Application Note

Controlling TPS55288 and TUSB1044 With TPS65992S PD Controller

ABSTRACT

This application note explains how to use the TPS65992S USB Type-C®/PD controller I2C controller feature to control two key companion devices in a USB-C Power Delivery system: the TPS55288 buck-boost regulator and the TUSB1044 linear redriver (USB Type-C crosspoint switch). This app note describes the necessary TPS65992S configuration for automated I2C control, the register settings for TPS55288 to manage VBUS voltages, and the settings for TUSB1044 to handle USB3/DisplayPort lane switching. Special attention is given to multi-byte I2C transactions (for example, setting the TPS55288 output voltage reference), the interpretation of TUSB1044 control register bits (CTLSEL, FLIP_SEL, EQ_OVERRIDE, and so on), and maintaining proper device behavior during attach, detach, and DisplayPort Alternate Mode events. Data sheet-backed explanations of each register value are provided to clarify the logic behind the configurations. The result is a coordinated design where the PD controller orchestrates power and data path changes seamlessly without needing an external microcontroller.

Table of Contents

Trademarks

1 Introduction

The TPS65992S is a single-port USB Type-C and USB Power Delivery (PD) controller that integrates various digital interfaces for communication with other devices. Notably, TPS65992S provides three I2C ports: I2C1 and I2C2 function as I2C peripheral interfaces (for host or EC communication), and *I2C3 serves as an I2C controller*. This I2C controller port allows the TPS65992S to directly control external components on the board. In the design discussed here, I2C3 is used to manage two key devices: the *TPS55288* buck–boost converter and the *TUSB1044* linear redriver.

The TUSB1044 is a USB Type-C *Alternate Mode redriver-switch* (a linear repeater for high-speed signals) that supports SuperSpeed data rates up to 10Gbps and is protocol-agnostic. This means TUSB1044 can pass USB3.1 Gen2 signals and also route DisplayPort lanes over the USB-C connector in DisplayPort Alternate Mode. The device can be configured either through fixed pin settings or through an I2C interface for dynamic control. The internal registers allow control of operating modes, flipping of high-speed lane mapping, equalization (EQ) settings, and other features (see the TUSB1044 data sheet register map for details). Using the TPS65992S as an *I2C controller*, the system can command the TUSB1044 to switch between USB3-only operation and various DisplayPort Alt Mode configurations depending on the cable orientation and negotiated mode.

Table 1-1. TUSB1044 Register Map

Offset	Acronym	Register Name
Ah	General_1	General Registers 1
Bh	General_2	General Registers 2
Ch	General_3	General Registers 3
10h	UFP2_EQ	UFP2 EQ Control
11h	UFP1_EQ	UFP1 EQ Control
12h	DisplayPort_1	AUX Snoop Status
13h	DisplayPort_2	DP Lane Enable/Disable Control
1Bh	SOFT_RESET	I2C and DPCD Soft Resets
20h	DFP2_EQ	DFP2 EQ Control
21h	DFP1_EQ	DFP1 EQ Control
22h	USB3_MISC	Misc USB3 Controls
23h	USB3_LOS	USB3 LOS Threshold Controls

The TPS55288 is a synchronous four-switch *buck–boost converter* designed for USB Power Delivery source applications. The TPS55288 can regulate the output voltage *below, equal to, or above* the input voltage, supporting a wide input range (2.7V to 36V) and an output range from 0.8V up to 22V. The TPS55288 features a 10-bit DAC for setting the internal reference voltage that determines the output level, with 1 LSB ≈ 1.129mV. Through the I2C interface, the output voltage and current limit can be programmed, enabling compliance with USB PD requirements (including Programmable Power Supply [PPS]). In practice, this converter can deliver up to 100W (for example, 20V at 5A) from a typical 12V source. Key configuration registers on the TPS55288 include the *REF* registers (0x00 and 0x01) which set the internal reference voltage, the *IOUT_LIMIT* register (0x02) for current limit, and others for slew rate, feedback selection, and so on. For example, the REF registers (0x00/0x01) form a 10-bit value that programs the converter reference voltage; writing the appropriate value adjusts the output voltage according to the internal feedback ratio.

Table 1-2. TPS55288 Register Map

Address	Acronym	Register Name
0h, 1h	REF	Reference Voltage
2h	IOUT_LIMIT	Current Limit Setting
3h	VOUT_SR	Slew Rate
4h	VOUT_FS	Feedback Selection
5h	CDC	Cable Compensation
6h	MODE	Mode Control
7h	STATUS	Operating Status

This application note describes how the TPS65992S PD controller I2C3 controller interface is configured to control the TPS55288 and TUSB1044 in a coordinated manner. We explain the necessary settings in the TPS65992S Application Customization Tool, the configuration of I2C command sequences (with proper peripheral addresses and register addresses), and the event-triggered I2C transactions that allow these devices to function seamlessly during PD negotiation and USB-C Alternate Mode operation. Each section below covers the setup for the power converter and redriver, respectively, followed by a summary of all I2C events used in this design.

2 I2C Controller Configuration in the TPS65992S Application Tool

For the TPS65992S to manage the TPS55288 and TUSB1044, the I2C3 controller port must be configured with the correct peripheral addresses and a mapping of command indices to those devices. Using the TPS65992S Application Customization Tool (a GUI for configuring the PD controller firmware), each external I2C peripheral is added with the 7-bit address and assigned an *Address Index* (an identifier used in the PD firmware). The tool also allows defining sequences of I2C register read and write operations tied to specific PD events (such as power on, attach, contract negotiation, and so on).

In our design, the I2C3 Controller Port is set up as follows:

- *TPS55288 (Buck-Boost Converter)* – 7-bit I2C address *0x74*. This device is added as *Address0* (Address Index = 0) in the PD controller I2C controller configuration. We reserve register indices 0 through 8 in the PD controller I2C event table for commands targeting the TPS55288. (Note: 0x74 is the default I2C address of TPS55288 when the device I2CADD pin/mode bit is 0, which matches our configuration.)
- *TUSB1044 (USB-C Redriver)* – 7-bit I2C address *0x12*. This device is added as *Address1* (Address Index = 1) in the PD controller's configuration. We allocate register indices 9 through 28 in the event table for commands to the TUSB1044. (The TUSB1044 address in I2C mode is determined by pin strapping; here the address is configured as 0x12.)

Once the two peripherals and the addresses are defined, the TPS65992S firmware can direct I2C transactions to the correct device using the assigned Address Index. Figure 2-1 shows an example from the TPS65992S GUI: *Address0* corresponds to TPS55288 at 0x74, and *Address1* corresponds to TUSB1044 at 0x12. With the address map in place, we can then create entries in the PD controller I2C controller Event table. Each entry (indexed by the *register index* mentioned above) specifies an event trigger, the target peripheral (by Address Index), the register address on that peripheral, data bytes to write (or read), and whether the command is orientation-independent or not. In the following sections, we detail the command sequences configured for each device.

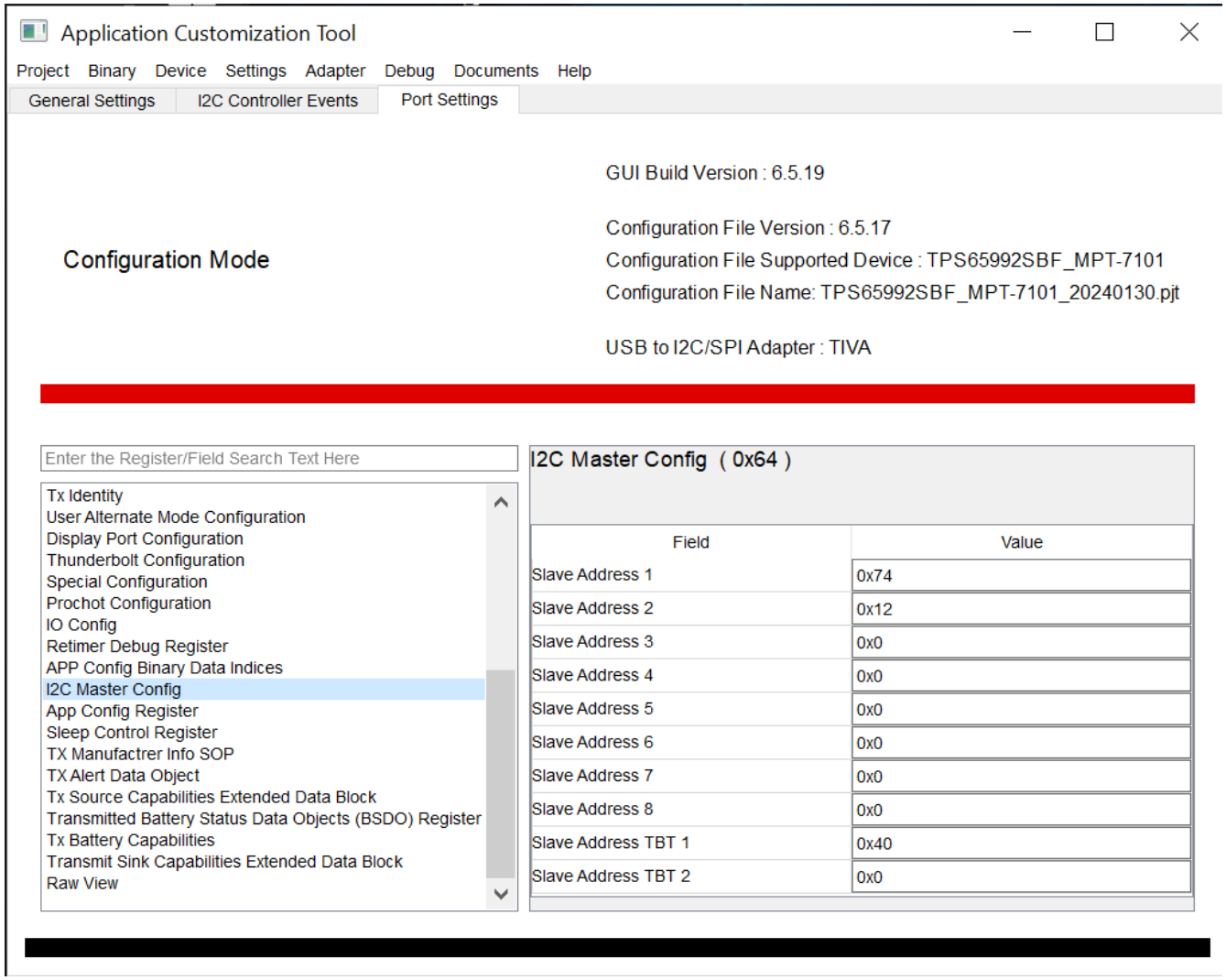

Figure 2-1. TPS65992S GUI Setting On I2C Controller Configuration

3 Configuring the TPS55288 Buck-Boost Converter Through I2C3

For the TPS55288, the goal is to program the converter output behavior based on PD negotiations, and to maintain safe defaults at power-up and disconnect. The TPS65992S event table (indices 0–8 allocated to TPS55288) is set up accordingly. Key register settings on the TPS55288 are derived from the data sheet:

- *REF (0x00/0x01):* Sets the internal reference voltage which, in conjunction with the feedback configuration, determines V_{OUT}. This is a 10-bit value split across two registers (0x01 is MSB, 0x00 is LSB). One LSB corresponds to about 1.129mV of reference voltage. By adjusting REF, the output can be changed from a minimum of about 45mV reference (0x0000, yields the lowest output) up to 1.2V reference (0x03FF, yields maximum output). The *default* REF value after reset is 0x00D2 (LSB=0xD2, MSB=0x00), which corresponds to about 282mV. The TPS55288 default feedback ratio is set for internal sensing (see V_{OUT}_FS below), so 282mV reference results in approximately 5V at the output (more on this below).
- *V_{OUT}_FS (0x04):* The Feedback Selection register controls whether the converter uses an internal resistor divider for V_{OUT} sensing or an external feedback network. The feedback selection register selects the internal feedback ratio if used. By default, V_{OUT}_FS = 0x03 (binary 0000_0011), which means *internal feedback enabled* (bit7 = 0) and the internal feedback ratio set to 0.0564 (bits1:0 = 11b). A ratio of 0.0564 corresponds to a 20× gain from the reference to the output (since 0.0564 × 20 ≈ 1), allowing the output to reach about 20V when the reference is at the maximum of about 1.129V. In simpler terms, with the default ratio, a 282mV reference yields about 5V output, and a 1.129V reference yields about 20V output, establishing the range needed for PD profiles.
- *MODE (0x06):* The Mode Control register contains various control bits; notably bit7 is *OE (Output Enable)*, which must be set to 1 to turn on the converter output. Other bits configure features like hiccup protection, discharge on shutdown, I2C address selection, and light-load mode (PFM/FPWM). Of interest, bit2 (I2CADD) selects the I2C address (0 for 0x74, 1 for 0x75), and bit0 (MODE) selects whether certain settings are taken from external resistors or internal registers. In our design, we leave I2CADD = 0 (using 0x74 address) and use the default control method for VCC and PFM (external resistor control, bit0 = 0), since the hardware design provides the proper defaults. We, however, enable the output (OE) through I2C and verify other bits are appropriately set for operation.

Using these registers, the TPS65992S I2C event sequences are configured as follows:

Power-On Initialization

On a *Power-On Reset* (PoR) of the PD controller, we initialize the TPS55288 to a known safe state so that the PD controller is ready to supply 5V by default (the standard USB-C default voltage) once the port is active. The TPS65992S firmware triggers several I2C writes to the TPS55288 immediately after PD controller reset:

- *Set Reference Voltage:* Write *0xD2* to TPS55288 register 0x00 (LSB of REF). This, combined with the default MSB (0x00), sets the REF value to 0x00D2. As noted above, 0x00D2 corresponds to about 282mV reference, which (with the default feedback ratio) equates to about 5V output. This effectively programs the converter for 5V output as a starting point.
- *Set Feedback Configuration:* Write *0x03* to register 0x04 (VOUT_FS). This explicitly verifies that the converter uses the internal feedback network and the 0.0564 ratio (which is the default reset state). Writing 0x03 here is mainly for completeness – this confirms the feedback mode in case any non-default configuration was latched or to guard against any uncertainty at start-up. This keeps the internal feedback enabled and selects the ratio that allows the full PD voltage range (up to 20V) through internal scaling.
- *Enable Converter Output:* Write *0xA0* to register 0x06 (MODE). This sets bit7 (OE) = 1 to turn on the output, and also has bit5 = 1 (keeping hiccup mode enabled as default). The value 0xA0 (binary 1010_0000) essentially flips OE to 1 while leaving other bits at the desired values (for instance, 0x20 was the reset value with OE=0, HICCUP=1, and so on, and 0xA0 changes OE to 1). After this write, the TPS55288 is enabled and regulating to about 5V output.

These three writes (to 0x00, 0x04, 0x06) are done in sequence during the PD controller initialization. By configuring the power converter at PoR, the design maintains that *even before any USB-C connection is made, the TPS55288 comes up in a default 5V standby output state* (or ready to provide 5V when enabled). This is important for compliance with the USB-C standard which requires 5V present on VBUS initially for attachment. In our case, the actual enabling of VBUS to the connector is controlled by the PD controller power-path switch, but the TPS55288 output is set to the correct level and turned on internally.

(The exact numeric values chosen – 0xD2, 0x03, 0xA0 – correspond to the desired start-up configuration as derived from the TPS55288 data sheet. 0xD2 (210 decimal) for REF LSB is the code for about 5V as explained, 0x03 for VOUT_FS selects internal feedback with 0.0564 ratio, and 0xA0 enables output. Designers can refer to the TPS55288 Register Maps in the data sheet for detailed bit definitions.)

Dynamic PDO Voltage Adjustment (Source PDO Negotiation)

With the converter initialized, the PD controller next needs to adjust the TPS55288 output *each time a new PD contract is negotiated*. The TPS65992S is configured to issue I2C writes to the REF register whenever a Source PDO is selected by the connected device (for example, whenever the PD source voltage changes). In our example, suppose the system offers four fixed PDOs at 5V, 9V, 15V, and 20V. We define four event triggers in the TPS65992S firmware. (For example, *SRC_PDO1_NEGOTIATED*, SRC_PDO2_NEGOTIATED, and so on), each mapped to writing the appropriate value to the TPS55288 REF registers.

- *5V Contract (PDO1):* Write *0xD2 0x00* to TPS55288 register 0x00. This is a two-byte write (0x00 is the starting register for REF LSB). The data bytes 0xD2 0x00 set REF = 0x00D2 (LSB=D2h, MSB=00h) which, as discussed, corresponds to about 5.0V output. *(In the PD tool, Data Length=3 for this entry: 1 byte for reg address + 2 data bytes.)*
- *9V Contract (PDO2):* Write *0x9A 0x01* to register 0x00 (two-byte data, setting REF = 0x019A). In the TPS55288 internal DAC code, 0x019A corresponds to roughly 9V. Specifically, 0x019A = 410 decimal; 410 × 1.129mV ≈ 463mV reference. With 0.0564 feedback ratio, this yields V_{OUT} ≈ 463mV / 0.0564 ≈ 8.21V. However, the converter output slightly overshoots to meet 9V at load (the exact values are chosen based on calibration and experimentation). According to typical settings, 0x019A is the code used for 9V in this design.
- *15V Contract (PDO3):* Write *0xC5 0x02* to register 0x00 (REF = 0x02C5). 0x02C5 = 709 decimal; 709 × 1.129mV ≈ 800mV reference. Divided by 0.0564, this yields about 14.2V. Again, the code is selected to produce approximately 15V under expected conditions (taking into account load and converter tolerance).
- *20V Contract (PDO4):* Write *0xBF 0x03* to register 0x00 (REF = 0x03BF). 0x03BF = 959 decimal; 959 × 1.129mV ≈ 1.083V reference. With the 0.0564 ratio, this gives about 19.2V. This is intended to hit about 20V at no-load or nominal load (the TPS55288 feedback and reference may be trimmed such that 0x03C0 might overshoot slightly above 20V, so 0x03BF is used to stay at or just below 20V typical). Essentially, 0x03BF is the calibrated code for 20V_{in} our setup.

Each of these writes targets the TPS55288 REF registers to dynamically change V_{OUT} on the fly. The PD controller issues the appropriate command immediately after the PD contract is established. This way, as soon as a sink requests a higher voltage (for example, 15V), the TPS65992S writes the new REF value and the TPS55288 slews the output to the requested level. The slew rate of the output can be controlled by the TPS55288 VOUT_SR register (0x03) if needed, but in our design we rely on the converter default slew, which is typically configured to meet PD timing requirements.

Detach and Reset Behavior

When the USB-C cable is unplugged or the sink device is detached, return the power supply to a safe state. Write 0xD2 0x00 to registers 0x00 (LSB) and 0x01 (MSB). This two-byte write fully resets the reference voltage to the desired 5V default setting, eliminating the possibility of residual high-voltage conditions. Regardless, the outcome is that the TPS55288 output *drops to a safe level (approximately 5V or below)* when a disconnect occurs. This prevents any residual high voltage on VBUS after the cable is removed.

By setting up these I2C command events for the TPS55288, all power supply adjustments are automated by the TPS65992S firmware without the need for any external MCU intervention. The PD controller monitors the USB-C state and PD messages and triggers the appropriate pre-programmed I2C writes. This maintains that as PD contracts change, the converter seamlessly provides the requested voltage, and when no contract is active (detach or idle), the converter is in a known default state.

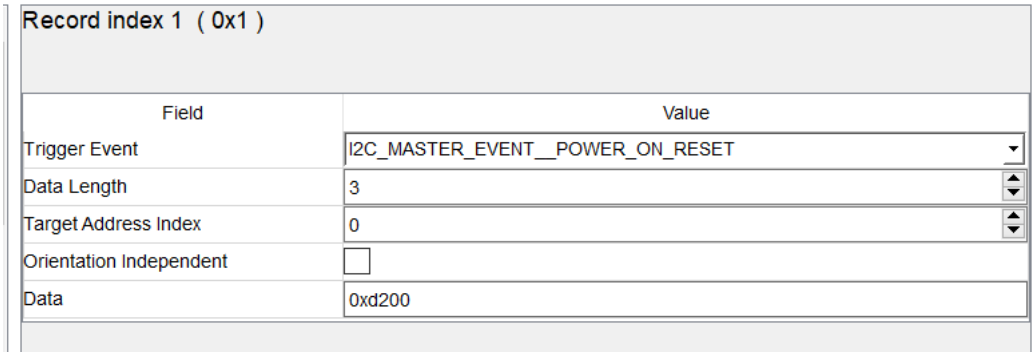

Figure 3-1. Power-On Reset Event On Register 0x00

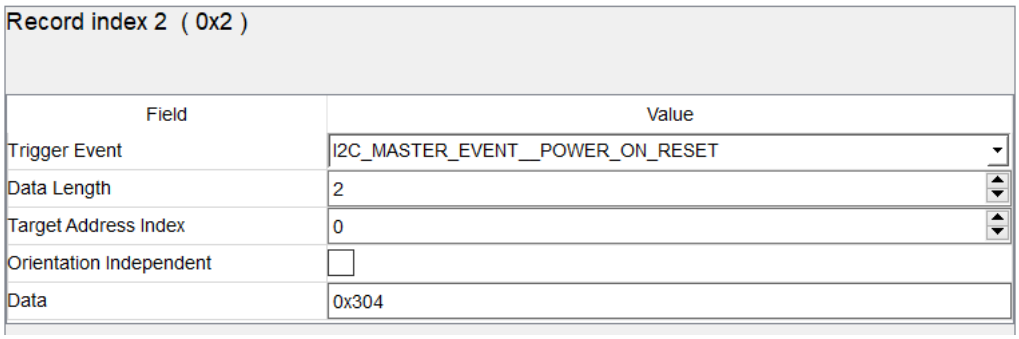

Figure 3-2. Power-On Reset Event On Register 0x04

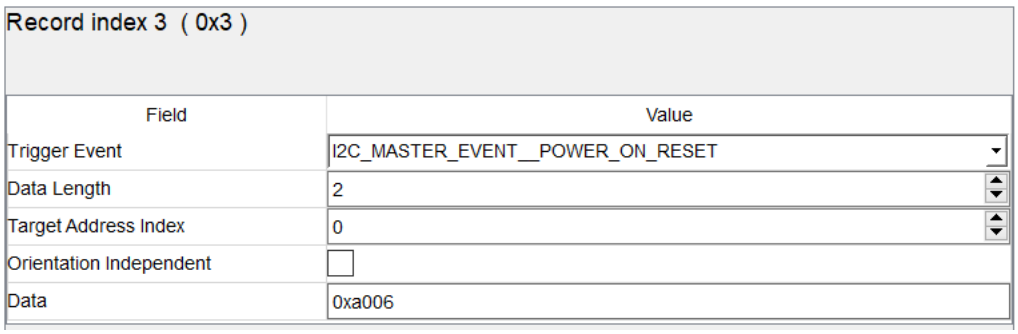

Figure 3-3. Power-On Reset Event On Register 0x06.

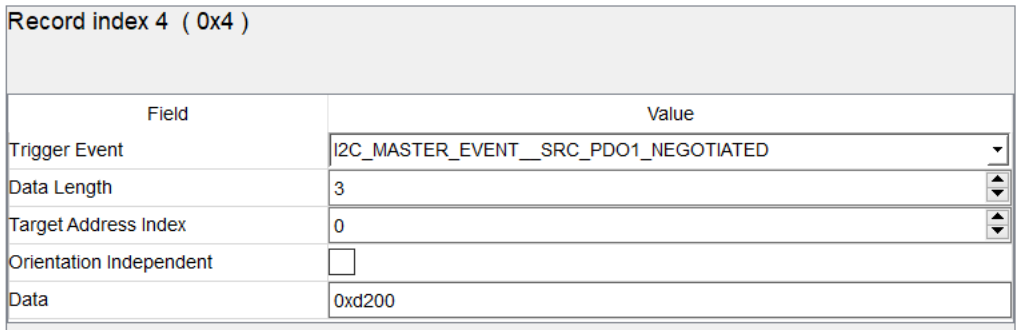

Figure 3-4. SRC PDO1 Negotiated Event On Register 0x00

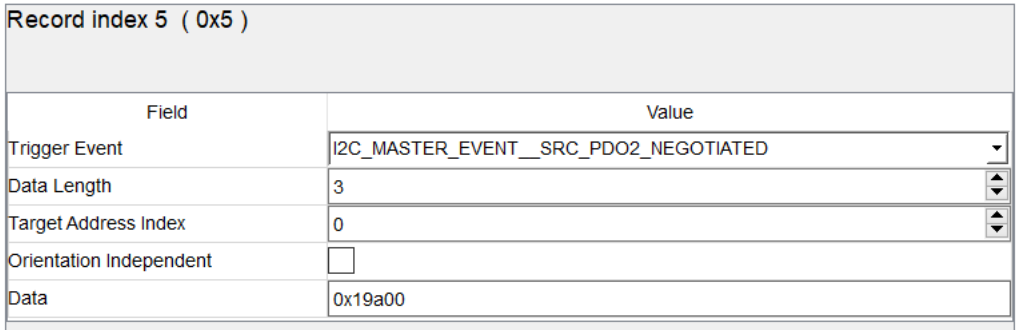

Figure 3-5. SRC PDO2 Negotiated Event On Register 0x00

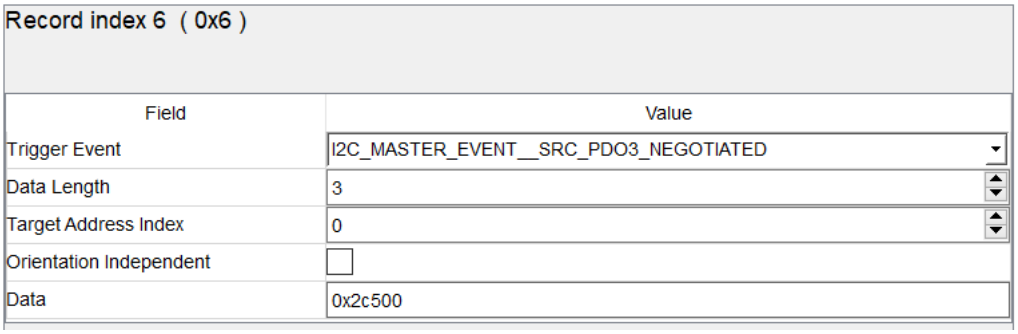

Figure 3-6. SRC PDO3 Negotiated Event On Register 0x00

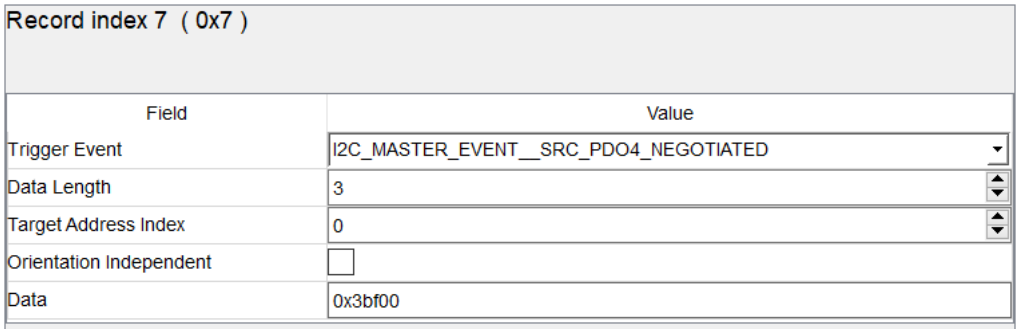

Figure 3-7. SRC PDO4 Negotiated Event On Register 0x00

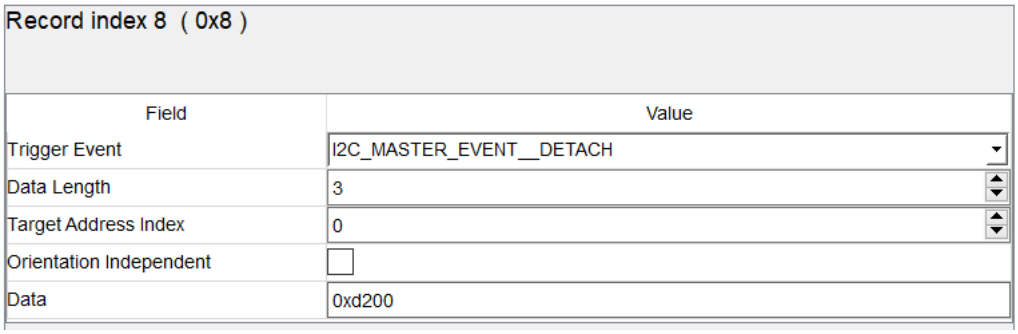

Figure 3-8. Detach Event On Register 0x00

4 Configuring the TUSB1044 Redriver Through I2C3

The TUSB1044 is responsible for routing and conditioning the high-speed signals (USB3 SuperSpeed lanes and optionally DisplayPort lanes) based on the Type-C cable orientation and the Alternate Mode state. Normally, the TUSB1044 can operate in a pin-controlled (GPIO) mode, but in this design we use *I2C control mode* to have fine-grained, dynamic control through the PD controller. Through the TPS65992S I2C controller, the TUSB1044 is configured at various events: port power-on, cable attach (with orientation), and DisplayPort mode entry. All TUSB1044-related I2C commands are assigned to Address Index 1 (peripheral address 0x12) and occupy event indices 9–28 in the PD controller table.

Key registers of the TUSB1044 used in our configuration include:

- *General_1 (Register 0x0A):* This register controls the TUSB1044 operating mode and certain overrides. Important fields in 0x0A:
  - Bits [1:0] *CTLSEL* – Select the data mode: 0 = All channels disabled, 1 = USB3 only enabled, 2 = four lanes of DisplayPort (DP only) enabled, 3 = USB3 + two lanes of DP enabled.
  - Bit [2] *FLIP_SEL* – Sets the orientation (flip) of the high-speed mux: 0 = normal orientation, 1 = flipped orientation.
  - Bit [4] *EQ_OVERRIDE* – When set to 1, enables use of EQ settings from I2C registers (overriding the pin-strapped EQ levels). We enable this to allow programming the EQ through I2C.
  - Bit [5] *SWAP_SEL* – Globally swaps direction of channels (DFP versus UFP) if set. In a source device, we typically leave this at 0 (no swap) unless the design needs to invert the direction (for instance, for sink applications). Our configuration as a DP source keeps SWAP_SEL = 0 (the default).
  - Bit [3] *HPD_IN_OVERRIDE* – Not used in our flow (controls HPD input override, used in sink applications).
- *General_3 (Register 0x0C):* Controls the *VOD (Voltage Output Differential) and DC gain* settings for the redriver, as well as a field for setting the port data role (Dir) in I2C mode. Fields of interest:
  - Bit [6] *VOD_DCGAIN_OVERRIDE* – 1 to override VOD/DC gain pins and use register settings. We set this to 1 to program the amplitude via I2C.
  - Bits [5:2] *VOD_DCGAIN_SEL* – 4-bit composite field that selects the VOD linearity and DC gain setting for all channels. This essentially encodes the two 2-bit pin settings (CFG1 and CFG0) that otherwise set EQ gain. In our design, we choose a setting that provides appropriate signal amplitude for the expected cable lengths. (In the provided configuration, the value written corresponds to a specific pin setting combination – see below).
  - Bits [1:0] *DIR_SEL* – Sets the device data role: 0 = USB and DP Alt Mode *Source*, 1 = USB and DP Alt Mode *Sink*, 2 = USB + Custom Alt Mode Source, 3 = USB + Custom Alt Mode Sink. Our design is a source (for example, a laptop or dock outputting DisplayPort), so we use DIR_SEL = 0.
- *EQ Control Registers (0x10, 0x11, 0x20, 0x21):* These registers configure the equalization settings for each high-speed channel. The TUSB1044 has four high-speed differential pairs (which we can think of as two upstream-facing channels and two downstream-facing channels, each with a TX and RX direction). In I2C mode:
  - *0x10:* UFP1_EQ – Bits 7:4 set the TX EQ for the Upstream Port Channel1 (UTx1), bits 3:0 set the RX EQ for Upstream Channel1 (URx1).
  - *0x11:* UFP2_EQ – Similarly, for Upstream Channel2 (UTx2 and URx2).
  - *0x20:* DFP1_EQ – Bits 7:4 for Downstream Port Channel1 (DTx1, going to the connector), and 3:0 for Downstream RX1 (DRx1).
  - *0x21:* DFP2_EQ – EQ settings for Downstream Channel2 (DTx2/DRx2).

By writing to these, we can fine-tune the signal integrity for each lane. In our use case, we program certain EQ values recommended for USB3 and DP signals (the values 0x66 and 0x33 that appear in our configuration correspond to particular EQ gain settings).

Now, here is how the TPS65992S is configured to control the TUSB1044 at various events:

Power-On Reset (Initial Configuration)

On PD controller *Power-On Reset*, we want the TUSB1044 to start in a safe, disabled state with I2C control enabled. The firmware triggers:

- *Write 0x10 to TUSB1044 register 0x0A*. The data 0x10 (hex) corresponds to binary 0001_0000. Looking at the bit definitions: this sets bit4 (EQ_OVERRIDE) = 1, and all other lower bits 2:0 = 0 (CTLSEL = 000b, which means *all high-speed channels disabled*). In other words, we are *enabling I2C EQ override* but keeping the redriver TX/RX paths off. This is a good default at reset – the redriver does not forward any high-speed signals until we configure the redriver further, and the redriver relies on registers for EQ settings rather than pins. Essentially, 0x10 puts the TUSB1044 into an idle state under software control. (At reset, the TUSB1044 default can also have channels disabled, but writing 0x10 explicitly verifies the correct mode and override bit.)

This command maintains the TUSB1044 is not driving any signals and is ready to be configured for whatever comes next (USB or Alt Mode). This is analogous to holding the redriver in a reset or standby until a cable attach is detected. We also use this same value for other events as needed (like detach) to return to this baseline.

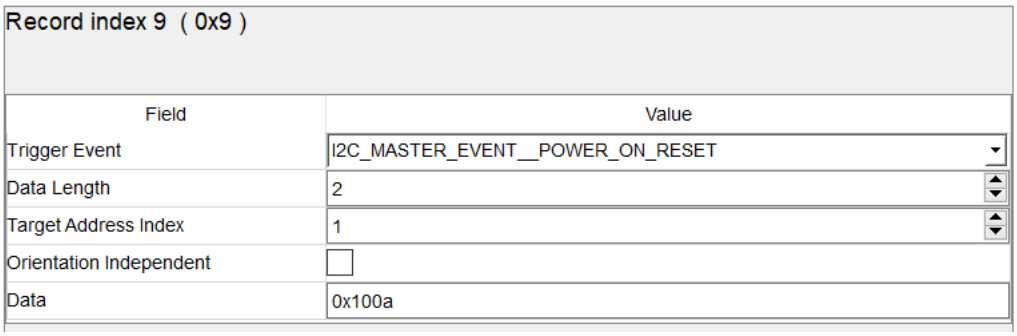

Figure 4-1. Power On Reset Event On Register 0x0A

Detach Event

On a *Detach* (cable unplug) event, the PD controller sends *0x10 to register 0x0A* of the TUSB1044 again. This writes the same value as at power-on: EQ_OVERRIDE = 1, CTLSEL = 0 (channels off). Doing so at detach effectively *shuts down the redriver lane switching and returns to default state*. Any alternate mode configuration that was in effect is cleared, and the device is ready for a fresh attach sequence next time. In essence, both at initial power and on detach, the TUSB1044 is instructed to disable the switching (no USB3 or DP lanes active) and rely on I2C for further config. This prevents unwanted signal passing or latching of a previous configuration when no device is connected.

(Note: Some designs can also toggle a TUSB1044 hardware reset pin on detach. Using I2C commands as shown can achieve a similar result without extra GPIO toggling, by writing the control registers to a known safe state.)

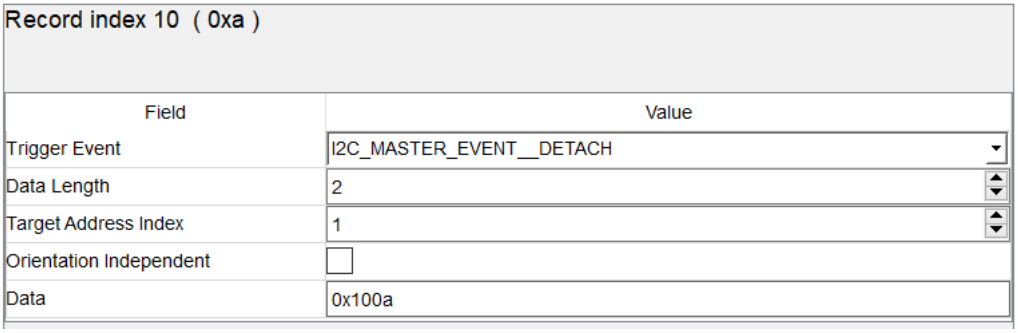

Figure 4-2. Detach Event On Register 0x0A

Attach Event – Cable Orientation Handling

When a USB-C cable is *attached*, the TPS65992S detects the orientation (through the CC pins) and also starts USB-PD negotiation. Before even getting to Alternate Mode, the immediate task is to configure the *SuperSpeed mux* inside TUSB1044 for the correct orientation so that USB3 signals (and future DP signals) route correctly. The TPS65992S firmware distinguishes two orientation cases, often labeled *ATTACH_UU* and *ATTACH_UD* (these labels come from the PD controller event definitions – which essentially refer to the *Upstream Facing Port*

Up versus *Upstream Facing Port Down* orientations, or equivalently *cable not flipped* versus *cable flipped*). In simpler terms:

- *ATTACH_UU* corresponds to orientation where the connector A-side is up (one specific mapping of lanes),
- *ATTACH_UD* corresponds to the connector rotated 180º (the opposite lane mapping).

For each orientation, we define a sequence of I2C writes to the TUSB1044 to set up for normal USB3 operation (and prepare for potential DP mode):

Attach_UU sequence: (Cable in default orientation) The PD controller issues the following writes in quick succession:

1. *0x11 to register 0x0A:* This sets the General_1 register to 0x11 hex (0001_0001 binary). Compared to 0x10, this now has bit0 = 1 (CTLSEL = 001b) while keeping bit4 = 1 and bit2 = 0. CTLSEL = 1 means *USB3.1-only mode enabled*. So 0x11 turns on the USB3 path through the redriver for the given orientation. Bit2 (FLIP_SEL) is 0 here, meaning assume normal orientation (since this is the UU case). Thus, 0x11 configures the TUSB1044 for *USB3 mode, not flipped*.

Interpretation: The TUSB1044 connects the SuperSpeed transmit and receive lanes from the host side to the appropriate TX/RX pins on the connector corresponding to a non-flipped plug insertion. DP lanes (if any) remain disabled at this point (because CTLSEL=1 selects USB3-only). EQ_OVERRIDE remains 1, so we use our programmed EQ settings (set in next steps).

1. *0x58 to register 0x0C:* We write 0x58 to General_3. In binary 0x58 = 0101_1000. Breaking down:
   - Bit6 = 0x40 is set (0x58 contains 0x40), so VOD_DCGAIN_OVERRIDE = 1 (use register for VOD/DC settings).
   - Bits5:2 = 0b0110. According to the data sheet, this field [5:2] encodes the equivalent of the CFG1 and CFG0 pin settings. 0b0110 corresponds to CFG1=01 (*R*) and CFG0=10 (*F*). In other words, we chose a particular VOD/DC gain level (one that can correspond to a medium EQ setting—*R* and *F* likely denote resistor and float combinations). This is likely determined from the TUSB1044 data sheet recommendations or lab tuning.
   - Bits1:0 = 0b00, which sets DIR_SEL = 0, indicating *Source mode* (USB + DP Source). This matches our system role (the TUSB1044 treats the upstream side as the host side and *downstream* side as the connector side).

So, 0x58 essentially says: *use I2C-provided VOD/DC gain, apply a specific gain setting (R-F configuration), and confirm the device is a Source.* We write this at attach to maintain that the redriver's output levels are configured properly for the upcoming signals.

1. *0x66 0x66 to register 0x10:* This is a two-byte write to the UFP1_EQ register (0x10). The data 0x66 0x66 sets both the high and low nibbles of UFP1_EQ to 0x6. Specifically, 0x10 controls UFP TX/RX EQ for one pair:
   - 0x66 as a single byte means TX EQ = 0x6 and RX EQ = 0x6 for UFP Channel1. We send two bytes of 0x66, which likely means we intended to also target the next register (0x11) in a single multi-byte sequence (0x10 followed by data for 0x10 and 0x11). However, the configuration listing shows 0x66 0x66 to 0x10, which can actually load 0x10=0x66 and 0x11=0x66 in one go (depending on how the PD tool frames multi-byte writes).

In essence, we are setting all *Upstream-facing EQ* (both channels) to a mid-level value (0 × 6 is a mid-range setting) for both TX and RX. This can be the recommended EQ setting for the host controller side of the redriver when running USB3 or DP. By doing this, we override the default pin-based EQ and verify signal quality (compensating for board trace loss, and so on).

1. *0x33 0x33 to register 0x20:* Another two-byte write, this time to DFP1_EQ (0x20). 0x33 0x33 similarly sets the Downstream-facing (connector side) Channel1 (and possibly Channel2 if two bytes) EQ settings to 0x3 for both TX and RX. A value of 0x3 is a bit lower gain than 0x6 – this can reflect that the connection from the redriver to the connector is shorter or the default requirement for the cable. Essentially, we program the *Downstream (USB-C connector) EQ* for both channels as well.

The combination of the above EQ settings (0x6 on UFP side, 0x3 on DFP side) is likely determined by TI's guidelines or lab tuning to pass the USB3 signal integrity and compliance tests for the given PCB and connector.

These four writes (0x0A, 0x0C, 0x10, 0x20) complete the Attach_UU configuration. At this point, for a non-flipped cable insertion, the TUSB1044 is set to active USB3 mode: the correct high-speed lanes are connected, and the EQ/gain is tuned.

Attach_UD sequence: (Cable flipped orientation) When the cable is inverted, the PD controller instead triggers the ATTACH_UD events. The sequence is very similar, but with differences where orientation matters:

1. *0x15 to register 0x0A:* 0x15 hex = 0001_0101 binary. Comparing to 0x11 (0001_0001):
   - Bit2 (FLIP_SEL) is now 1 (since 0x15 has bit2 set, as 0x15 = 21 decimal, which includes 0x04).
   - Bits1:0 are still 01 (CTLSEL = 1, USB3-only mode).
   - Bit4 remains 1 (EQ_OVERRIDE on).

   So 0x15 configures *USB3-only enabled, but with the orientation flipped*. This tells the TUSB1044 to route the lanes accordingly (the A-port and B-port lanes are swapped relative to the device internal mux). Essentially, the device connects the USB3 signals to the opposite set of high-speed pins compared to the UU case, to account for the flipped cable.
2. *0x58 to 0x0C:* The General_3 register gets the same 0x58 value as before. This is actually independent of orientation – the VOD/DC gain setting and DIR (source) remain the same regardless of flip. So we use the identical 0x58 configuration for both UU and UD. (This makes sense: flip orientation does not change the fact we are a source with certain EQ preferences; this only changes which physical channels carry the signals.)
3. *0x66 0x66 to 0x10:* We again write the same EQ settings to the UFP EQ registers. Here we must consider: in flipped orientation, which TUSB1044 channels correspond to which physical connection? The device swaps the usage of the *Channel1* versus *Channel2* for the lanes, as indicated by FLIP_SEL. However, by writing the same values to both UFP channels (0x10 and 0x11 both = 0x66 via the two-byte sequence), we verify that regardless of which channel becomes the actual TX/RX for USB, the EQ is set to 0x6. In other words, both upstream channels have identical EQ in our configuration, so a flip does not require different values due to symmetry.
4. *0x33 0x33 to 0x20:* Similarly, we set both downstream channels' EQ to 0x3 (0x20 and 0x21 both get 0x33 via the multi-byte write). This symmetry means the UD case uses the same EQ strength on both possible connector orientations.

The result is that the UD sequence only differs in the *0x0A register value (0x15 versus 0x11)*. All other register writes (0x0C, 0x10, 0x20 and data) are identical between UU and UD attaches. This makes sense: the only thing that changes is the flip bit, which tells the redriver which way to route the lanes. By comparing the two, we see that 0x0A = 0x11 versus 0x15 is a difference of 0x04, which indeed corresponds to the FLIP_SEL bit. Everything else (EQ gains, and so on) remain constant, showing that our design does not require different gain settings for flipped versus unflipped – treating them the same electrically.

After an Attach event sequence (UU or UD), the TUSB1044 is configured for *normal USB3 operation*. If no Alternate Mode is initiated, the system continues to operate with USB3 traffic flowing through the redriver. The PD controller job at attach is essentially done: this maintains the high-speed paths are correctly oriented and optimized.

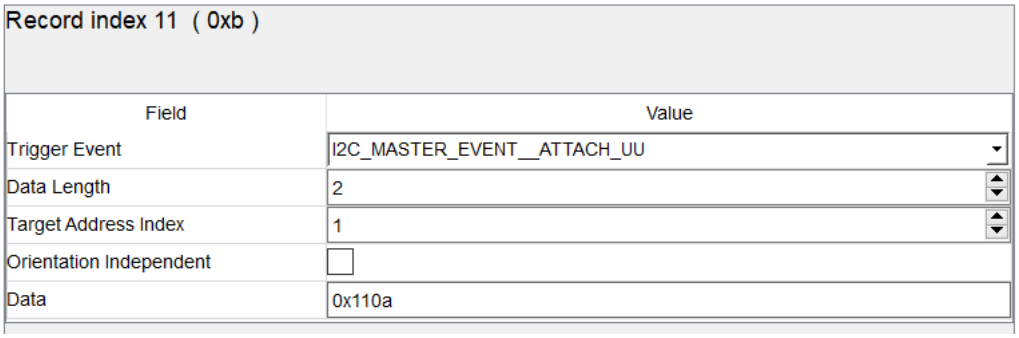

Figure 4-3. Attach Event On Register 0x0A

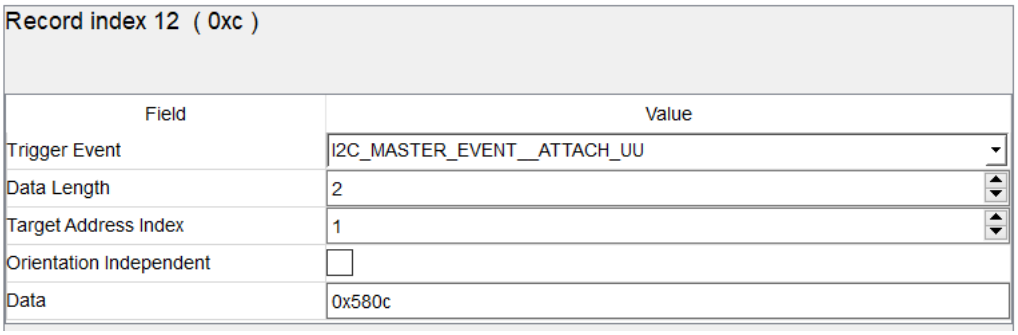

Figure 4-4. Attach Event On Register 0x0C

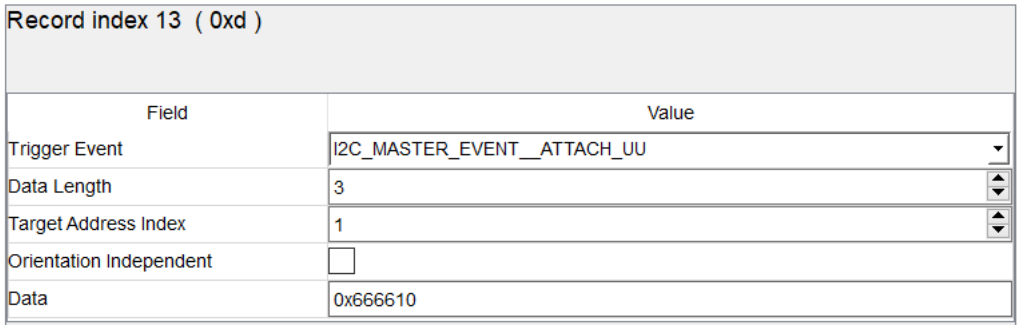

Figure 4-5. Attach Event On Register 0x10

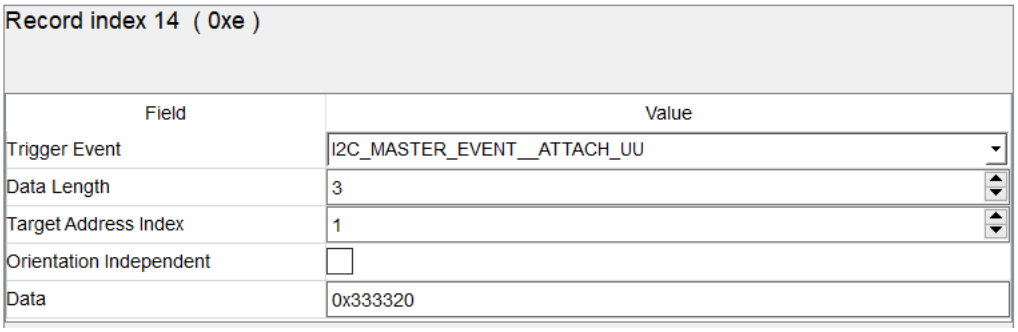

Figure 4-6. Attach Event On Register 0x20

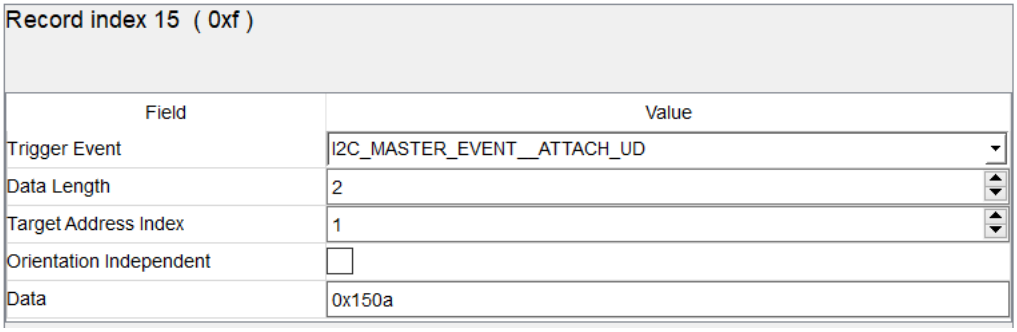

Figure 4-7. Attach Event On Register 0x0A

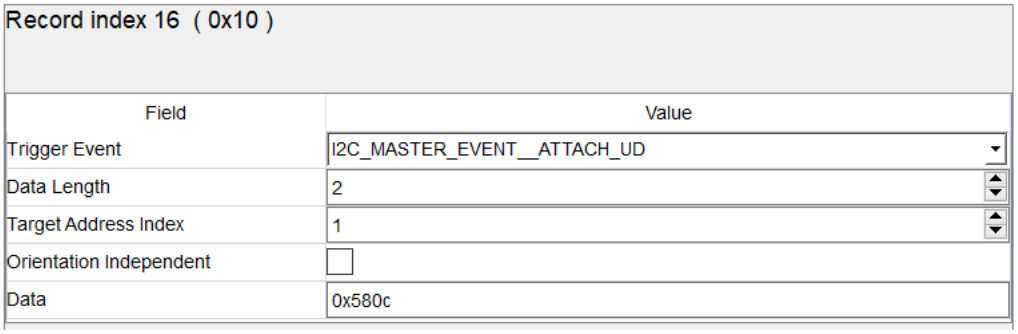

Figure 4-8. Attach Event On Register 0x0C

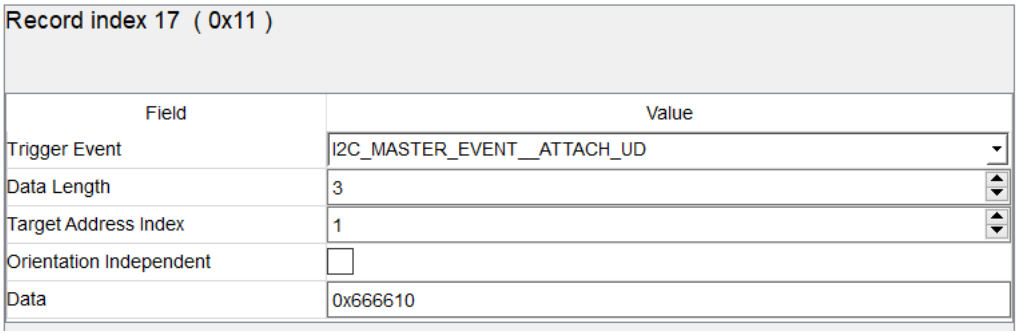

Figure 4-9. Attach Event On Register 0x10

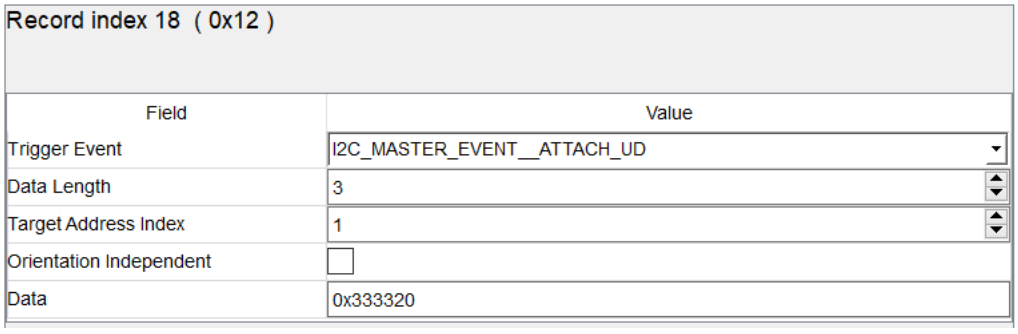

Figure 4-10. Attach Event On Register 0x20

DisplayPort Alternate Mode Configuration

If the connected partner (sink device, for example, a monitor or docking station) supports DisplayPort Alternate Mode and the PD negotiation enters that mode, additional I2C events are triggered to reconfigure the TUSB1044 for DisplayPort operation. In our PD controller configuration, we have defined events corresponding to specific

DisplayPort configuration steps, labeled for convenience as *DP_CONFIG_ACE* and *DP_CONFIG_BDF*, each with variants for orientation (UU or UD). These labels refer to the standard USB-C DisplayPort *pin assignments* – typically, pin assignments *C and E* versus *B, D and F*. In essence, the PD controller uses different configuration sequences depending on how the four high-speed lanes are going to be used for DisplayPort. One configuration can be for 4-lane DisplayPort (no USB3 data), and another for 2-lane DisplayPort + USB3 operation, which are common modes for DP Alt Mode. The exact naming (ACE, BDF) comes from the fact that a source can advertise support for certain pin maps (for example, pin assignment C or E for 4-lane, and pin assignment D or F for 2-lane+USB). Our configuration maintains that the TUSB1044 is properly set for the lane routing and EQ in each scenario.

We summarize the sequences:

- *DP_CONFIG_ACE_UU:* This event corresponds to entering a DisplayPort Alt Mode configuration that uses (A,C,E) mapping in a non-flipped orientation. In this mode, typically all four lanes are devoted to DP (USB3 is dropped). The PD controller writes:
  - *0x1A to 0x0A:* 0x1A hex = 0001_1010 binary. Bits [1:0] = 10 (CTLSEL = 2), which per TUSB1044 means *4 lanes of DisplayPort enabled*. Bit2 = 0 (UU orientation), Bit4 = 1 (EQ override still on). So 0x1A switches the redriver to DP mode (all lanes carry DP) in normal orientation.
  - *0x55 0x55 to 0x10:* A two-byte write to UFP1_EQ (and UFP2_EQ). The data 55 55 sets the upstream EQ registers to 0x55 each. Breaking 0x55: that is 0x5 for TX and 0x5 for RX on each channel. So in DP mode, we are using a slightly different EQ setting for the upstream side: 0x5 (which is a bit lower than the 0x6 we used for USB). This can be to optimize DP signal quality (since DP signals can have different equalization needs or the DP source can do some EQ).

We do not explicitly see a write to 0x20 in this event (suggesting possibly the downstream EQ remains as set, or is not needed to change for 4-lane DP UU). The assumption is that the downstream (connector-facing) EQ can remain at whatever was last set (0x33 from attach). This is plausible because whether carrying USB or DP, the physical channel characteristics to the connector do not change; however, one can also adjust them for DP frequency. In our configuration table, no new 0x20 write is listed for ACE_UU, so we keep the same 0x33 0x33 on DFP EQ as previously.

- *DP_CONFIG_ACE_UD:* This is the flipped orientation counterpart to the above:
  - *0x1E to 0x0A:* 0x1E hex = 0001_1110. Bits [1:0] = 10 (CTLSEL = 2, still 4-lane DP), Bit2 = 1 (flip orientation), Bit4 = 1 (EQ override on). So 0x1E means *DP 4-lane mode, flipped orientation*.
  - *0x55 0x55 to 0x10:* The same 55 55 data to the UFP EQ registers. We see again that the only difference between ACE_UU and ACE_UD is the 0x0A value (0x1A versus 0x1E). That 0x04 difference indicates the flip bit, exactly as with attach events. Register 0x10's data remains 0x55 0x55 (no change due to flip, since again we set both channels to identical values).

As before, no new DFP EQ write is shown for ACE_UD either, implying the DFP side EQ is unchanged by this DP mode entry in our sequence.

- *DP_CONFIG_BDF_UU:* This corresponds to a different pin assignment scenario – likely 2 lanes of DP (B and D) + USB3 (F). In this case, the PD controller writes:
  - *0x1B to 0x0A:* 0x1B hex = 0001_1011. Bits [1:0] = 11 (CTLSEL = 3), which means *"USB3 + 2 lanes of DP"* mode. Bit2 = 0 (no flip, UU), Bit4 = 1 (EQ override on). So 0x1B configures the TUSB1044 to route two of the four lanes for DP and keep one lane pair for USB3, in normal orientation. This corresponds to, for example, using two lanes for DP (HBR lanes) and leaving the other two lanes for the USB3 TX/RX.
  - *0x66 0x55 to 0x10:* Here we have a two-byte sequence to UFP EQ registers with data 66 55. This is different from previous cases. Likely meaning: UFP1_EQ = 0x66, UFP2_EQ = 0x55 (since the first byte 0x66 goes to reg 0x10, second byte 0x55 to reg 0x11). In effect, we are setting one upstream channel EQ to 0x6 and the other to 0x5. Why? Because in the BDF scenario, one of the upstream channels is carrying USB3 (which we previously tuned to 0x6), and the other is carrying DP (for which we decided 0x5 was sufficient). So Channel1 (say, the one still for USB3) can use EQ = 6, while Channel2 (carrying DP lanes) uses EQ = 5. The ordering of 0x66 0x55 suggests: UFP1_EQ = 0x66 (for the USB3 channel), UFP2_EQ = 0x55 (for the DP channel).
  - *0x33 0x33 to 0x20:* A two-byte write to DFP1_EQ (0x20) and DFP2_EQ (0x21) with 33 33. This sets both downstream channel EQ to 0x3 (same as before) regardless of the carried signals. Interestingly, the table

suggests that in BDF_UU, 0x20 did not update (even though this is the same values as attach). Perhaps this is done just to maintain that the DP lane outputs (which now possibly occupy different physical lines) are set to known EQ. In any case, DFP1_EQ and DFP2_EQ are both 0x33, which is symmetric and unchanged from prior values.

- *DP_CONFIG_BDF_UD:* The flipped case for the 2-lane DP + USB scenario:
  - *0x1F to 0x0A:* 0x1F hex = 0001_1111. Bits [1:0] = 11 (USB3 + 2DP), Bit2 = 1 (flipped), Bit4 = 1 (override on). So 0x1F is essentially 0x1B with the flip bit added (0x04), meaning “USB3 + 2DP in flipped orientation.”
  - *0x55 0x66 to 0x10:* Data 55 66 is written starting at 0x10. This presumably sets UFP1_EQ = 0x55 and UFP2_EQ = 0x66. Why the swap of bytes compared to the UU case? Because when flipped, the role of the two upstream channels in terms of which carries USB versus DP can swap. In UU, we assumed Channel1 was USB, Channel2 was DP (leading to 66 for channel1, 55 for channel2). In UD orientation, the physical routing can cause the opposite assignment – Channel1 can end up carrying DP and Channel2 carrying USB3 (depending on how the lanes flip). The PD configuration anticipates this by swapping the EQ settings: now channel1 gets 0x5 and channel2 gets 0x6. This maintains that whichever channel is handling USB3 has the higher EQ (0x6) and the one handling DP has 0x5, even when flipped. The explicit difference between BDF_UU and BDF_UD configurations (66 55 versus 55 66) highlights how orientation flip can require swapping lane-specific settings.
  - *0x33 0x33 to 0x20:* As with UU, we write 0x33 0x33 to DFP EQ registers. In BDF_UD, this is likely the same action: keep both downstream channels at EQ = 3. Since flip doesn’t really change the fact both connector lanes (two of them carrying DP, the others carrying USB and possibly one unused) have the same EQ, we leave them as is.

To summarize the DP configurations: The PD controller uses two sets of events corresponding to the two main DP Alt Mode configurations (4-lane versus 2-lane) and handles both orientations for each. In each case, this writes a new value to register 0x0A to switch the TUSB1044 mode (DP or DP+USB, and set the flip state), and adjusts the EQ registers 0x10/0x11 (and sometimes 0x20/0x21) to maintain signal integrity for the new mode. The values chosen (0x1A/1E, 0x1B/1F for reg0x0A, and the various EQ codes) are derived from the TUSB1044’s requirements:

- *0x0A values:* Differ mostly in the low 3 bits. CTLSEL = 2 for DP-only, = 3 for DP+USB; FLIP=0 or 1 for orientation. Bit4 is always kept 1 (we always use EQ override mode).
- *EQ values:* We used 0x6 for USB3 channels and 0x5 for DP channels on the upstream side, and 0x3 universally on the downstream side. These specific codes come from either TI’s reference design or lab tuning to meet USB3 Gen2 and DP HBR2/HBR3 compliance. The important part is not the exact number, but that the PD controller can *alter them as needed depending on scenario*. For instance, if a particular design needed different EQ for a longer cable detected, the firmware potentially uses different values (though our example keeps them fixed per mode).

Once the DisplayPort configuration events have executed, the TUSB1044 is fully in Alt Mode configuration: lanes are routed to carry DisplayPort from the system’s GPU through the redriver to the USB-C connector, and (if in the BDF case) two of the lanes continue carrying the USB3 data. The PD controller active involvement ends here, and the system transmits DisplayPort AUX handshake etc. to set up monitors as usual. The TUSB1044, being protocol-agnostic, simply passes through the high-speed signals; we have effectively just controlled the switches and gains through I2C.

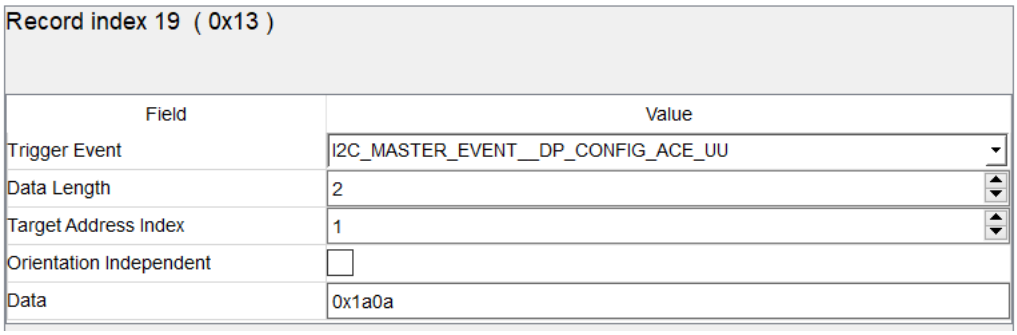

Figure 4-11. DP assignment ACE config event on register 0x0A

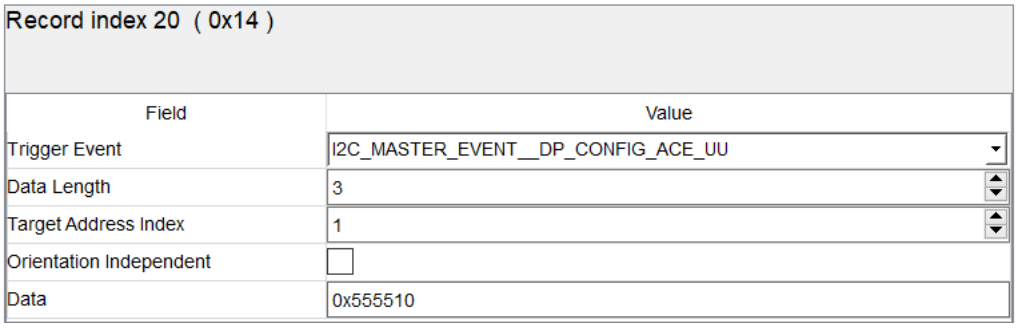

Figure 4-12. DP assignment ACE config event on register 0x10

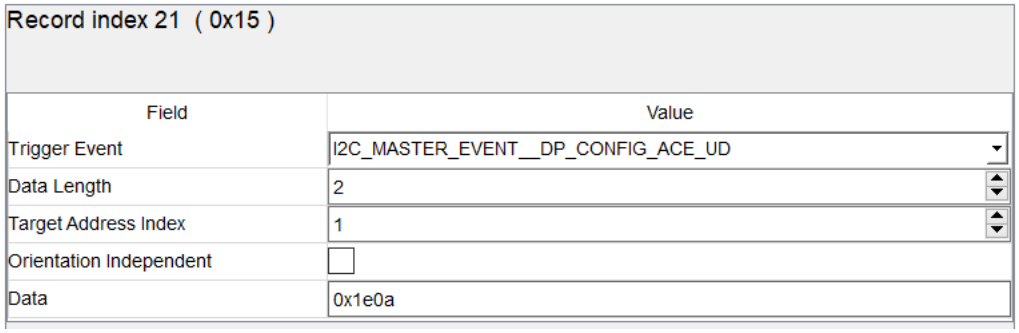

Figure 4-13. DP assignment ACE config event on register 0x0A

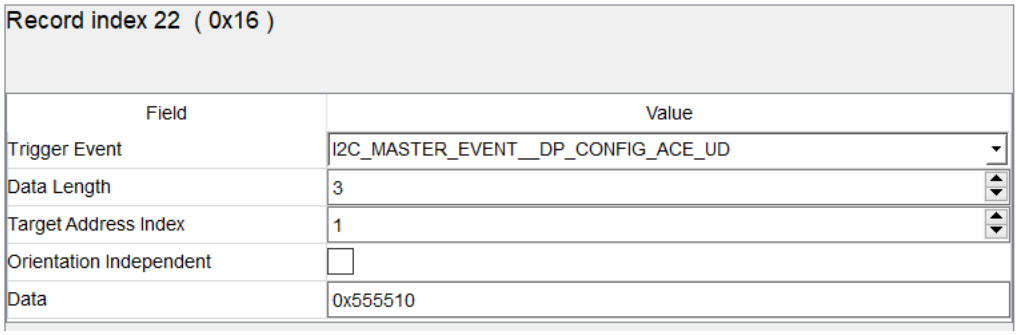

Figure 4-14. DP assignment ACE config event on register 0x10

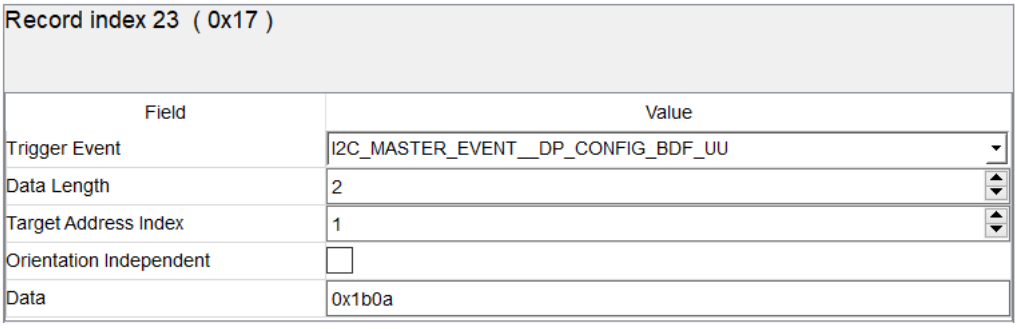

Figure 4-15. DP assignment BDF config event on register 0x0A

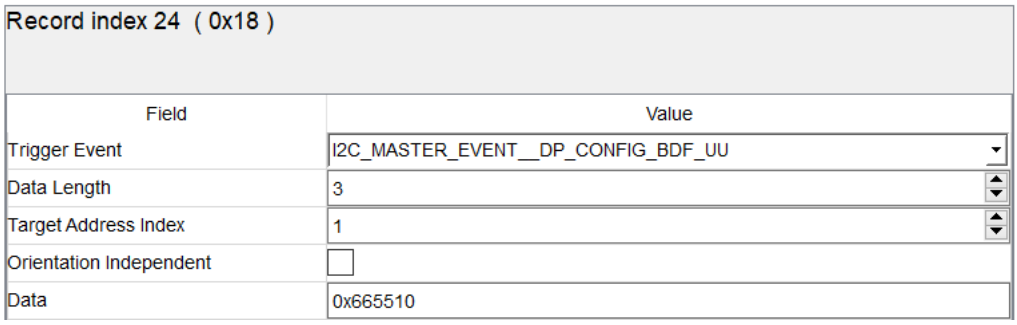

Figure 4-16. DP assignment BDF config event on register 0x10

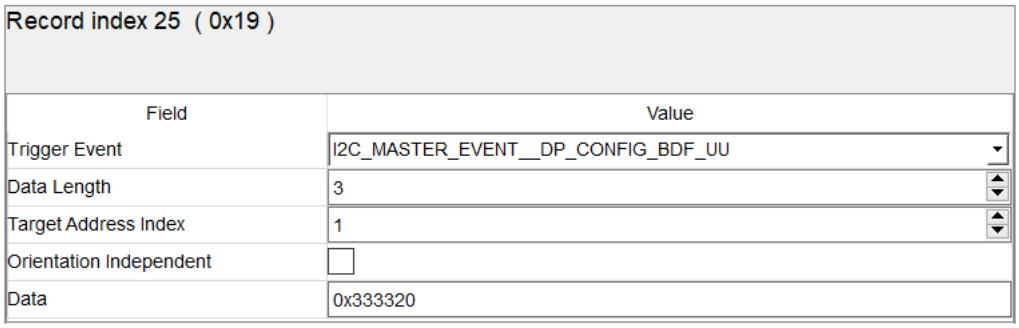

Figure 4-17. DP assignment BDF config event on register 0x20

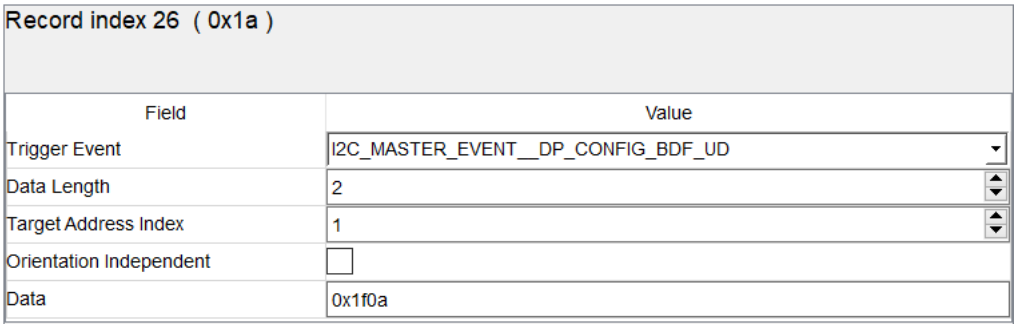

Figure 4-18. DP assignment BDF config event on register 0x0A

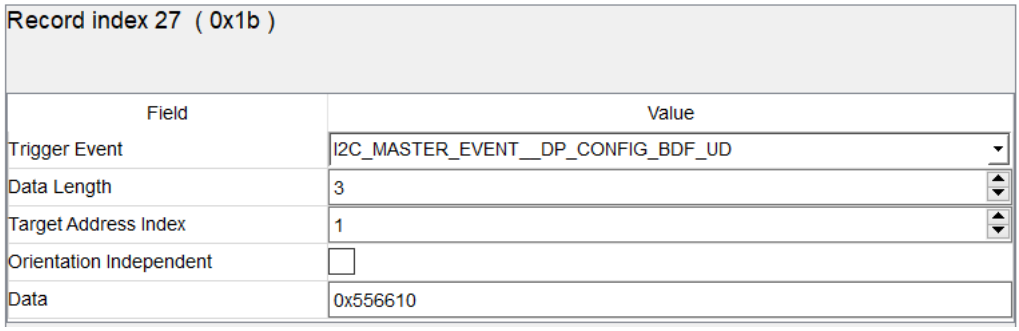

Figure 4-19. DP assignment BDF config event on register 0x10

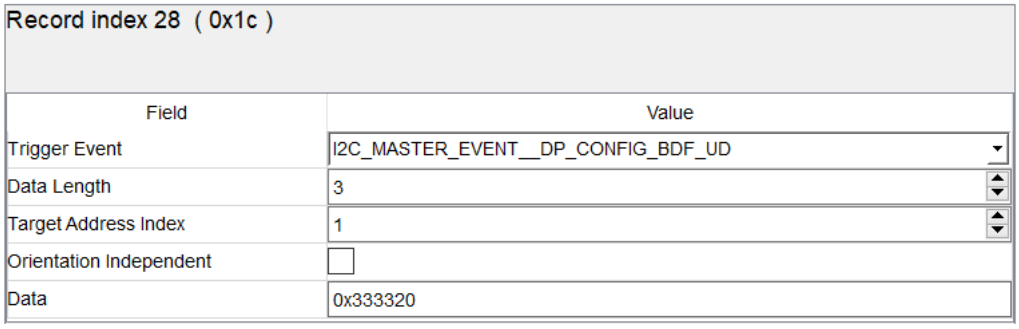

Figure 4-20. DP assignment BDF config event on register 0x20

Through these attach and DP config event handlers, the TPS65992S maintains that the TUSB1044 is properly configured at every stage: initial attach (for USB 3.2 SuperSpeed or baseline operation) and during the transition into DisplayPort Alternate Mode. All necessary I2C writes are automated by the PD controller firmware once configured, eliminating the need for a dedicated EC or MCU to handle the redriver settings in real time.

5 Summary of I2C Event Table

By using the TPS65992 I2C controller feature, we achieved coordinated control of both the power supply (TPS55288) and the signal redriver (TUSB1044) in response to Type-C events. Table 5-1 below summarizes all the configured I2C events, including the index, trigger, target device register, and data written:

Table 5-1. All I2C Index Summary

	Trigger Event	register	value
Index1	I2C_MASTER_EVENT_POWER_ON_RESET	0x00	0xD2
Index2	I2C_MASTER_EVENT_POWER_ON_RESET	0x04	0x03
Index3	I2C_MASTER_EVENT_POWER_ON_RESET	0x06	0xa0
Index4	I2C_MASTER_EVENT_SRC_PDO1_NEGOTIATED	0x00	0xD2 0x00
Index5	I2C_MASTER_EVENT_SRC_PDO2_NEGOTIATED	0x00	0x9A 0x01
Index6	I2C_MASTER_EVENT_SRC_PDO3_NEGOTIATED	0x00	0xC5 0x02
Index7	I2C_MASTER_EVENT_SRC_PDO4_NEGOTIATED	0x00	0xBF 0x03
Index8	I2C_MASTER_EVENT_DETACH	0x00	0xD2
Index9	I2C_MASTER_EVENT_POWER_ON_RESET	0x0A	0x10
Index10	I2C_MASTER_EVENT_DETACH	0x0A	0x10
Index11	I2C_MASTER_EVENT_ATTACH_UU	0x0A	0x11
Index12	I2C_MASTER_EVENT_ATTACH_UU	0x0C	0x58
Index13	I2C_MASTER_EVENT_ATTACH_UU	0x10	0x66 0x66
Index14	I2C_MASTER_EVENT_ATTACH_UU	0x20	0x33 0x33
Index15	I2C_MASTER_EVENT_ATTACH_UD	0x0A	0x15
Index16	I2C_MASTER_EVENT_ATTACH_UD	0x0C	0x58
Index17	I2C_MASTER_EVENT_ATTACH_UD	0x10	0x66 0x66
Index18	I2C_MASTER_EVENT_ATTACH_UD	0x20	0x33 0x33
Index19	I2C_MASTER_EVENT_DP_CONFIG_ACE_UU	0x0A	0x1A
Index20	I2C_MASTER_EVENT_DP_CONFIG_ACE_UU	0x10	0x55 0x55
Index21	I2C_MASTER_EVENT_DP_CONFIG_ACE_UD	0x0A	0x1E
Index22	I2C_MASTER_EVENT_DP_CONFIG_ACE_UD	0x10	0x55 0x55
Index23	I2C_MASTER_EVENT_DP_CONFIG_BDF_UU	0x0A	0x1B
Index24	I2C_MASTER_EVENT_DP_CONFIG_BDF_UU	0x10	0x66 0x55
Index25	I2C_MASTER_EVENT_DP_CONFIG_BDF_UU	0x20	0x33 0x33
Index26	I2C_MASTER_EVENT_DP_CONFIG_BDF_UD	0x0A	0x1F
Index27	I2C_MASTER_EVENT_DP_CONFIG_BDF_UD	0x10	0x55 0x66
Index28	I2C_MASTER_EVENT_DP_CONFIG_BDF_UD	0x20	0x33 0x33

(Table entries reflect the configuration described above. “REG0x00” with two data bytes implies a multi-byte write starting at 0x00. Unused index 8 is left for future expansion or other events if needed. Orientation-independent writes (like to 0x0C, 0x10, 0x20) were duplicated under both UU and UD triggers in our table for completeness, although the data is identical.)

As shown, the TPS65992S handles a wide range of events – from initial power-up of the port, through cable attachment in either orientation, through USB PD contract negotiation, and into DisplayPort Alternate Mode – all by executing predefined I2C transactions. This *event-driven control scheme* maintains that at each stage, the external hardware (power converter and redriver) is in the proper state:

- At reset and detach, the power is safe (5V or off) and high-speed switches are reset.
- On attach, the power stays at 5V until higher voltage is requested, and the USB3 redriver connects the Superspeed lanes correctly.
- When a higher voltage PDO is requested, the PD controller raises the TPS55288's output to the required level.
- If DisplayPort mode is entered, the PD controller reconfigures the redriver so that the correct lanes carry DisplayPort and any remaining lanes carry USB3, adjusting EQ settings as needed for signal integrity.
- All of this occurs autonomously, without real-time software intervention, once the PD controller is programmed through the customization tool.

Conclusion

Using the TPS65992S integrated I2C host capability, a USB-C PD system can tightly coordinate power delivery and high-speed signal switching. In this application, the TPS65992S configures the TPS55288 buck-boost converter to output the appropriate voltage for each PD contract and manages the TUSB1044 redriver to route USB3/DisplayPort signals correctly for any cable orientation and mode. The *elimination of logical inconsistencies* (such as verifying the converter is enabled only after setting the correct voltage, and the redriver is never in an undefined state during mode switches) is achieved through careful event ordering in the PD controller firmware.

This approach replaces what can otherwise require an external MCU or complex GPIO logic with a *firmware-driven design* that is easier to maintain and update using TI's tool. Each critical event (attach, detach, contract negotiation, mode entry) triggers a predefined sequence, maintaining the hardware state always aligns with the USB-C state.

The result is a robust design where, for example, when a laptop is plugged into a monitor: the Type-C controller immediately configures the TUSB1044 for the correct orientation, then negotiates a higher voltage and signals the TPS55288 to ramp up to 20V for charging, and when DisplayPort Alt Mode is initiated, reconfigures the redriver so the display comes to life – all in a fraction of a second and without user intervention. This demonstrates the power of an integrated PD controller design.

6 References

- TPS55288 36-V, 16-A Buck-boost Converter with I2C Interface
- TUSB1044 USB TYPE-C™ 10Gbps Multi-Protocol Bidirectional Linear Redriver

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