Application Note

Common TPS25751 Use Cases and Setting Using EC

Roy Chou

ABSTRACT

The TPS25751 is a highly integrated stand-alone USB Type-C® and Power Delivery (PD) controller optimized for applications supporting USB-C PD Power. In general, the TPS25751 loads the patch file by EEPROM initially when powering on the device. If there is no EEPROM, the system needs to use EC (Embedded Controller) to issue patch file to I2Ct. The *TPS25751 Technical Reference* demonstrates the flow for Pushing a Patch Bundle Over the I2Ct Bus to Multiple PD Controllers at the same time, but can be hard to understand. This application note demonstrates the step-by-step explanation for PTCH mode to APP mode transition by using I2C command through I2Ct and also shows the commonly used function in TPS25751.

Table of Contents

Trademarks

1 Introduction

The application note focuses on using EC to issue 4CC commands to TPS25751 through I2Ct from PTCH mode to APP mode. Before the action, the article introduces the ADCINX configuration and dead battery configuration which are related to the TPS25751 initial configurations. After entering into the APP mode, there are some common used function for TPS25751 like I2Ct setting and GPIO setting.

2 ADCINX Setting

Set the appropriate ADCINX value is the first way to use I2Ct normally. The recommendation is to refer to Table 2-1 from TPS25751 data sheet. For the application, ADCINX uses the EC and so AlwaysEnableSink or NegotiateHighVoltage is recommended for dead battery configuration. Note, that if there is no EEPROM that cannot load patch file, use the SafeMode. Then, set the I2C ADDRESS INDEX that does not conflict the other I2C address on I2C bus. Here, ADCINX set #1 for the example.

After defining the ADCINX decode value, the ADCINX can get unique I2C address of TPS25751 from Table 2-2. For the setting, TPS25751 unique I2C address is 0x20.

Table 2-3 show the recommended resistance for setting the desired ADCINX decoded value. The ADCINX pins must be externally dire to the LDO_3V3 pin via a resistive divider as shown in Figure 2-1.

Table 2-1. Device Configuration Using ADCIN1 and ADCIN2

ADCIN1 Decoded Value	ADCIN2 Decoded Value	I2C Address Index	Dead Battery Configuration
7	5	#1	AlwaysEnableSink: The device always enables the sink path regardless of the amount of current the attached source is offering. USB PD is disabled until configuration is loaded. This configuration is used with an external embedded controller. The embedded controller manages the battery charger in the system when present.
5	5	#2	
2	0	#3	
1	7	#4	
7	3	#1	Negotiate High Voltage: The device always enables the sink path during the initial implicit contract regardless of the amount of current the attached source is offering. The PD controller enters the *APP*' mode, enable USB PD PHY and negotiate a contract for the highest power contract that is offered up to 20V. The configuration cannot be used when a patch is loaded from EEPROM. This option is not recommended for systems that can boot from 5V. This configuration is not valid to use with any supported battery chargers.
3	3	#2	
4	0	#3	
3	7	#4	
7	0	#1	SafeMode: The device does not enable the sink path. USB PD is disabled until configuration is loaded. Note that the configuration can put the device into a source-only mode. This is recommended when the application loads the patch from EEPROM. This configuration is recommended when the PD controller manages the battery charger when present.
0	0	#2	
6	0	#3	
5	7	#4	

Table 2-2. I2C Default Target Address for I2Ct_SCL, SDA

I2C Address Index (Decoded From ADCIN1 and ADCIN2)	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Available During Boot
#1	0	1	0	0	0	0	0	R/W	YES
#2	0	1	0	0	0	0	1	R/W	YES
#3	0	1	0	0	0	1	0	R/W	YES
#4	0	1	0	0	0	1	1	R/W	YES

Table 2-3. Decoding of ADCIN1 and ADCIN2 Pins

DIV = Rdown/(Rup and Rdown)			Without Using RUP or RDOWN	ADCINX Decode Value
MIN	Target	MAX		
0	0.0114	0.0228	Tie to GND	0
0.0229	0.0475	0.0722	N/A	1
0.0723	0.1074	0.1425	N/A	2
0.1425	0.1899	0.2372	N/A	3
0.2373	0.3022	0.3671	N/A	4
0.3672	0.5368	0.7064	Tie to LDO_1V5	5
0.7065	0.8062	0.9060	N/A	6
0.9061	0.9530	1.0	Tie to LDO_3V3	7

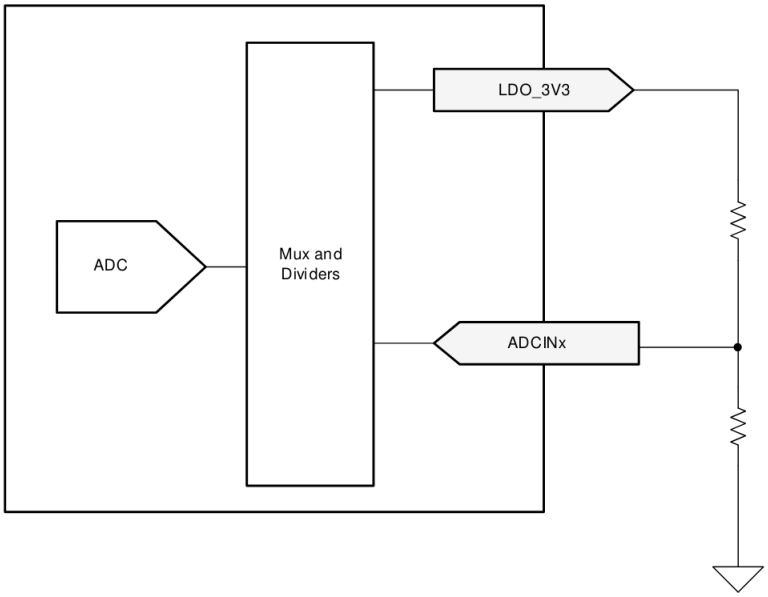


Figure 2-1. ADCINX Resistor Divider

3 Unique Address Interface Protocol

The Unique Address Interface allows for complex interactions between an I2C controller and a single PD Controller. The I2C target unique address is used to receive or respond to Host Interface protocol commands. Figure 3-1 and Figure 3-2 show the write and read protocols, respectively. The Byte Count used during a register write can be longer than the number of bytes actually written, in other words the controller can issue the stop bit without writing N bytes. Similarly, during a register read, the controller can issue the stop bit before reading all N bytes. N bytes refers to the number of bytes to be read or written.

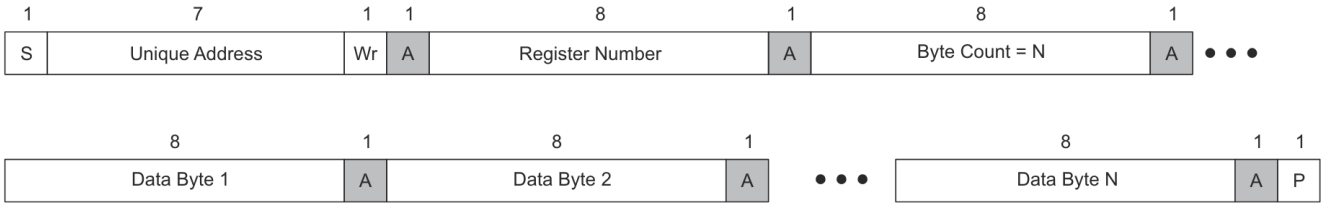


Figure 3-1. I2C Unique Address Write Register Protocol

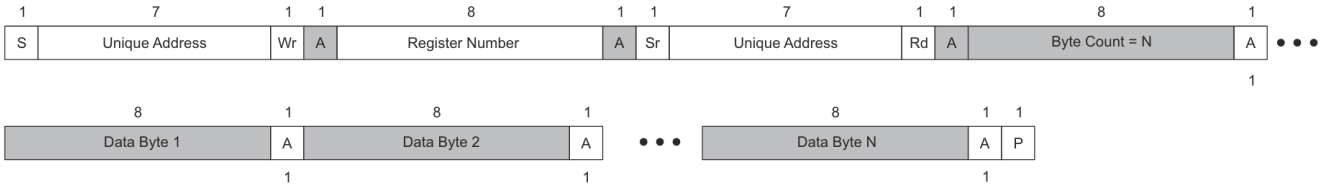
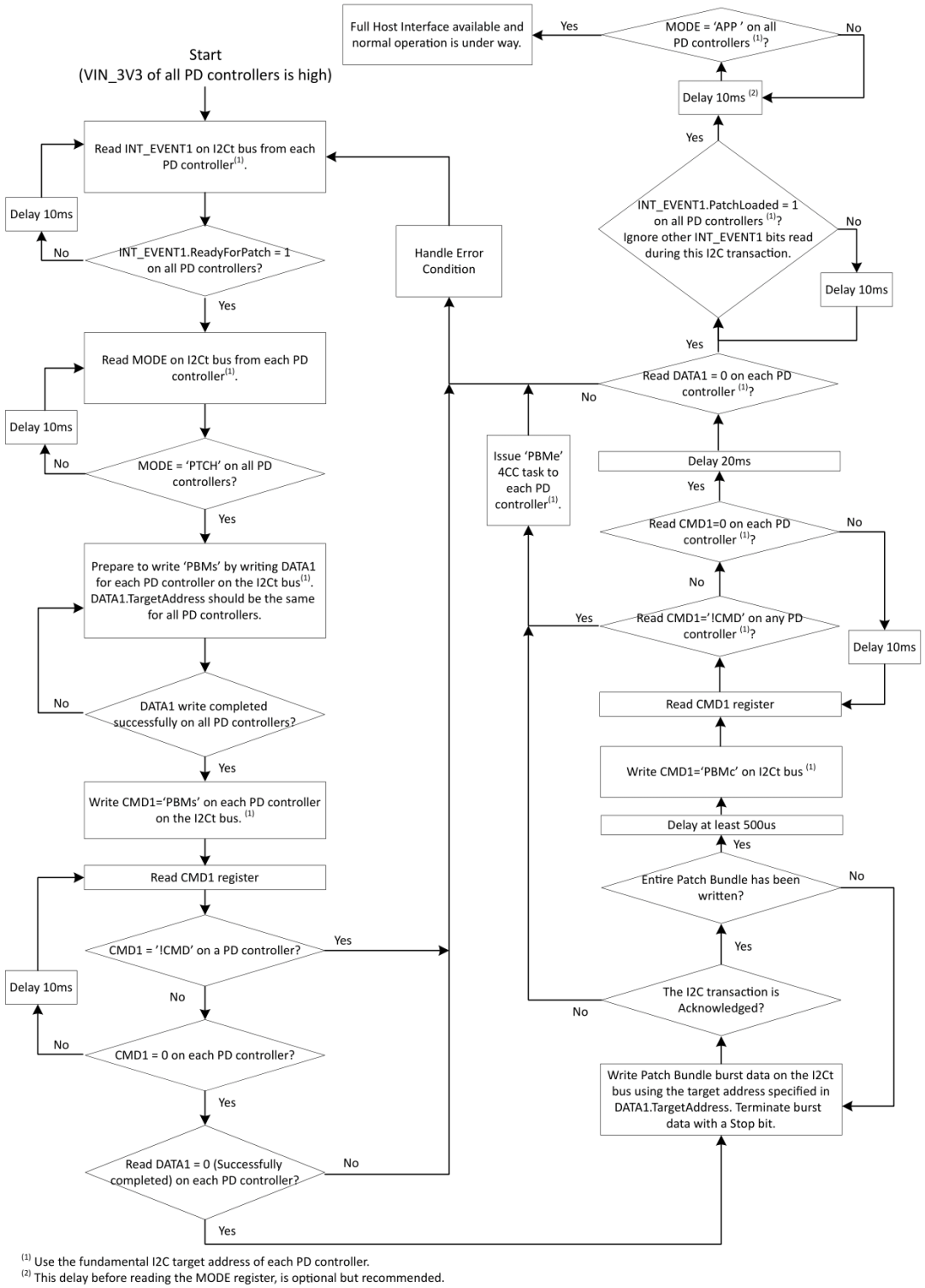


Figure 3-2. I2C Unique Address Read Register Protocol

4 PTCH Mode to APP Mode

TPS25751 Technical Reference Manual shows the Figure 4-1 flow for Pushing a Patch Bundle Over the I2Ct Bus to Multiple PD Controllers at the Same Time, but not intuitive to understand. The section demonstrates the step of mode transition from *PTCH* to *APP* to let users understand more about how to achieve transition.



(1) Use the fundamental I2C target address of each PD controller.
(2) This delay before reading the MODE register, is optional but recommended.

Figure 4-1. Pushing a Patch Bundle Over the I2Ct Bus to Multiple PD Controllers at the Same Time

4.1 Step of PTCH Mode to APP Mode

Step 1: After applying the VIN_3V3 to the TPS25751, the host can read Interrupt Event for I2C1 Register bit[81] (Offset = 14h) to know if device is read for the patch bundle for the host. The following is the example for the command and the expected result.

[0x20] + ACK (Unique Address/Wr/A)

0x14 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x0B (Byte Count)

0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 (MSB)

Step 2: Read Mode (Offset = 3h) to make sure the TPS25751 operating in *PTCH* mode. The following is the example for the command and the expected result.

[0x20] + ACK (Unique Address/Wr/A)

0x03 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x50 0x54 0x43 0x48 (‘PTCH’ in 4ASCII characters)

Step 3: Then, prepare to write *PBMs* by writing DATA1(9h) for each PD controller on the I2Ct bus. DATA1. TargetAddress needs to be the same for all PD controllers. For 4CC command, check if the DATA1 needs to be written corresponding value or just CMD1(8h) for 4CC command. For *PBMs* 4CC command needs to write DATA1(9h) first and write *PBMs* 4CC command in CMD1.

[0x20] + ACK (Unique Address/Wr/A)

0x09 + ACK (Register Number/A)

0x06 (Byte Count)

0x80 0x2C 0x00 0x00 0x30 0x32 (Byte1/2/3/4 of bundle size, I2C Target Address, Timeout value)

The bundle size can refer to next section.

Table 4-1. PBMs Task - Start Patch Burst Download Sequence

Description	**The PBMs Task starts the patch loading sequence. This Task initializes the firmware in preparation for a patch bundle load sequence and indicates what the patch bundle can contain**			
INPUT DATAX	**Bit**	**Name**	**Description**	
	Byte 6: Burst Mode Timeout			
	7:6	Reserved		
	5:0	Timeout value	Timeout value for this task. A non-zero value must be used, always use 0x32 in this field (5 seconds) (LSB of 100ms).	
	Byte 5: I2C target for downloading patch.			
	7	Reserved		
	6:0	I2C Target Address	The following target addresses are not valid: • 0x00 • The I2Ct target address of any port selected using the ADCINx pins. Refer to data-sheet.	
	Bytes 0-3: Low Region Binary bundle size in of bytes: [Byte4, Byte3, Byte2, Byte1]			
	39:32	Byte4 of bundle size		
	31:24	Byte3 of bundle size		
	23:16	Byte2 of bundle size		
	15:8	Byte1 of bundle size		
OUTPUT DATAX	**Bit**	**Name**	**Description**	
	7:0	PatchStartStatus	Status of the patch start.	
			0x00	Patch start success
			0x04	Invalid bundle size
			0x05	Invalid target address
			0x06	Invalid Timeout value
Task Completion	The PBMs Task completes after output has a valid PatchStartStatus. If MODE register (0x03) is equal to APP , then this Task can be rejected.			
Side Effects	When the 'PBMs' is successful, the second target address can be set to the input value.			
Additional Information	The host can only issue a PBMs Task to the I2Ct port of the PD controller. If the host issues PMBs a second time, then the PD controller ignores the DATAX input, restarts the burst-mode timer, and resets the pointer to the beginning of the patch space in RAM. If the MODE register is APP' indicating that the PD controller is in the APP mode, then the host can reject the PBMs Task.			

Step 4: After issuing the ‘PBMs’ DATA1, then write CMD1 = ‘PBMs’ on each PD controller on the I2Ct bus.

[0x20] + ACK (Unique Address/Wr/A)

0x08 + ACK (Register Number/A)

0x04 (Byte Count)

0x50 0x42 0x4D 0x73 (‘PBMs’ in 4ASCII characters)

Step 5: Then Read CMD1 register, the expected result is showing as below.

[0x20] + ACK (Unique Address/Wr/A)

0x08 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x00 0x00 0x00 0x00 (All ‘0x00’ are ok.)

Step6: Then Read DATA1 = 0 (Successfully completed) on each PD controller.

[0x20] + ACK (Unique Address/Wr/A)

0x09 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x00 0x00 0x00 0x00 (All ‘0x00’ are ok.)

Step7: Write Patch Bundle burst data on the I2Ct bus using the target address specified in DATA1. TargetAddress. Terminate burst data with a Stop bit.

[0x30] + ACK (Unique Address/Wr/A)

0x01 + ACK (Register Number/A)

Write the patch bundle burst data. Next section shows using GUI to generate it.

Step8: Delay at least 500us and Write CMD1=‘PBMc’ on I2Ct bus

[0x20] + ACK (Unique Address/Wr/A)

0x08 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x50 0x42 0x4D 0x63 (‘PBMc’ in 4ASCII characters)

Step9: Read CMD1 register

[0x20] + ACK (Unique Address/Wr/A)

0x08 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x00 0x00 0x00 0x00 (All ‘0x00’ are ok.)

Step10: Then Read DATA1 = 0 (Successfully completed) on each PD controller.

[0x20] + ACK (Unique Address/Wr/A)

0x09 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x00 0x00 0x00 0x00 (All ‘0x00’ are ok.)

Step11: The host can read Interrupt Event for I2C1 Register bit[80] (Offset = 14h) to know if the patch is loaded to the device or not.

Step12: Check the MODE = ‘APP’ on all PD controllers

[0x20] + ACK (Unique Address/Wr/A)

0x03 + ACK (Register Number/A)

[0x20] + ACK (Unique Address/R/A)

0x04 (Byte Count)

0x41 0x50 0x50 0x20 (APP in 4ASCII characters)

4.2 Step of Generating Low Region Binary

Use USBCPB Application Customization Tool can easily generate the low region binary file. Select the system configuration and condition following the questionnaire. Then export the low region binary file shown as Figure 4-2. After generating the low region binary file, the host can see the content shown as Figure 4-3 for the bundle size and this is corresponding with the Section 4.1 step3.

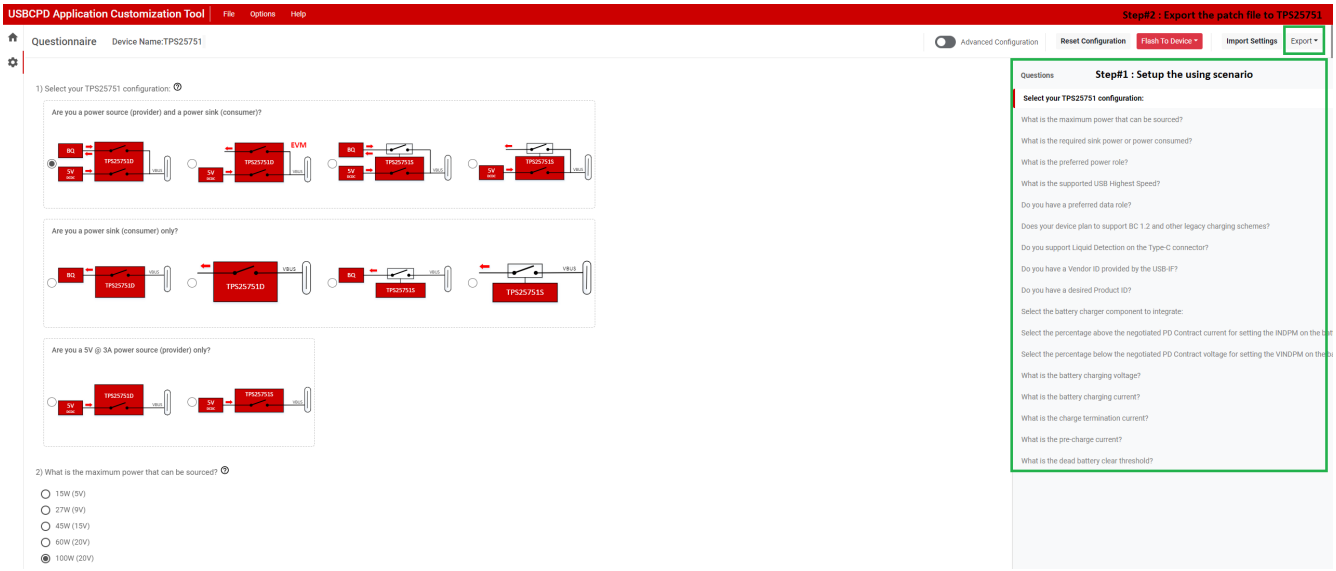


Figure 4-2. Setup for Generating the low Regional Binary File

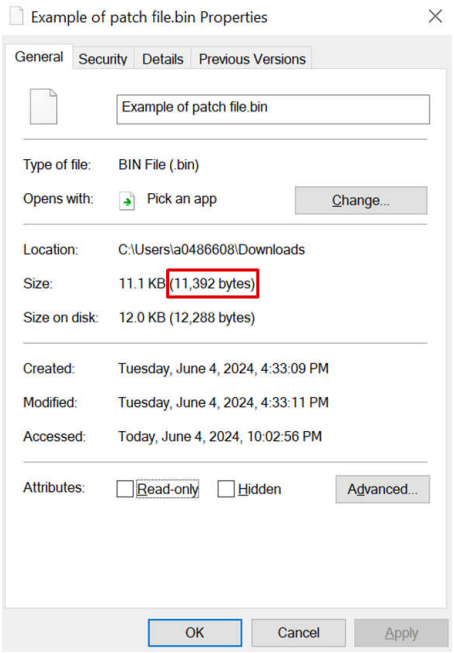


Figure 4-3. Bundle Size Example

5 Dead Battery Configurations

Dead battery mode entering rule is that TPS25751 only has power from the VBUS which means that the TPS25751 can be turned on by the VBUS pin in this mode. And the mode register (Offset = 3h) can show the BOOT indicating that the PD controller is booting in dead battery. When using EC with TPS25751, set dead battery configuration as AlwaysEnableSink or NegotiateHighVoltage is recommended.

If system wants to take the TPS25751 out of the dead battery mode after there is VIN_3V3 and PP5V on TPS25751D, write over I2C to clear the PD‘s dead battery flag by using 4CC DBfg command to let TPS25751D rely on VIN_3V3 and PP5V instead of VBUS. After clearing dead battery flag, PD can get power from VIN_3V3. PD can now be able to source power from PP5V, but cannot change the active PD contract on the port. PD cannot automatically swap to source power role unless commanded to do so.

If TPS25751 has VIN_3V3 and PP5V first, the TPS25751 does not enter in dead battery mode and does not need to clear the dead battery flag.

6 Interrupt Event, Mask, Clear for I2Ct IRQ

For interrupt event, interrupt mask and interrupt clear, refer to 0x14h, 0x16h and 0x18h. For I2Ct_, use 0x16 interrupt mask to let I2Ct_ report low if the requirement event occurred. For example, if we set 0x16[3] = 1 which means that we mask off the *Plug Insert or Removal* event. If there is Type-C port that has the event of *Plug Insert or Removal*, I2Ct_ can report low. If the system wants to recover I2Ct_ state status as high, the system need to clear the corresponding bit in 0x18h.

In general, if the user does not clearly understand which interrupt action corresponds the which event, the interrupt action can mask off all to bit on 0x16h but usually recommends EC mask only the events needed or only the events that checks for when IRQ asserted low. When the I2Ct_ report low, check the 0x14h interrupt event.

7 GPIOx Function

GPIOx is commonly used by users. GPIOs configurations can refer to 0x5C[12:0] register. For example, if 0x5C[12:0] = 0110000001101b, it can know the GPIO type in below table. But GPIO8 and GPIO9 are not available in TPS25751. And GPIOx status can be read by using 0x72[7:0] and 0x72[12] if the GPIOx configured as output type. It cannot be read when GPIOx configured as input type.

Table 7-1. GPIOx Configuration Reset Value 0x5C[12:0]

GPIOx	12	11	10	9	8	7	6	5	4	3	2	1	0
Data	0	1	1	NA	NA	0	0	0	0	1	1	0	1
Type	I	O	O	NA	NA	I	I	I	I	O	O	I	O

GPIOx can be configured as input type and output type with corresponding mapped event. USBCPB Application Customization Tool can easily generate the customized GPIOx setting in low region binary file. For example, if GPIOx needs to configure GPIO0 as output type with plugevent_port1 and initial value is 0. When the event occurring, GPIO0 report the 1. After finishing the setup and export the low region binary and issue the file when PTCH mode to APP mode transition.

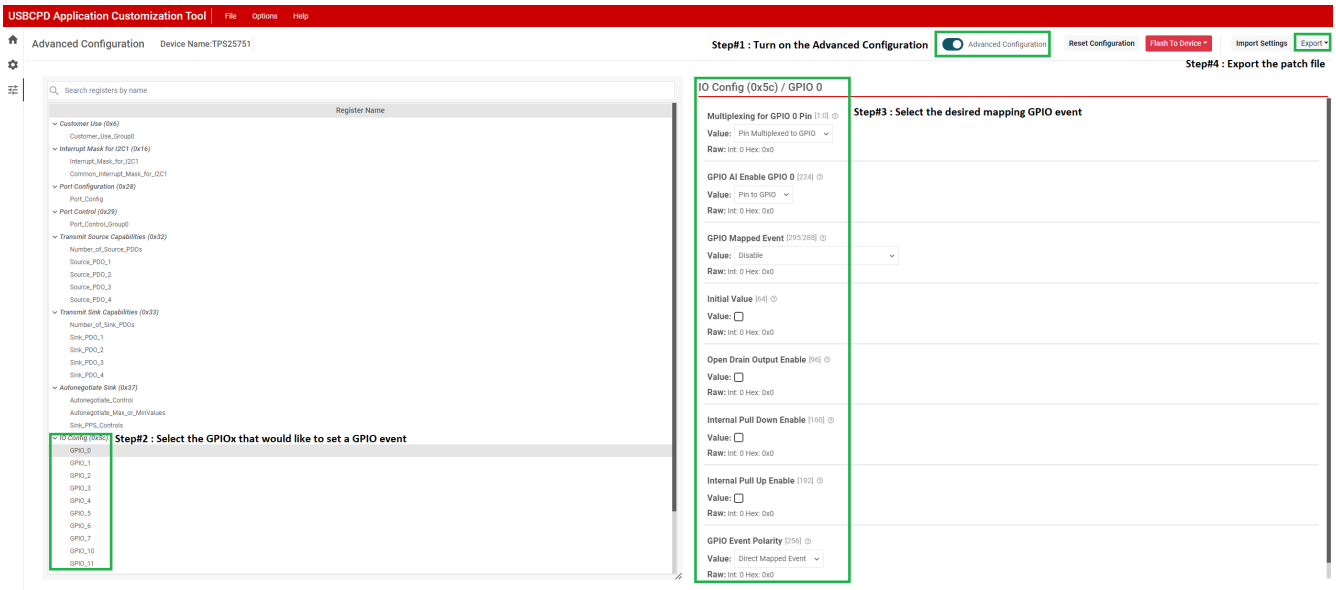


Figure 7-1. GPIOx Setup Using USBPCB Application Customization Tool

For GPIOx type, currently, these values are fixed by interrupt event and cannot be changed the type which means that TPS25751 does not allow use for general GPIOs to be input or output and read the status. Instead, based on which GPIO event you configure, the PD can automatically change the GPIO settings of the pin to be input or output.

8 4CC Command

Full name of 4CC is a four characters command. To use these 4CC commands, the ASCII command needs to be converted to hex and written to register 0x08. After the command is written, the command can be *complete* when register 0x08 clears. Then the user can check register 0x09 for the *Output DataX* to see if the task completed successfully. TPS25750 TRM shows the detailed DataX for task return code to see if the task is successfully written or not.

9 Summary

This application note describes how to bring up the TPS25751 in early stage.

10 References

- Texas Instruments, *GPIO*, E2E™ design support forum.
- Texas Instruments, *Dead battery*, E2E™ design support forum.
- Texas Instruments, *PTCH to APP*, E2E™ design support forum.
- Texas Instruments, *Salae code*, E2E™ design support forum.
- Texas Instruments, *Mask off*, E2E™ design support forum.

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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

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