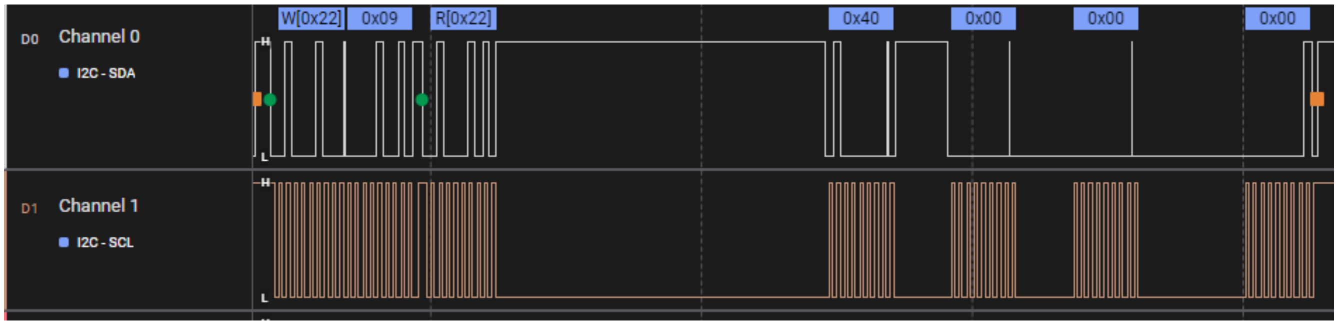
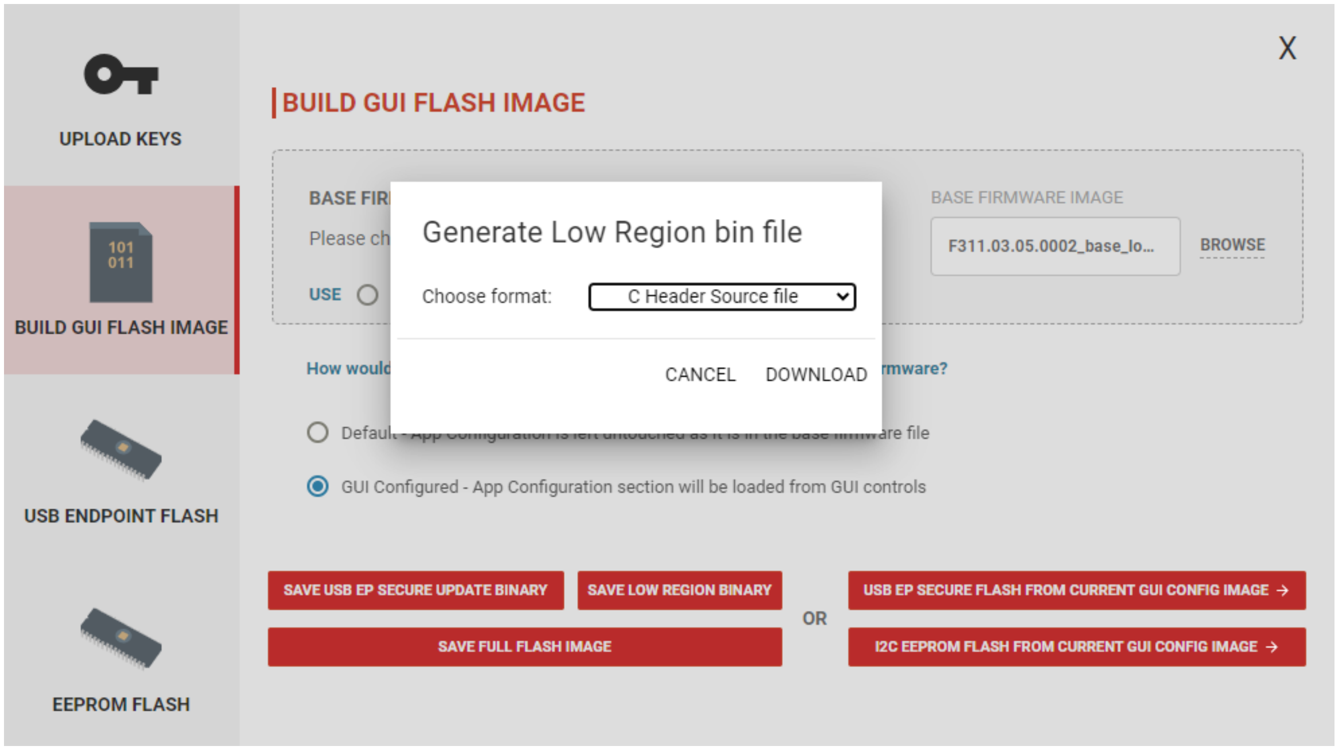


**Figure 4-9. DATAx Output - Successful Patch Initialization**

5. Transmit patch bundle data to the DATAx.SlaveAddress configured in step 1 in packets of 256 byte. The patch bundle can be generated as a C style array using the *TPS257XX-Q1-GUI* (available from v1.2.0). From the BUILD GUI FLASH IMAGE menu, select SAVE LOW REGION BINARY. Then choose the C Header Source File as the format and click SAVE. This file will also include the patch bundle size.



Generate Low Region bin file

Choose format: C Header Source file

CANCEL DOWNLOAD

BASE FIRMWARE IMAGE: F311.03.05.0002_base_lo...

Default - App Configuration is left untouched as it is in the base firmware file

[✓ GUI Configured - App Configuration section will be loaded from GUI controls]

**Figure 4-10. Generating Patch Bundle as C Array**

6. After the patch bundle data is successfully written, wait 500us and write the *PBMc* task to the CMDx register to complete the patch loading sequence. Ensure you are back to writing to the original I2C slave address of the PD controller from this step.

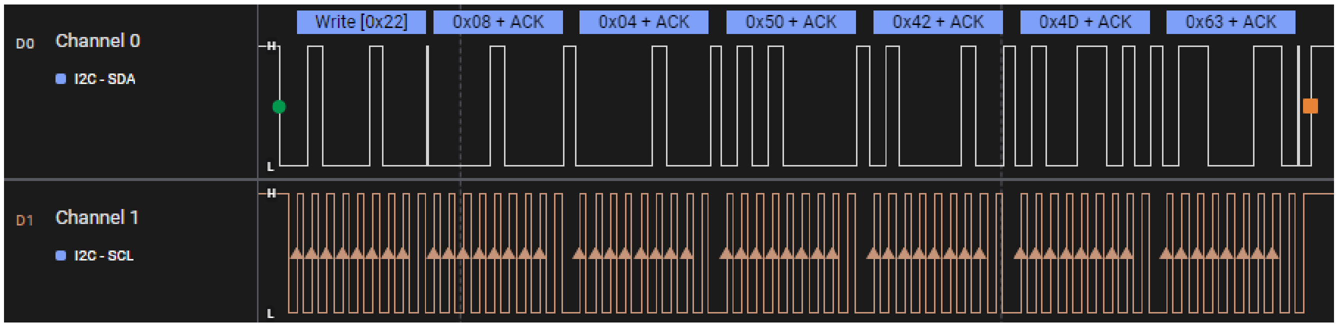


**Figure 4-11. PBMc Task Command**

7. Read and poll the CMDx register until Byte 1 is 0x00, indicating that the task processing is finished.
8. Read and poll the DATAx register until Byte 1 is 0x00, indicating that the device patch bundle download was successful. The I2C Interrupt, or GPIO9, will be released roughly 110 ms after.
9. Write the *PBMe* task to the CMDx register to end the patch loading sequence and enter 'APP' mode (optional).

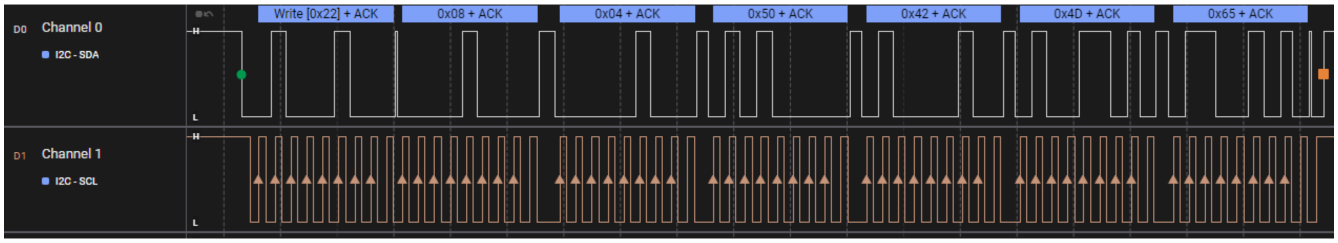


**Figure 4-12. PBMe Task Command**

10. Read the MODE register to check if the device is in APP mode which indicates that the PD controller received all patch and application configuration data and is fully functioning in the application firmware.

# Appendix A: TVSP Boot Configuration Settings

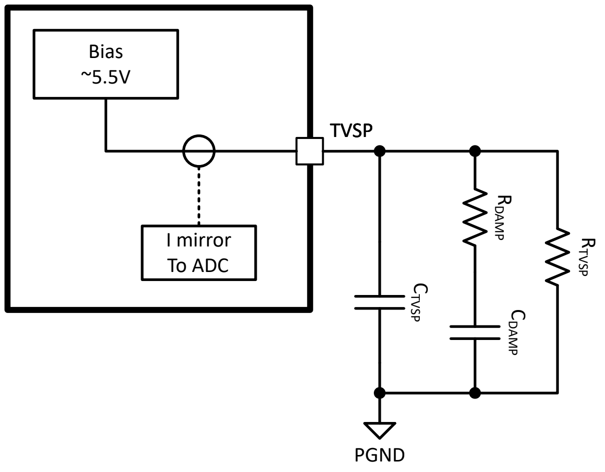


**Figure A-1. TVSP Setting**

The TVSP pin is a mufti-function pin, but for boot process, $R_{TVSP}$ is the resistor determining the boot behavior. The pin hardware connection is shown in Figure A-1. For more details, see the *$R_{TVSP}$ Configuration Settings* in the device-specific data sheet.

At power on, the resistance between the TVSP pin and PGND determines the boot method, USB PD port I2C addresses and I2C logic thresholds. During device initialization and boot, typically within 4 seconds after power on, VIN must be above 7.6V to ensure proper bias of the TVSP pin to 5.5V. Once boot is complete the device can operate over the full VIN range.

# Appendix B: Using 4CC Commands

4CC (4-byte character code) commands are a set of commands that simplify the use of the PD controller's commonly used functions. It allows the user to send a single task that manages more complex subroutines and function specific register writes for them.

The 4CC command structure is similar to a software function, where you have input arguments (Input DATAx), a function call (writing the 4CC command to the CMDx register), and a returned output (Output DATAx).

The 4CC commands that are written to the CMDx register are obtained by converting the 4-character commands to ASCII. You can use an ASCII converter to help you translate the codes (for example, the 4CC command *PBMs* is converted to *50 42 4D 73*). Please keep in mind that the commands are case-sensitive.

The CMD1 (0x08) register will have the 4CC commands written to it over the I2C1 bus. Any *Data* (Input DATAx, Output DATAx) is written to or read from the DATA1 (0x09) register. There is a second set of registers for the I2C2 bus at 0x10, for CMD2, and 0x11, for DATA2. Table B-1 provides detailed Unique Address Interface descriptions.

**Table B-1. Unique Address Interface Registers**

| Register Address | Register Name | Access | Description |
|---|---|---|---|
| 0x03 | MODE | RO | Indicates the operational state of the port. |
| 0x08 | CMD1 | RW | Command register for the primary command interface. If an unrecognized command is written to this register, it is replaced by a 4CC value of "!CMD". |
| 0x09 | DATA1 | RW | Data register for the primary command interface (CMD1). |
| 0x10 | CMD2 | RW | Command register for the secondary command interface. If an unrecognized command is written to this register, it is replaced by a 4CC value of "!CMD". |
| 0x11 | DATA2 | RW | Data register for the secondary command interface (CMD2). |
| 0x14 | INT_EVENT1 | RO | Interrupt event bit field for I2C_EC_IRQ. If any bit in this register is 1, then the I2C_EC_IRQ pin is pulled low. |
| 0x15 | INT_EVENT2 | RO | Interrupt event bit field for I2C2s_IRQ. If any bit in this register is 1, then the I2C2s_IRQ pin is pulled low. |

## Revision History

**Changes from Revision * (August 2023) to Revision A (November 2024)** **Page**

- Updated the numbering format for tables, figures, and cross-references throughout the document................. 1
- Updated document title from *TPS2576x-Q1/77x-Q1 Firmware Update* to *TPS257x-Q1/77x-Q1 Firmware Update with a Host* .................................................................................................................................. 1
- Updated topic title from *4CC Task Command Set* to *EEPROM Update - 4CC Task Command Set* .................6
- Updated topic title from *4CC Task Command Set* to *Patch Bundle Download - 4CC Task Command Set* .......8
- Updated topic title from *Patch Bundle Download Process* to *Burst Mode Patch Download Process* ................9
- Added additional steps to the Burst Mode Download process........................................................................... 9