Designing with the Host Processor Interface of EZ-PD™ USB Type-C Controllers

About this document

Scope and purpose

AN211898 introduces the host processor interface (HPI) of the EZ-PD™ CCG3 and CCG4 devices. It describes the HPI architecture and register set, gives application examples, and explains how the embedded controller (EC) can update the CCG3/4 device's firmware over the host processor interface (HPI).

Table of contents

Table of contents

Table of contents

1 Introduction

USB Power Delivery (PD) Specification Revision 2.0 defines power delivery up to 100 W (20 V at 5 A) over existing USB standards. The **USB Type-C Cable and Connector Specification** details a new reversible and sub-3-mm slim connector design that supports 100 W of power along with USB and non-USB signals such as DisplayPort. CCG3 and CCG4 devices (referred to collectively as CCG3/4 in the remainder of this document) comply with the latest USB Type-C and PD standards and provide a complete USB Type-C and Power Delivery solution for notebooks, monitors, docking stations, power adapters, and other USB PD applications.

CCG3 is a single Type-C PD port device, whereas CCG4 supports dual Type-C PD ports. Refer to **AN210403 – Hardware Design Guidelines for Dual Role Port Applications Using EZ-PD USB Type-C Controllers** to learn about these Type-C PD controllers and the differences among them. See **AN96527 – Designing USB Type-C Products Using Cypress's CCG1 Controllers** for details on Type-C and power delivery channels.

CCG3/4 devices communicate with the attached Type-C PD device to manage the USB Type-C events and control the power delivery as defined in the USB Type-C and USB PD Specifications. For example, in a notebook system design, battery charging or discharging is managed by the battery charger controller (BCC) and embedded controller (EC) (also called the host processor). The EC communicates with the CCG3/4 device over the HPI to negotiate for power with the attached Type-C device based on the charge present in the battery of the notebook. The HPI is provided over a standard I^2C interface such that an EC can monitor and control the run-time operation of the CCG3/4 device in any application.

This application note describes the HPI architecture and explains the HPI register set to kick-start the design. It explains how the HPI in the CCG3 and CCG4 devices provides capabilities to the EC in a system to read or change power profiles, monitor status, and update the CCG3 or CCG4 firmware. Various application examples of the HPI illustrate how CCG3/4 can communicate with the EC to read or change power profiles with an attached port partner. Also covered is communication of the CCG3/4 device with the EC to control the alternate mode, such as DisplayPort, as well as reconfiguration of display mux controllers.

1.1 Target Applications with HPI

The HPI of a CCG3/4 device can be used in applications where the embedded controller needs to communicate with the CCG3/4 device for power negotiations or firmware update. Typical applications of HPI are listed as below:

- Notebooks
- Docking stations
- Monitors
- Dongles

1.2 CCG3/4 Related Resources

The CCG4 design resources include datasheets, application notes, evaluation kits, reference designs, and firmware development and debugging tools. **Table 1** summarizes the resources.

Table 1 CCG3/4 Related Resources

Category	Available Resources	Where to Find Resources
Datasheet	CCG3 datasheet	**CCG3 Datasheet**
	CCG4 datasheet	**CCG4 Datasheet**
Hardware	Evaluation kit – schematic, board files, and documentation	**CY4531 CCG3 EVK** **CY4541 CCG4 EVK**

Category	Available Resources	Where to Find Resources
Application Notes	Hardware design guidelines including recommendations for resistors, decoupling capacitors for power supplies, and PCB layout	**AN210403**
	Getting started with CCG4	**AN210771**
	Designing USB Type-C products using the CCG1 controllers	**AN96527**
Programming Specifications Guide	The programming reference manual gives information necessary to program the nonvolatile memory of the CYPD4xxx devices	**Programming Specifications Guide**
Host PC Software	Software Development Kit	**EZ-PD CCGx SDK**
	GUI-based Windows application to help users configure the CCG4 controller	**EZ-PD Configuration Utility**
	Firmware development tool	**PSoC Creator 3.3 SP1 or later**
	Firmware programming tool	**PSoC Programmer 3.24 or later**
Debugging Tools	CY4500 EZ-PD Analyzer -- schematic, board files, documentation, and the EZ-PD Analyzer Tool	**CY4500 EZ-PD Analyzer**
Other Collateral	Knowledge base articles	**Knowledge Base Articles for CCG4 Controller**

2 HPI Specification

This specification describes the HPI protocol used in CCG3/4. It includes a description of the HPI physical and transport layer, initialization sequence, and HPI registers.

2.1 HPI Protocol

This section documents the high-level protocol of the CCG3/4 device's HPI.

2.1.1 Command-Response Model

Figure 1 shows the HPI command-response model between the EC and CCG3/4. The EC sends commands to CCG3/4 by writing to the command registers. The CCG3/4 device sends command responses to the EC through Response registers maintained in the HPI space. The responses for device commands and USB PD commands for each Type-C port are reported through separate registers. The RESPONSE_REGISTER is used only to provide responses associated with device commands. The PD_RESPONSE registers corresponding to each USB PD port are used to provide responses associated with PD commands.

CCG3/4 uses the INTR# pin to notify the EC that a response is available in a register. The EC needs to read the interrupt register (INTR_REG) to identify the RESPONSE register that has the response. EC reads only that particular RESPONSE register. Subsequently, the EC reads the response and then clears the interrupt by writing to the Interrupt Status register (INTR_REG). INTR_REG contains separate status bits for each Response register. The EC is expected to read the RESPONSE register and clear the interrupt status before sending a new command (register write). The CCG3/4 device clears the contents of the RESPONSE register only after the EC writes to INTR_REG. If the EC sends a new command before reading an outstanding response to a previous command, the response to the new command is queued in the internal message queue. The INTR# pin will be asserted if at least one of the RESPONSE registers contains a message. The CCG3/4 device de-asserts the INTR# pin when the interrupt status has been cleared.

Note: *The rest of this document uses the term "RESPONSE register" to refer to both the RESPONSE_REGISTER and the PD_RESPONSE register for simplicity, unless a specific reference is required. The RESPONSE_REGISTER is used only for operations related to firmware update and switch firmware mode. Responses and events related to PD operation are sent through the PD_RESPONSE register.*

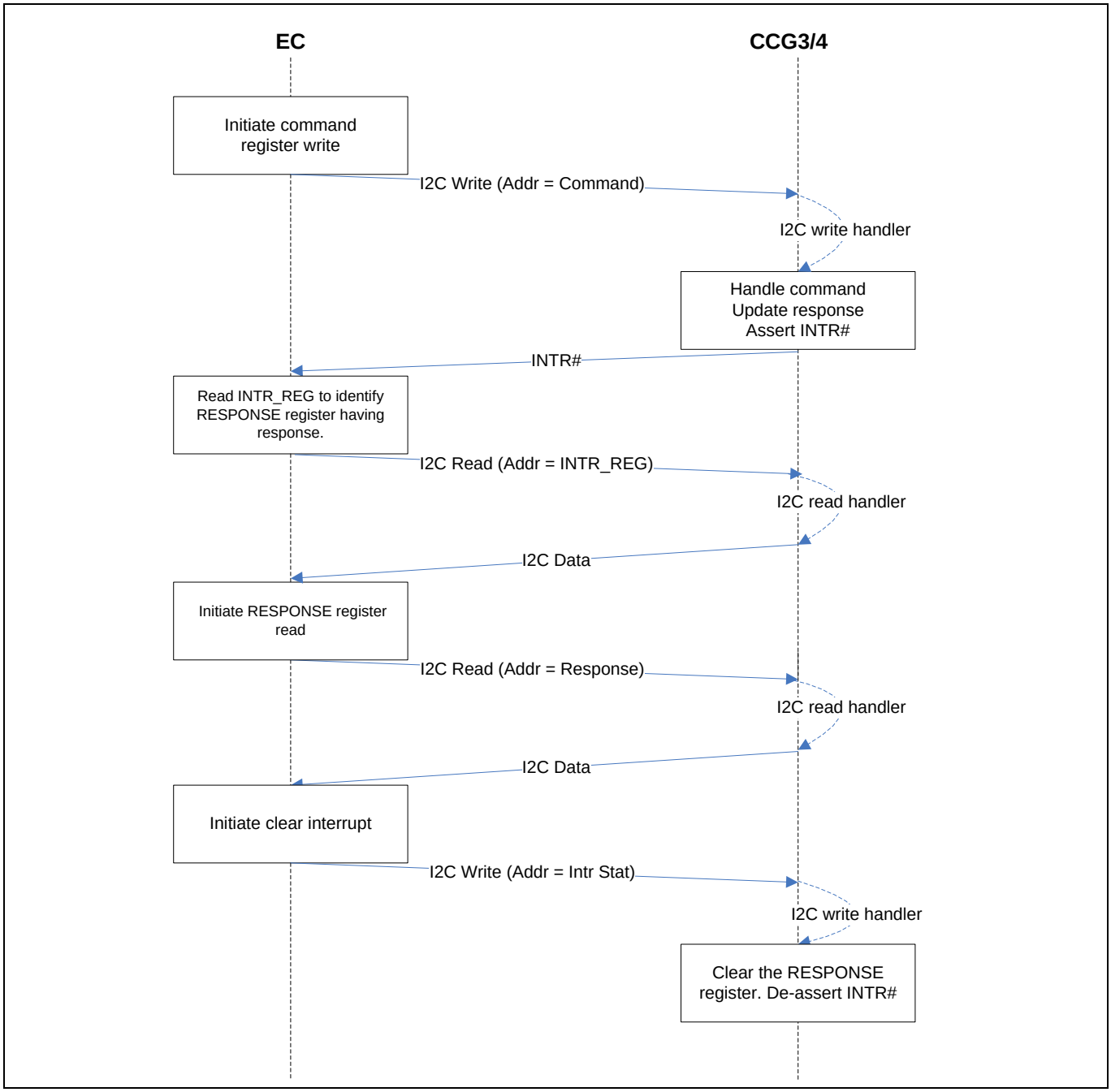

Figure 1 HPI Command-Response Model

2.1.2 Asynchronous PD Message and Event Reporting

In addition to issuing responses to commands received, the PD_RESPONSE registers are also used by CCG3/4 to notify the EC about asynchronous PD messages and device events. **Figure 2** shows the asynchronous PD message and event reporting mechanism between the EC and CCG3/4 over the HPI. CCG3/4 will assert the INTR# pin when a new response or event is stored in the RESPONSE register. The EC is expected to clear the interrupt after reading the response/event from the registers.

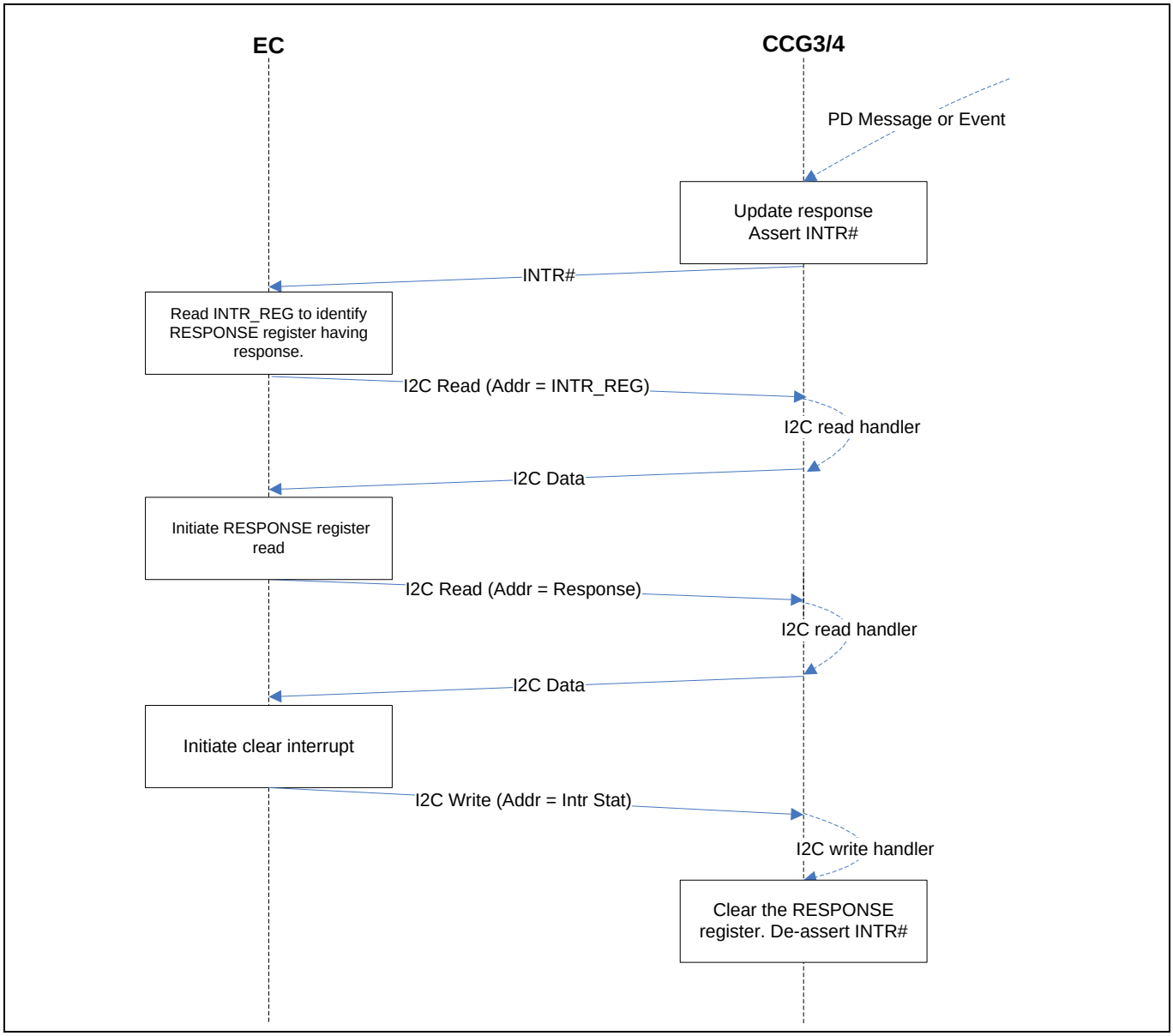

Figure 2 Asynchronous PD Message and Event Reporting

- **Asynchronous PD message and event reporting in CCG3/4 device application firmware mode:** CCG3/4 notifies the EC about any state changes or command exchanges (such as Type-C port connect/disconnect detected) happening on the USB-PD interface through asynchronous events. **Table 2** shows the various event codes used by CCG3/4 to communicate updates to the EC.

Table 2 Events and Asynchronous Message Codes

Message Code(1 Byte)	Description
Device-Specific Events	
0x80	Reset Complete. Device underwent a reset and is back in operation mode. CCG uses this event as an indication to EC that CCG is ready to receive commands after initialization is complete.
0x81	Message Queue Overflow. Message queue overflow detected.

Message Code(1 Byte)	Description
Type-C Specific Events	
0x82	Over Current Detected.
0x83	Over Voltage Detected.
0x84	Type C Port Connect Detected.
0x85	Type C Port Disconnect Detected.
PD Control Messages and Contract-Specific Events	
0x86	PD Contract Negotiation Complete.
0x87	SWAP Complete. EC is expected to start moving to new power requirements in sink mode within tSnkNewPower(max) – 15 ms and should complete within tSrcTransition(min) – 25 ms.
0x88-0x89	Reserved
0x8A	PS_RDY Message Received. EC is expected to meet new power requirements in sink mode within 15 ms.
0x8B	GotoMin Message Received. On receiving a GotoMin Message, the EC is expected to reduce its power consumption to the previously agreed minimum value within tSnkNewPower(max) – 15 ms. The EC should initiate contract renegotiation for returning to previous current level. The GotoMin request is valid until the next contract re-negotiation or disconnect event.
0x8C	Accept Message Received.
0x8D	Reject Message Received.
0x8E	Wait Message Received.
0x8F	Hard Reset Received.
PD Data Message-Specific Events	
0x90	VDM Received. This event indicates that CCG received a VDM from Port Partner. See section **4.3.2**. When acting as a UFP, the response VDM should be sent back within tVDMSenderResponse(max) specified by the USBPD specification. To support all command handling, the request should be responded to within 25 ms.
Capability Message-Specific Events	
0x91	Source Capabilities Message Received.
0x92	Sink Capabilities Message Received.
Resets and Error Scenario Events	
0x9A	Hard Reset Sent to Port Partner This event is reported when CCG3/4 sends HARD_RESET to port partner.
0x9B	Soft Reset Sent to Port Partner Section **4.2.11**
0x9C	Cable Reset Sent to EMCA Section **4.2.12**
0xA0	Unexpected Voltage on VBUS CCG3/4 notifies EC with this event if CCG3/4 is DFP and unexpected voltage is detected on VBUS before CCG3/4 turns on VBUS. CCG3/4 does not continue with TYPE C Connect tasks in this case and does not start PD tasks. CCG3/4 stays in this state until a TYPE C disconnect. EC can choose to disable the TYPE C interface of CCG3/4 using the

Message Code(1 Byte)	Description
	PORT_DISBALE command in the PD_CONTROL register or notify the user to disconnect the port partner from the TYPE C connector.
0xA1	Type C Error Recovery CCG3/4 notifies EC with this event when CCG3/4 executes TYPE C ERROR Recovery.
0xA2-0xA5	Reserved
Miscellaneous Events	
0xAA	Rp Change Detected CCG notifies EC with this event when CCG3/4 detects a change in the TYPE C “Rp” resistor value in Sink Mode. Note that Type C Source uses the Rp resistor to advertise the TYPE C Current level if PD Connection does not exist between the Port Partners. After a TYPE C Connection is established, CCG3/4 monitors the CC Voltage to determine the Rp resistor value. If Rp changes and PD Contract does not exist, CCG3/4 notifies EC with this event. The TYPE C status register’s TYPE C Current field holds the present Rp value. EC is expected to adjust the current consumption over the TYPE C Interface accordingly. The current adjustment should be done within tSinkAdj(max) - tPDDebounce(max) = 60 ms – 20 ms = 40 ms.
Alternate-Mode-Related Events	
0xB0	Alternate Mode event This event is sent for notification of Alternate mode-specific conditions like Alternate mode discovery, mode entry, and mode exit. The Event data specifies the SVID corresponding to the alt. mode, the event type, and any event-specific data.
0xB1	Alternate Mode Hardware event This event is sent for notification of the Alternate mode control hardware (MUX, HPD signal, etc.) state changes. The event data specifies all of the information related to the event.

These events need to be enabled through explicit event mask updates performed at start-up time by the EC. CCG3/4 maintains internal queues to store these outstanding events, responses, and PD messages if the EC has not completed the read of the current message in the RESPONSE register. Separate queues are maintained for each USB PD port as well as for device-specific commands and events. Note that there is only one INTR# signal, which gets asserted when any Type-C or PD activity is detected on either of the Type-C ports. EC reads the RESPONSE register over HPI for both the Type-C ports as shown in **Figure 5**. CCG3/4 does not de-assert INTR#, even after the EC writes to INTR_REG, if more events or messages are available in the internal queue. The EC must read out all the queued events and messages before sending a new command. It should always check the status of INTR# before sending a new command. The depth of the message queue is 8 events/responses per port. If the message queue overflows, CCG3/4 drops the last queued message and replaces it with a Message Queue Overflow event.

- **Asynchronous PD message and event reporting in boot mode:** The CCG3/4 device does not maintain event queues in boot mode. It also does not support Type-C and PD functionality in boot mode. If the EC sends a new command before reading the response of a previous command, the CCG3/4 bootloader overwrites the RESPONSE register with the new response code and asserts INTR# pin to notify the EC that a response is available in a register.

  In certain situations, CCG3/4 may not be able to handle the EC’s command, for example, if a command is received from the EC when the CCG3/4 device is busy handling PD transactions on the Type-C interface.

CCG3/4 device responds with a PD Command Failed response status, and the EC may retry the command in following situations:

- If EC requests to transmit a PD control/data message when the Type-C port is not connected or does not exist.
- If EC issues a PD policy command when CCG3/4 is actively communicating on the Type-C interface.
- If CCG3/4 is not in PD Source/Sink Ready states as defined by **USB Power Delivery (PD) Specification Revision 2.0**.

2.1.3 Data Memory Read and Write Operations

The CCG3/4 HPI implementation supports separate data memory regions for device-specific USB PD PORT_0, and USB PD PORT_1 (applicable only to CCG4) communication. All data memories are divided into two regions: The upper 512 bytes are write only, and the lower 512 bytes are read only (which includes 508 bytes read data memory and 4 bytes PD_RESPONSE register) as shown in **Figure 8**.

CCG3/4 collects data written by the EC in an 8-byte-deep RX FIFO (internal hardware FIFO). It uses FIFO interrupts to determine the availability of data. CCG3/4 treats the first 2 bytes of the transaction as register address and the subsequent data bytes as register data. It collects data from the RX FIFO and keeps track of the internal write pointer. If the write pointer crosses the 128 or 256 byte boundary (as each flash memory row of CCG3 is 128 bytes and CCG4 is 256 bytes), CCG3/4 treats the write operation as invalid and drops the data bytes written by the EC.

CCG3/4 uses an I^2C NACK mechanism to indicate error conditions to the EC. CCG3/4 may not be able to NACK the very first byte that crosses the 128-/256-byte boundary because it may be busy with other Type-C and PD interface interrupts such as Type-C attach or detach, initiation of power role swap or data role swap etc. Since RX FIFO is 8 bytes deep, one of the first 8 bytes that crosses the 128-/256-byte boundary is NACKed by CCG3/4.

Any data memory access operation (valid or invalid) does not result in response or event generation. If a read operation crosses the data memory boundary, the read pointer wraps around to offset 0. If a write operation crosses the data memory boundary, CCG3/4 drops further data bytes written by the EC. No action is taken if the write transaction fails or data memory does not get updated.

2.2 Power-Up Initialization Sequence for CCG3/4 and EC

After a power up, CCG3/4 will go through the bootloader and will execute the application firmware upon successful validation of the firmware image. The bootloader execution takes approximately 150 msec. **Figure 3** shows the CCG3/4 power-up initialization sequence. After the application is launched, a 100-msec timer is enabled to allow the EC to change the settings of the CCG3/4 device before establishing the USB Type-C power contract.

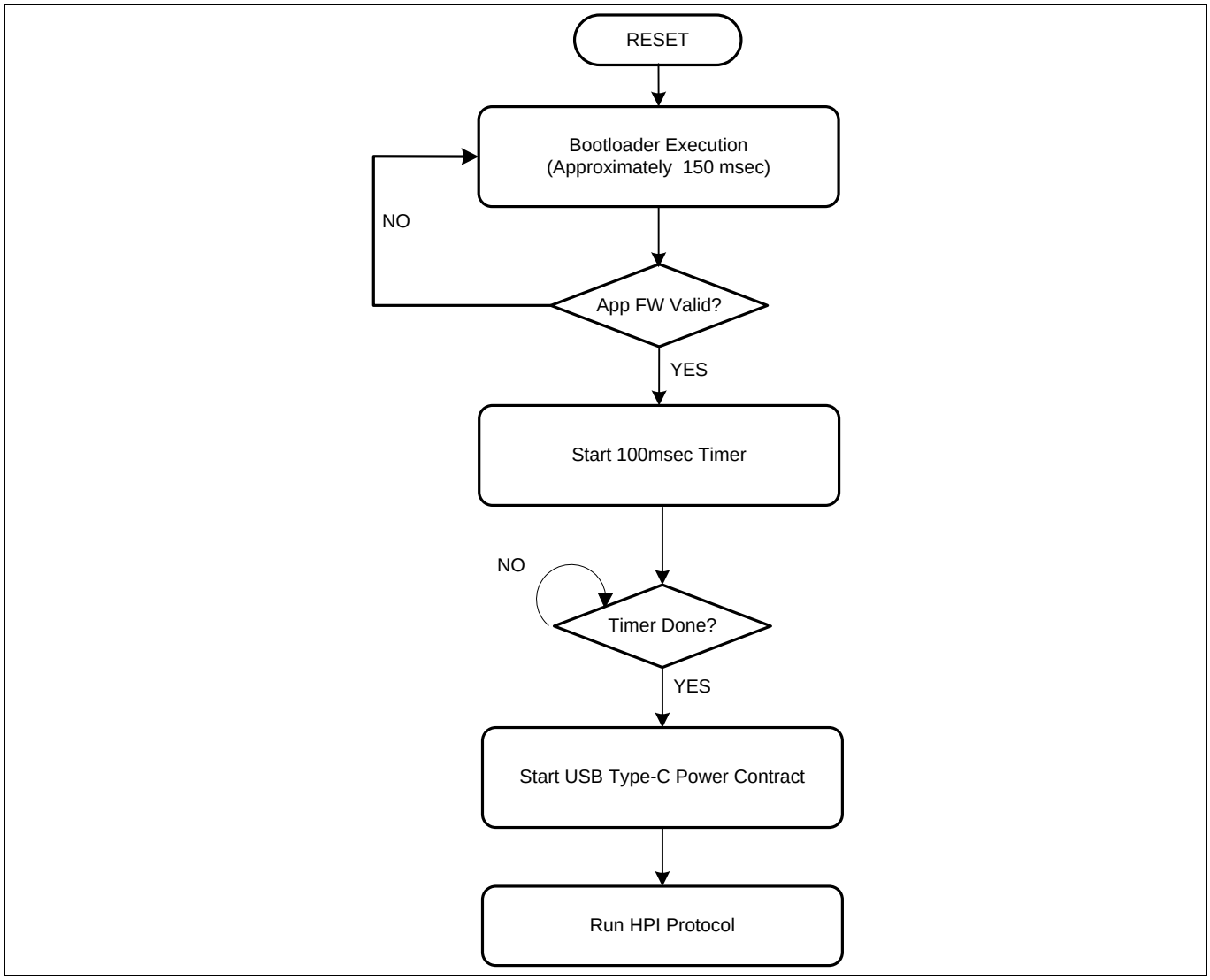

Figure 3 Power-Up Initialization of CCG3/4

After the EC powers up, it may optionally initialize CCG3/4 and update its firmware, if required. The firmware update procedure for the CCG3/4 device is explained in section **4.1**. The EC uses the EVENT_MASK register to choose the events notified by the CCG3/4 device. By default, CCG3/4 handles all events and asynchronous PD messages autonomously and does not notify the EC. Depending on the mask value in this register, the EC can choose which events and messages will be notified. The EC uses the SELECT_SINK_PDO register to select sink mode PDOs at run time when the CCG3/4 device is configured as a power consumer or sink device. A typical EC boot-up sequence is shown in **Figure 4**.

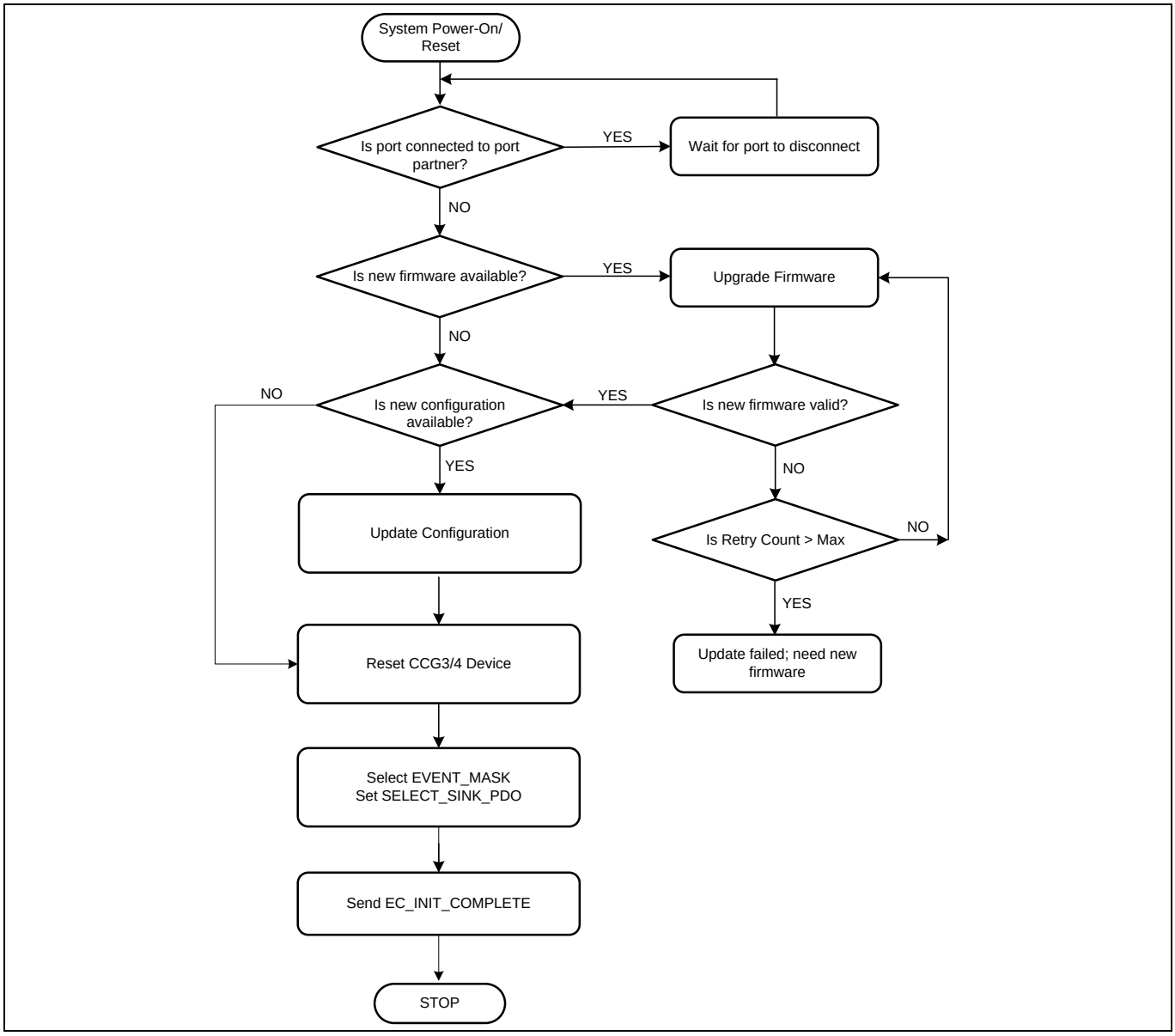

Figure 4 Typical EC Boot-Up Sequence

2.3 HPI Physical Layer

The physical connection to the EC is an I²C interface with an additional interrupt line (INTR#, active low signal), as shown in **Figure 5**. The CCG3/4 device will pull the INTR# pin low when it requires attention. CCG3/4 implements the HPI as an I²C slave interface (supported clock frequencies are 1 MHz, 400 kHz, and 100 kHz) and requires that the EC's I²C master support clock stretching. The CCG3/4 device has a 12-KB register space, which can be accessed over the I²C interface using a 2-byte register address.

The CCG3/4 device's I²C slave address (7 bit) can be set to one of three addresses depending on the bias set on the SWD_CLK pin of the CCG3/4 device after reset. The CCG3/4 device uses the state of SWD_CLK IO to configure the I²C slave address, which should be held stable at the desired value for a minimum of 200 ms (This 200ms time window includes the boot wait window of 100 ms and time taken to validate the application firmware image) whenever the CCG3/4 device goes through reset or performs a jump to bootloader/firmware operation. **Table 3** shows the HPI slave address selection options. If more than three CCG3/4 device addresses are needed, contact **ccg@cypress.com** to obtain customized firmware.

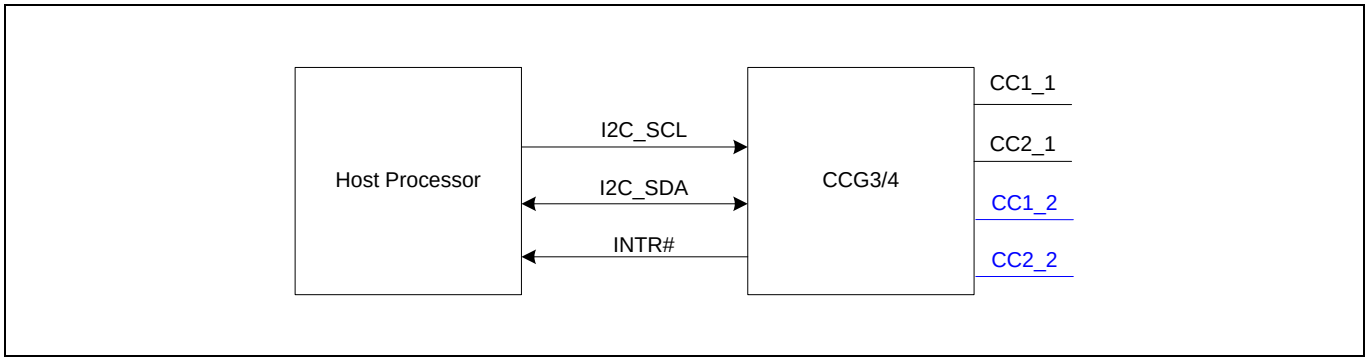

Figure 5 HPI Implemented as I²C Slave Interface

Table 3 Selecting the CCG3/4 I²C Slave Address

SWD_CLK	Slave Address
Floating	0x08
Pulled Low (1 kΩ)	0x40
Pulled High (1 kΩ)	0x42

Note: The CC1_2 and CC2_2 lines correspond to the second PD port, which is currently available only on CCG4. Pin#15 of the CCG4 device is a fixed-function I/O, which is configured as an I²C interrupt pin. While any CCG3 device's GPIO can be configured as an I²C interrupt pin, care should be taken to ensure that the application firmware and bootloader utilize the same GPIO as an interrupt pin.

2.4 HPI Transport Layer

This section explains I²C bus communication and INTR# GPIO-specific details for the CCG3/4 register space's read and write operations. CCG3/4 supports the following read and write operations.

2.4.1 I²C Write to CCG3/4

Figure 6 shows the I²C transfer sequence for a 2-byte write operation from the CCG3/4 device.

The first 2 bytes following a write preamble (device address) are the start address of the register write. The LS byte of the address is transferred first, followed by the MS byte. All bytes following the address bytes are the register data. CCG3/4 receives data in a temporary buffer and updates the register space only after STOP signal is received. The register space is updated after validation that the fields that are write-enabled are updated.

Write restart is not supported. Writes across invalid address regions are not supported. If the EC updates more than one command register data in a single write, the first command is processed, and the rest of the commands are ignored. Partial and unaligned register writes are treated as errors, and CCG3/4 responds with the Invalid Command code.

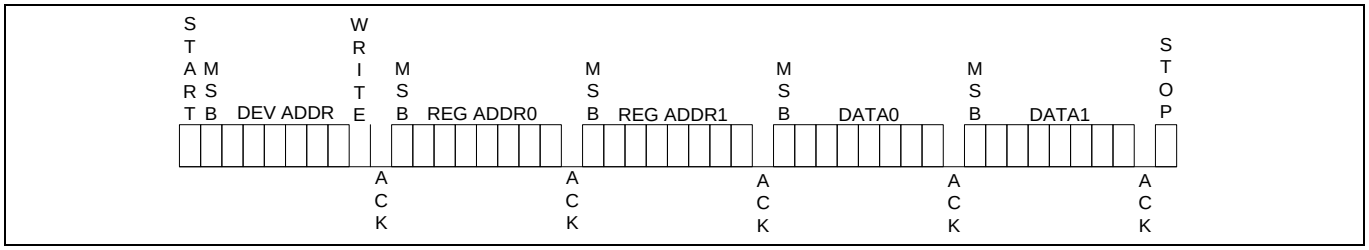

Figure 6 I²C Write to CCG3/4

2.4.2 I²C Read from CCG3/4

The EC uses this operation to read from the HPI register space. It has two phases:

- EC writes the register address followed by a restart on the bus instead of a stop.
- EC sends the read preamble and starts to read the data from the register space.

The maximum read size is 512 bytes. Reads beyond the valid address region will be NACKed by the CCG3/4 device. **Figure 7** shows the I²C transfer sequence for a 2-byte read operation from the CCG3/4 device.

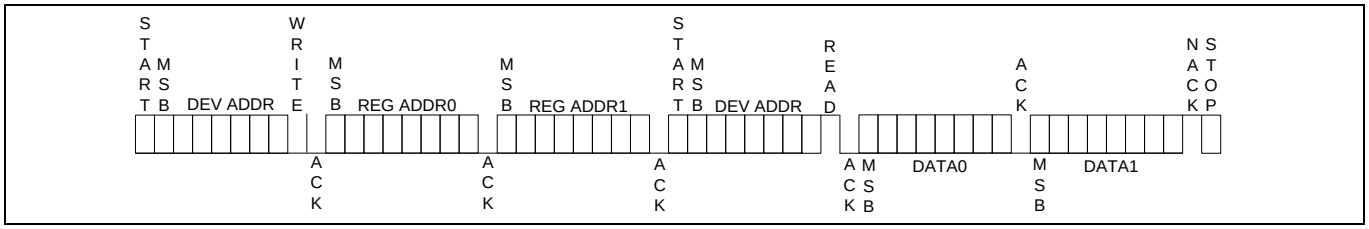

Figure 7 I²C Read from CCG3/4

The CCG3/4 device will NACK the read preamble if the EC does not write a register address followed by a restart before reading the register contents.

The I²C master on the EC must support clock stretching. The CCG3/4 device stretches the I²C clock in the following scenarios:

- At the ACK/NACK phase of the preamble: The preamble byte received from the EC is not automatically responded to with an ACK/NACK by the CCG3/4's hardware I²C block. The CCG3/4 device's firmware responds to the preamble byte received from the EC. If CCG3/4 is servicing Type-C and PD interrupts while the preamble is received, the clock is stretched until CCG3/4's firmware services the I²C request.
- The CCG3/4 device has an RX FIFO of 8 bytes to receive data transmitted by the EC. The clock is stretched at the ACK phase of the data byte if the RX FIFO is full and CCG3/4 cannot receive any more data. CCG3/4 stops the clock stretch and ACKs the next data byte when the CCG3/4 device reads the RX FIFO contents and the FIFO has space to receive subsequent data bytes.

2.4.3 INTR# GPIO

INTR# GPIO is an active low signal. CCG3/4 drives INTR# GPIO low to notify the EC of responses, events, and asynchronous messages. The drive mode of INTR# GPIO is open drain. An external pull-up is required to detect INTR# GPIO assertion.

2.5 HPI Register Overview

CCG3 supports one USB PD port whereas CCG4 supports two USB PD ports, which can be independently configured and used. A 2-byte addressed register space is used for CCG3/4 devices, so that the implementation allows for the addition of new registers and provides independent sets of registers to manage each USB PD port.

Figure 8 shows the HPI register space, which is divided into three sections as follows:

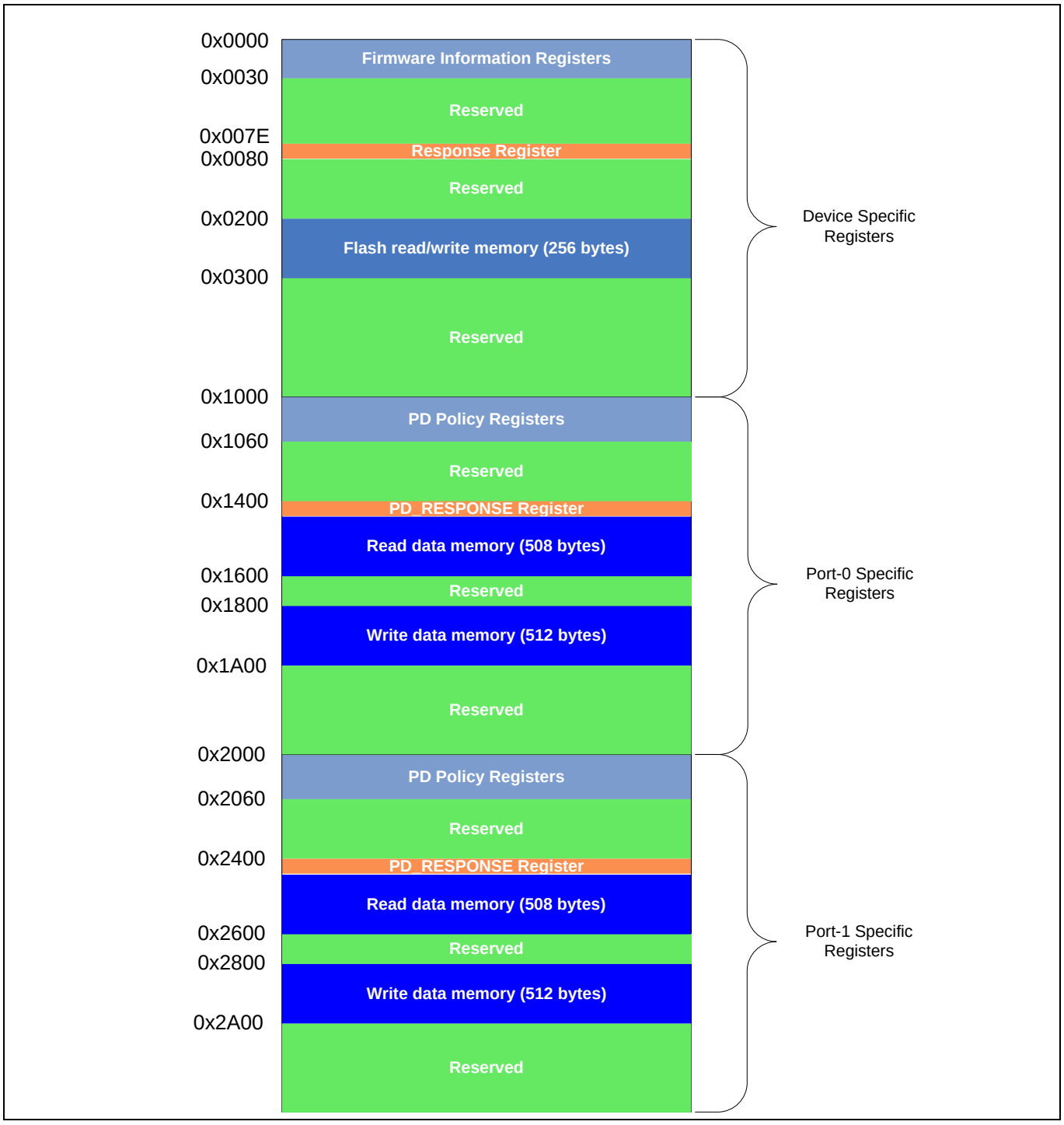

Figure 8 HPI Register Overview

- Device-specific registers (0x0000-0x0FFF): This space contains all the command registers and status registers common to the CCG3/4 device (and unrelated to specific USB PD ports). It also contains the RESPONSE register, data memory, and flash read/write memory.
  - Status registers: These registers provide information about the operating mode (such as firmware and bootloader) of the CCG3/4 device.
  - Command registers: These registers are used to send device-level commands such as flash read/write to the CCG3/4 firmware.
  - RESPONSE register: This space is used to respond to device-specific commands and to notify device-specific events to the EC.
  - Flash read/write memory: This space is used as a buffer to read or write a complete flash row.
- Port-specific registers for PORT_0 (0x1000-0x1FFF): This space contains all the command registers, status registers, and PD policy registers related to Type-C port 0. Only PORT_0 will be available for single Type-C port devices such as CCG3 and the single Type-C port CCG4 (CYPD4125) device. It also contains a RESPONSE register similar to the device-specific registers and the data memory.
- Port-specific registers for PORT_1 (0x2000-0x2FFF): This space is applicable only to the CCG4 device. These registers are related to Type-C port 1 on the CCG4 device and have the same structure as the PORT_0 registers.

Refer to HPI Register Set, for further details on the HPI register set.

2.6 HPI Differences between CCG1/2 and CCG3/4

Differences in the HPI implementation for CCG1/2 and CCG3/4 are as follows:

- CCG4 device has flash memory with 256-byte rows, requiring a 256-byte data memory region to do flash read/writes.
- CCG3/4 devices support additional features such as Thunderbolt mode that require additional PD configuration and status registers.
- CCG3/4 devices support a dual firmware application usage model, which requires additional registers to report firmware information.
- CCG4 device has two USB PD ports, which need to be independently managed through different sets of registers
- The configuration table is attached to the application firmware as a single firmware image file in CCG3/4 devices.

The changes in HPI definition are made in such a way that changes to the HPI implementation on the EC side are minimized. In summary, the changes made are:

- I^2C addressing is 2 bytes instead of 1 byte. This change allows separate sets of registers for each USB-PD port and also provides scalability for future implementations with more than two ports.
- I^2C addresses for each register are changed to implement a hierarchy based on the functionality of the registers. The addressing is done so as to make it easy for the EC to calculate the address for each PD port.
- New registers and fields are added in the device information registers to provide information about two firmware images.
- New PD configuration registers are added to support new firmware features (such as dynamic PDO update).

3 HPI Register Set

This section explains the CCG3/4 device's register set in detail. The HPI registers are listed in **Table 4**. They are categorized into device-specific registers and port-specific registers. Port-specific registers are further classified into PD registers, VDM registers, DisplayPort registers, and Response registers. The register set in **Table 4** is color coded based on the different types, as shown in **Figure 9**.

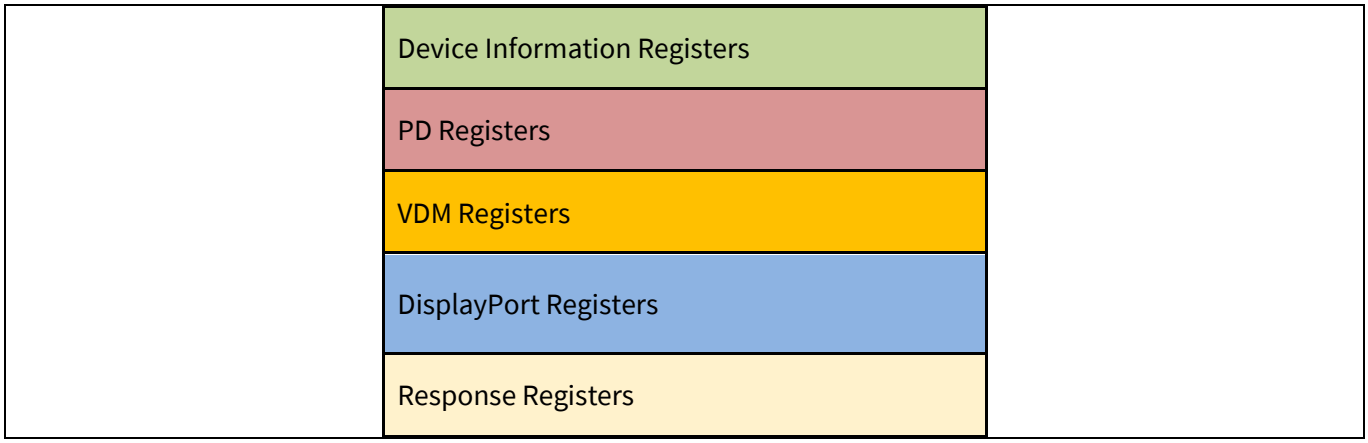

Figure 9 Types of HPI Registers

Note that registers are 1 byte, 2 bytes, or 4 bytes in length, and one of them is 16 bytes in length. The byte order for multi-byte registers is little-endian.

Most of the registers are status registers that the EC can read to discover the current state of the CCG3/4 device. The CCG3/4 device updates these registers at run time. Control registers determine how the CCG3/4 device responds to certain events such as role swaps. Command registers are used by the EC to trigger changes such as source/sink PDO selection and VDM transmission. The EC and CCG3/4 use a command-response model, as explained in section **2.1.1**. **It is mandatory to read the response for a previous command prior to initiating the next command.**

Table 4 Definition of HPI Registers

Address	Name	Field	Mode	Access	Section
0x0000	DEVICE_MODE	Current mode	BOOT/FW	R	**3.1.1**
0x0001	BOOT_MODE_REASON	Reason	BOOT	R	**3.1.2**
0x0002	READ_SILICON_ID	Silicon ID LSB	BOOT/FW	R	**3.1.3**
0x0003	READ_SILICON_ID	Silicon ID MSB	BOOT/FW	R	
0x0004	BOOT_LOADER_LAST_ROW	Last Row LSB	BOOT	R	**3.1.4**
0x0005	BOOT_LOADER_LAST_ROW	Last Row MSB	BOOT	R	
0x0006	INTR_REG	Interrupt	BOOT/FW	R/W	**3.1.5**
0x0007	JUMP_TO_BOOT	Signature	CTRL	W	**3.1.6**
0x0008	RESET	Signature	CTRL	W	**3.1.7**
0x0009	RESET	Type	CTRL	W	
0x000A	ENTER_FLASHING_MODE	Signature	BOOT	W	**3.1.8**
0x000B	VALIDATE_FW	FW Mode	BOOT	W	**3.1.9**
0x000C	FLASH_ROW_READ_WRITE	Signature	BOOT	W	**3.1.10**
0x000D	FLASH_ROW_READ_WRITE	Command	BOOT	W	

Address	Name	Field	Mode	Access	Section
0x000E	FLASH_ROW_READ_WRITE	Row Number LSB	BOOT	W	
0x000F	FLASH_ROW_READ_WRITE	Row Number MSB	BOOT	W	
0x0010	READ_ALL_VERSION	BL Build Number LSB	BOOT/FW	R	3.1.11
0x0011	READ_ALL_VERSION	BL Build Number MSB	BOOT/FW	R	
0x0012	READ_ALL_VERSION	BL Patch Version	BOOT/FW	R	
0x0013	READ_ALL_VERSION	BL Version Major(7:4) Minor(3:0)	BOOT/FW	R	
0x0014	READ_ALL_VERSION	BL Application Name LSB	BOOT/FW	R	
0x0015	READ_ALL_VERSION	BL Application Name MSB	BOOT/FW	R	
0x0016	READ_ALL_VERSION	BL External Circuit Specific Version	BOOT/FW	R	
0x0017	READ_ALL_VERSION	BL Version Major(7:4) Minor(3:0)	BOOT/FW	R	
0x0018	READ_ALL_VERSION	FW Build Number LSB	BOOT/FW	R	
0x0019	READ_ALL_VERSION	FW Build Number MSB	BOOT/FW	R	
0x001A	READ_ALL_VERSION	FW Patch Version	BOOT/FW	R	
0x001B	READ_ALL_VERSION	FW Version Major(7:4) Minor (3:0)	BOOT/FW	R	
0x001C	READ_ALL_VERSION	FW Application Name LSB	BOOT/FW	R	
0x001D	READ_ALL_VERSION	FW Application Name MSB	BOOT/FW	R	
0x001E	READ_ALL_VERSION	FW External Circuit Specific Version	BOOT/FW	R	
0x001F	READ_ALL_VERSION	FW Version Major (7:4) Minor (3:0)	BOOT/FW	R	
0x0020	FW2_VERSION	FW2_VERSION[0]	BOOT/FW	R	3.1.12
0x0021	FW2_VERSION	FW2_VERSION[1]	BOOT/FW	R	
0x0022	FW2_VERSION	FW2_VERSION[2]	BOOT/FW	R	
0x0023	FW2_VERSION	FW2_VERSION[3]	BOOT/FW	R	
0x0024	FW2_VERSION	FW2_VERSION[4]	BOOT/FW	R	
0x0025	FW2_VERSION	FW2_VERSION[5]	BOOT/FW	R	

Address	Name	Field	Mode	Access	Section
0x0026	FW2_VERSION	FW2_VERSION[6]	BOOT/FW	R	
0x0027	FW2_VERSION	FW2_VERSION[7]	BOOT/FW	R	
0x0028	FIRMWARE_BINARY_LOCATION_REGISTER	FW1_START[0]	BOOT/FW	R	3.1.13
0x0029	FIRMWARE_BINARY_LOCATION_REGISTER	FW1_START[1]	BOOT/FW	R	
0x002A	FIRMWARE_BINARY_LOCATION_REGISTER	FW2_START[0]	BOOT/FW	R	
0x002B	FIRMWARE_BINARY_LOCATION_REGISTER	FW2_START[1]	BOOT/FW	R	
0x002C	PDPORT_ENABLE	Enables or disables PD port	FW	RW	3.1.14
0x002D	SLEEP_CTRL	Controls Deep Sleep mode	FW	RW	3.1.15
0x002E	BATTERY_STAT	Informs about dead battery conditions	FW	RW	3.1.16
0x0040	VENDOR SPECIFIC	Vendor specific register space (48 bytes)	FW	RW	3.1.17
0x007E	RESPONSE_REGISTER	Response/Message Code	STATUS	R	3.2.3.1
0x007F	RESPONSE_REGISTER	Length	STATUS	R	
0x1000	VDM_CONTROL	VDM mode	CMD	W	3.3.1
0x1001	VDM_CONTROL	Length	CMD	W	
0x1002	EFFECTIVE_SOURCE_PDO_MASK	PDO Mask	STATUS	R	3.2.1.6
0x1003	EFFECTIVE_SINK_PDO_MASK	PDO Mask	STATUS	R	3.2.1.7
0x1004	SELECT_SOURCE_PDO	Command	CMD	W	3.2.2.2
0x1005	SELECT_SINK_PDO	Command	CMD	W	3.2.2.4
0x1006	PD_CONTROL	Command	CMD	W	3.2.2.6
0x1008	PD_STATUS	Status Bit Map (b7:0)	STATUS	R	3.2.1.2
0x1009	PD_STATUS	Status Bit Map (b15:8)	STATUS	R	
0x100A	PD_STATUS	Status Bit Map (b23:16)	STATUS	R	
0x100B	PD_STATUS	Status Bit Map (b31:24)	STATUS	R	
0x100C	TYPE_C_STATUS	Status Bit Map	STATUS	R	3.2.1.1
0x1010	CURRENT_PDO	PDO LSB	STATUS	R	3.2.1.3
0x1011	CURRENT_PDO	PDO	STATUS	R	
0x1012	CURRENT_PDO	PDO	STATUS	R	
0x1013	CURRENT_PDO	PDO MSB	STATUS	R	
0x1014	CURRENT_RDO	RDO LSB	STATUS	R/W	3.2.1.4
0x1015	CURRENT_RDO	RDO	STATUS	R/W	
0x1016	CURRENT_RDO	RDO	STATUS	R/W	

Address	Name	Field	Mode	Access	Section
0x1017	CURRENT_RDO	RDO MSB	STATUS	R/W	
0x1018	CURRENT_CABLE_VDO	VDO LSB	STATUS	R	3.2.1.5
0x1019	CURRENT_CABLE_VDO	VDO	STATUS	R	
0x101A	CURRENT_CABLE_VDO	VDO	STATUS	R	
0x101B	CURRENT_CABLE_VDO	VDO MSB	STATUS	R	
0x101C	ALT_MODE_CMD	SVID[0]	CTRL	R/W	3.4.1
0x101D	ALT_MODE_CMD	SVID[1]	CTRL	R/W	
0x101E	ALT_MODE_CMD	Alternate Mode ID	CTRL	R/W	
0x101F	ALT_MODE_CMD	Alternate Mode Data Role	CTRL	R/W	
0x1020	APP_HW_CMD	CCG Data Role	CTRL	R/W	3.4.2
0x1021	APP_HW_CMD	Hardware Role	CTRL	R/W	
0x1022	APP_HW_CMD	Command ID[0]	CTRL	R/W	
0x1023	APP_HW_CMD	Command ID[1]	CTRL	R/W	
0x1024	EVENT_MASK	Response Filter (b7:0)	CTRL	R/W	3.2.2.7
0x1025	EVENT_MASK	Response Filter (b15:8)	CTRL	R/W	
0x1026	EVENT_MASK	Response Filter (b23:16)	CTRL	R/W	
0x1027	EVENT_MASK	Response Filter (b31:24)	CTRL	R/W	
0x1028	SWAP_RESPONSE	Response Override	CMD	R	3.2.2.1
0x1029	ACTIVE_EC_MODES	Used by EC to indicate if it is maintaining active EC modes	STATUS	R/W	3.4.3
0x102A	VDM_EC_CONTROL	Used by EC to indicate that it is ready to handle VDMs	CTRL	R/W	3.3.2
0x1030	CMD_TIMEOUT	Timeout periods for VDMs and PD commands	CTRL	R/W	3.2.2.8
0x1031	PORT_INTR_STATUS	Reports the PD port status	STATUS	R/W	3.2.1.8
0x1032	PORT_INTR_STATUS	Reports the PD port status	STATUS	R/W	
0x1033	PORT_INTR_STATUS	Reports the PD port status	STATUS	R/W	
0x1034	PORT_INTR_STATUS	Reports the PD port status	STATUS	R/W	

Address	Name	Field	Mode	Access	Section
0x1400	PD_RESPONSE	Reports PD responses	STATUS	R	
0x1401	PD_RESPONSE	Reports PD responses	STATUS	R	

The following sections describe each register in detail. Rather than presenting them in numerical order, they use a logical order since several registers are linked and need an introduction to their category.

3.1 Device Information Registers

This section covers the system registers that can be read at any time.

3.1.1 DEVICE_MODE

This register indicates the active device mode, as described in **Table 5**. The CCG3/4 device can be either in bootloader or normal (application firmware) operation mode. The device is in boot mode only if the application firmware image is not valid or if the application requests the CCG3/4 device to jump to boot mode. This register is available for access in both the bootloader and normal firmware mode.

Table 5 DEVICE_MODE Register

DEVICE_MODE: 1 Byte CCG3/CCG4 Address : 0x0000			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Current mode.	R	b7 0 – HPIv1 mode. Single-byte HPI addressing, only bootloader can do flash read/write. 1 – HPIv2 mode. Two-byte HPI addressing. Dual firmware mode with two copies of firmware that can do mutual updates. b6-b4: Flash row size 0 – 128 bytes 1 – 256 bytes Other values reserved b3-b2: Number of PD ports supported 0 – 1 port 1 – 2 ports Other values reserved b1-b0: 0 – Boot mode 1 – Firmware 1 2 – Firmware 2

3.1.2 BOOT_MODE_REASON

This register specifies why the CCG3/4 device is in boot mode instead of normal operation mode, as described in **Table 6**. The EC can read this register to determine the reason for boot mode if CCG3/4 stays in boot mode for a long time. This register is available for access only in boot mode. It contains the value 0x00 in normal operation mode.

Table 6 BOOT_MODE_REASON Register

BOOT_MODE_REASON: 1 Byte
CCG3/CCG4 Address: 0x0001

Field	Field Name	R/W	Description
Byte 0	Reason	R	b0 : Boot mode request by firmware 0 – No boot mode request 1 – Firmware switched to boot mode due to JUMP_TO_BOOT request. b1 : Configuration Table Status 0 – Table valid 1 – Table invalid Note: This bit is not used in CCG3/4 as the configuration table is part of firmware. b2: Firmware application 1 status 0: Application image valid 1: Application image invalid b3: Firmware application 2 status 0: Application image valid 1: Application image invalid b4:b7: Reserved. This nibble is always 0.

3.1.3 READ_SILICON_ID

This register contains the upper 2 bytes of the Silicon ID of the CCG3/4 device, as described in **Table 7**. It is available for access in both bootloader and normal firmware mode.

Table 7 READ_SILICON_ID Register

READ_SILICON_ID: 2 Bytes
CCG3/CCG4 Address : 0x0002

Field	Field Name	R	Description
Byte[0:1]	Silicon ID	R	2-byte Silicon ID of device

3.1.4 BOOT_LOADER_LAST_ROW

This register holds the index of the last flash row occupied by the bootloader, as described in **Table 8**. Each flash row of the CCG3/4 device contains 128 (for CCG3) or 256 (for CCG4) bytes, and the first flash row is indexed 0. This is a read-only register and is available only in boot mode.

Table 8 BOOT_LOADER_LAST_ROW Register

BOOT_LOADER_LAST_ROW: 2 Bytes CCG3/CCG4 Address: 0x0004			
Field	**Field Name**	**R**	**Description**
Byte[0:1]	Last flash row number	R	Number of the last flash row occupied by bootloader

3.1.5 INTR_REG

If the CCG3/4 device needs attention, it drives the INTR# pin low and sets the bit 0,1 and 2 to 1, depending on whether the device and/or Type-C port 0/1 requires attention. The CCG3/4 device then expects the EC to service this interrupt. Once the EC has serviced the interrupt, the EC must clear the interrupt by writing 0 to this register, as described in **Table 9**. Once it is written, the CCG3/4 device drives the INTR# pin high assuming that there are no pending events for the EC to process.

If the CCG3/4 device has other events for the EC to process, this register cannot be cleared (and the INTR# line may not go back high). In such situations, the EC must read and process all remaining events until the INTR# line goes back high.

Table 9 INTR_REG Register

INTR_REG: 1 Byte CCG3/CCG4 Address: 0x0006			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	INTR	R/W	Read of this byte indicates which part of register space has raised the INTR GPIO. Only the DEV_INTR bit is valid for CCG1/2. b0: DEV_INTR 0: No response in device-specific RESPONSE register 1: A new response is available in device-specific RESPONSE register. b1: PORT0_INTR 0: No response in PORT_0 specific PD_RESPONSE register 1: A new response is available in PORT_0 specific PD_RESPONSE register. b2: PORT1_INTR 0: No response in PORT_1 specific PD_RESPONSE register 1: A new response is available in PORT_1 specific PD_RESPONSE register.

INTR_REG: 1 Byte CCG3/CCG4 Address: 0x0006			
Field	**Field Name**	**R/W**	**Description**
			EC will write to this register to clear corresponding bit after reading corresponding response. b3:b7: Reserved for future use. CCG3/4 ignores these bits.

3.1.6 JUMP_TO_BOOT

- Boot mode: This register forces the device to jump to boot mode from normal operation mode, as described in **Table 10**. If the device is already in boot mode, the CCG3/4 device responds with Invalid Command response code. The CCG3/4 device has primary and alternate firmware images. Firmware update is supported in boot mode if a valid firmware image (both primary and alternate) is not present in the CCG3/4 device; otherwise, the CCG3/4 device can perform a firmware update in dual firmware mode.
- Dual firmware mode: In this mode, this register can also be used to switch control to the alternate firmware image. The firmware will update the metadata that informs the bootloader about which firmware binary to load and then go through a reset.

The JUMP_TO_BOOT and JUMP_TO_ALT_FW commands can only be used when CCG3/4 device is not in a PD contract, as the device blocks will be disabled as part of the switch process. The EC can send the command to the CCG3/4 device to move out of the PD contract and disable the Type-C interface in normal operation mode by using the PDPORT_ENABLE register or by initiating the Port Disable command in the PD_CONTROL register. Refer to the **PD_CONTROL** description.

If JUMP_TO_BOOT or JUMP_TO_ALT_FW is requested while the PD blocks are active, CCG3/4 will return the INVALID COMMAND (0x05) response.

Note: The firmware switch is a one-time operation and will only be effective if the alternate firmware is found to be valid. CCG3/4 will again boot to the original firmware on subsequent resets or power-up cycles.

Table 10 JUMP_TO_BOOT Register

JUMP_TO_BOOT: 1 Byte CCG3/CCG4 Address : 0x0007			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Signature	R/W	JUMP_TO_BOOT Handle command only if this byte is loaded with a valid signature by EC: “J”. If signature is not valid, CCG3/4 updates RESPONSE register with Invalid Command error code. JUMP_TO_ALT_FW Handle command only if this byte is loaded by the signature “A”. If signature is not valid, CCG3/4 updates the RESPONSE register with Invalid Command error code. A read of this field always returns 0.

3.1.7 RESET

This register performs either a device reset or I²C block reset, as described in **Table 11**. With an I²C block reset, the I²C module is reconfigured. All outstanding commands and responses are flushed. With a device reset, CCG3/4 undergoes a software-initiated device reset.

The EC should send a Device Reset command only when the CCG3/4 device is not in a PD contract with the port partner. The EC can send the command to the CCG3/4 device to move out of the PD contract and disable the Type-C interface in normal operation mode using the PDPORT_ENABLE register or by initiating the Port Disable command in the PD_CONTROL register. Refer to the **PD_CONTROL** description. If a device reset is requested while the PD blocks are active, CCG3/4 will return the INVALID COMMAND (0x05) response.

Table 11 RESET Register

RESET: 2 Bytes @ Address: 0x08			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Initiate Reset	W	Initiate reset only if “R” (0x52) is written to this field. Writing any other value will cause CCG3/4 to update RESPONSE register with Invalid Command error code.
Byte [1]	Reset Type	W	0 – I²C reset 1 – Device reset Writing any other value will cause CCG3/4 to update RESPONSE register with Invalid Command error code.

3.1.8 ENTER_FLASHING_MODE

In boot mode, the EC uses this register to enable read and write to the CCG3/4 device’s flash, as described in **Table 12**. It uses this register to enter flashing mode. This register is used for the following operations.

The CCG3/4 device handles flash write and read commands only if it has entered flashing mode. The EC needs to update this register with the appropriate signature before sending any request to update or read the device flash. This provides an extra level of security before any operation is performed on the device flash.

This register also provides a mechanism to keep the CCG3/4 device in boot mode even if a valid normal mode application firmware image exists in the device flash.

Table 12 ENTER_FLASHING_MODE Register

ENTER_FLASHING_MODE: 1 Bytes CCG3/CCG4 Address: 0x000A			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Signature	R/W	Handle command only if this byte is loaded with valid signature by EC: “P”. If signature is not valid, CCG3/4 updates RESPONSE register with Invalid Command error code. A read of this field always returns 0.

CCG3/4 responds with Invalid Command response code if this command is received and a flash update is not supported. CCG3/4 processes flash update requests only if it has entered flashing mode.

3.1.9 VALIDATE_FW

The EC uses this register to check the validity of the firmware, as described in **Table 13**. This request is handled by the CCG3/4 device only in boot mode. After receiving this request, the bootloader computes the checksum of the entire firmware image and compares it with the checksum stored in the firmware metadata table. If both the checksums match, the bootloader responds with Success status code. But if the checksum comparison fails, the bootloader responds with Firmware Invalid status code. In dual firmware mode, the checksum comparison is done on the inactive firmware copy.

Table 13 VALIDATE_FW Register

VALIDATE_FW: 1 Byte
CCG3/CCG4 Address: 0x000B

Field	Field Name	R/W	Description
Byte[0]	FW Mode	R/W	CCG1/2: This field will contain 0x01. Any other value is invalid, and CCG1/2 responds with Invalidate Argument error code. CCG3/4: ID of the firmware to be validated. The value can be 0x01 or 0x02 based on whether FW1 or FW2 is to be validated. A read of this field always returns 0.

3.1.10 FLASH_ROW_READ_WRITE

This register is used by the EC to request the CCG3/4 device to read or write one row of flash, as described in **Table 14**. The row number to be used is specified in a particular field (bytes 2 and 3) of this register. This register should be accessed only after flash access is enabled by writing to the ENTER_FLASHING_MODE register.

The CCG3/4 device's flash area also contains the bootloader code in addition to the application firmware. Any read or write to the bootloader section of flash is discarded by CCG3/4. The EC, therefore, can only update the application firmware using this command.

This register allows the following operations:

- **Flash Row Write**: Updates the specified flash row (128/256 bytes).

  The EC needs to first write 128/256 bytes of data (that is, one row of flash) in the data memory space and then send this request. CCG3/4 writes this data to the flash row and then notifies the EC by driving the INTR# pin low and updating the RESPONSE register with the Success event.

- **Flash Row Read**: Requests the CCG3/4 device to read 128/256 bytes from the specified flash row.

  CCG3/4 reads the flash row, loads the data memory space with the flash data, and then notifies the EC by driving the INTR# pin low and updating the RESPONSE register with the Flash Data Available event.

This register can be used in the following modes:

- Legacy boot mode

In this mode, the CCG3/4 device's flash contains a fixed bootloader plus one copy of the complete firmware application. A flash update is supported only by the bootloader and only works for the flash rows allocated to the firmware application. Any update/read addressed to the bootloader flash rows are discarded.

- Dual FW mode

  In this mode, the CCG3/4 device's flash contains two copies of the complete firmware application. Flash row updates are allowed while the firmware application is running and do not require a switch to the bootloader.

Each firmware application is capable of updating the flash rows corresponding to the other firmware copy (FW1 can update FW2 and vice versa). The firmware will block any update/read on the flash rows allocated to the bootloader as well as the active firmware application. The EC is responsible for identifying the firmware copy that is active and providing the correct data to update the redundant copy.

In addition, the CCG3 device supports a seamless firmware update operation that allows firmware updates to be performed while the device is still in a PD contract and functioning normally. CCG4 requires PD contracts to be terminated and ports to be disabled before a firmware upgrade can be started.

Note that the CCG3/4 device's firmware will modify the data written to the firmware metadata row (rows 510 and 511) to add information that controls the firmware boot priority. This means that if these rows are read back, then CCG3/4 device will return data that is different from that originally written from the EC side.

Table 14 FLASH_ROW_READ_WRITE Register

FLASH_ROW_READ_WRITE: 4 Bytes **CCG3/CCG4 Address: 0x000C**			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Signature	R/W	Handle command only if this field is loaded with valid signature by EC: "F". If signature is not valid, CCG3/4 updates RESPONSE register with Invalid Command error code. A read of this field always returns 0.
Byte[1]	Command	R/W	0 – Flash Row Read 1 – Flash Row Write These requests are handled only if EC has requested CCG3/4 to enter flashing mode. Otherwise, CCG3/4 responds with Not Supported error code. If this field contains any other value, CCG3/4 responds with Invalid Command response code. If flash write operation is successful, CCG3/4 responds with Success response code. If flash read operation is successful, CCG3/4 responds with Flash Data Available response code. A read of this field always returns 0.
Byte [2-3]	Row Number	R/W	Row number of flash Some flash rows are reserved for bootloader. No update/read is performed on these rows. If EC sends invalid row number, request is not handled by CCG3/4, and RESPONSE register is updated with Invalid Argument error code. In dual firmware mode, flash read/writes can only be done on the inactive firmware copy. If Firmware1 is active, flash access is limited to rows above

FLASH_ROW_READ_WRITE: 4 Bytes CCG3/CCG4 Address: 0x000C			
Field	**Field Name**	**R/W**	**Description**
			FW2_START. If Firmware2 is active, flash access is limited to rows between FW1_START and FW2_START. A read of this field always returns 0.

3.1.11 READ_ALL_VERSION

The CCG3/4 device's firmware has two main sections: bootloader and application firmware. The application firmware can be a customized version of a base firmware. This register holds the complete version information of the bootloader, the application firmware, and the base firmware, as described in **Table 15**. Version information is 4 bytes long as described in **Table 16** and **Table 17**.

Table 15 READ_ALL_VERSION Register

READ_ALL_VERSION: 16 Bytes CCG3/CCG4 Address: 0x0010			
Field	**Field Name**	**R**	**Description**
Byte[0] to Byte[7]	BL Version	R	8-byte version information for the bootloader
Byte[8] to Byte[15]	FW Version	R	8-byte version information for the firmware application. In dual firmware applications, this field reports the version as FW1.

The format of the base firmware and base bootloader version is as follows.

Table 16 Base Version Format

Byte 0:1	**Build Number**
Byte 2	Patch Version
Byte 3 (Bit 0:3)	Minor Version
Byte 3 (Bit 4:7)	Major Version

The format of the application-specific firmware version is as follows.

Table 17 Application Firmware Version Format

Byte 0:1	**Application (Notebook) Name (0x6e62 : ASCII for "nb")**
Byte 2	External Circuit Specific Version
Byte 3 (Bit 0:3)	Minor Version
Byte 3 (Bit 4:7)	Major Version

3.1.12 FW2_VERSION

The 8-byte version information for the FW2 is reported through this register, as described in **Table 18**.

Table 18 FW2_VERSION Register

FW2_VERSION: 8 Bytes CCG3/CCG4 Address: 0x0020			
Field	**Field Name**	**R**	**Description**
Byte[0] to Byte[7]	FW2 Version	R	8-byte version information for the FW2 firmware application

3.1.13 FIRMWARE_BINARY_LOCATION

This register reports the flash row location of the firmware binaries in dual firmware mode and reports the configuration table and firmware location in legacy boot mode, as described in **Table 19**.

Table 19 FIRMWARE_BINARY_LOCATION Register

FIRMWARE_LOCATION: 4 bytes CCG3/CCG4 Address: 0x0028			
Field	**Field Name**	**R**	**Description**
Byte[0] to Byte[1]	FW1_START	R	In legacy boot mode, this shows the flash row where the configuration table is stored. In dual firmware mode, this shows the flash row where FW1 is stored.
Byte[2] to Byte[3]	FW2_START	R	In legacy boot mode, this shows the flash row where the firmware is stored. In dual firmware mode, this shows the flash row where FW2 is stored

3.1.14 PDPORT_ENABLE

The DEVICE_MODE register reports the number of PD ports supported by the CCG3/4 device. The PD Policy related registers on the HPI are replicated for each PD port supported by the CCG3/4 device. When the device supports multiple PD ports, the ports can be enabled/disabled using the PDPORT_ENABLE register, as described in **Table 20**.

If the value written to the PDPORT_ENABLE register is invalid (bits are set for nonexistent ports), the INVALID ARGUMENT error (0x09) will be reported. The CCG3/4 device will send the Success (0x02) response after the status of the PD ports is updated as requested. Stopping an active PD port can take a long time (~1 second) if VBus is being provided and needs to be discharged. The Success response will be returned only after this process is complete.

Commands such as DEVICE_RESET or JUMP_TO_BOOT should be initiated only after the Success response for the PDPORT_ENABLE command has been received.

The power-up default value of this register will be 0x01 for one-port PD devices and 0x03 for two-port PD devices.

Table 20 PDPORT_ENABLE Register

PDPORT_ENABLE: 1 bytes CCG3/CCG4 Address: 0x002C			
Field	**Field Name**	**R**	**Description**
Byte 0	PORT_ENABLE	RW	Bit mask where each bit specifies whether the corresponding port should be enabled. b0 – Port 0 enable b1–- Port 1 enable

3.1.15 SLEEP_CTRL

The CCG3/4 device attempts to save power by going into deep sleep mode as often as possible. The SLEEP_CTRL register is a debug option that allows you to disable sleep mode on the CCG3/4 device, as described in **Table 21**.

The power-up default value of this register is 0 (sleep enabled). The EC can write any non-zero value into this register to disable deep sleep operation.

Table 21 SLEEP_CTRL Register

SLEEP_CTRL: 1 byte CCG3/CCG4 Address: 0x002D			
Field	**Field Name**	**R**	**Description**
Byte 0	SLEEP_DISABLE	RW	Deep sleep disable value: 0 – Deep sleep is enabled. 1 – Deep sleep is disabled. Other values are reserved.

3.1.16 BATTERY_STAT

The CCG3/4 device is designed so that it offers an Rd termination when it is in reset. This allows the system based on theCCG3/4 device to receive power from a Type-C source in the dead battery condition. The CCG3/4 device's firmware uses the presence of a voltage on VBUS at startup as an indication that the device is operating under dead battery conditions and will therefore not allow any power role swaps.

The EC can use the BATTERY_STAT register to inform the CCG3/4 device that the system is not operating under dead battery conditions so that the normal role negotiations can be enabled.

This register has a default value of 0, indicating that the system is operating under dead battery conditions. The EC can write a non-zero value to update the CCG3/4 device that the battery is charged and dead battery operation can be disabled, as described in **Table 22**.

Table 22 BATTERY_STAT Register

BATTERY_STAT: 1 byte CCG3/CCG4 Address: 0x002E			
Field	**Field Name**	**R**	**Description**
Byte 0	DB_DISABLE	RW	Dead battery operation disable command: Bit 0: 0 – Dead battery operation enabled 1 – Dead battery operation disabled Other bits are reserved.

3.1.17 Vendor-Specific Registers

Some CCG3/4 device solutions may require customers to implement their own registers for communication between the EC and CCG3/4 device. A vendor-specific register region has been allocated in the HPI address space to satisfy this requirement. The vendor-specific register region covers 48 bytes of space from 0x0040 to 0x006F.

3.2 PD Registers

This section describes the use of the command, response, and status registers used in the USB PD application scenario of HPI.

Figure 10 shows a typical connection of a notebook to a monitor via a Type-C port. The notebook is battery powered, can supply 5 V at 2 A, and requires 40 W of power at 12 to 20 V to charge its battery. The monitor is plugged into the AC main supply and can supply 5 V, 1 2 V, or 20 V at 3 A. The cable mentioned in this example is an Electronically Marked Cable Assembly (EMCA). Both the notebook, and the monitor are dual role ports (DRPs), but the notebook prefers to be a downstream facing port (DFP) and the monitor, an upstream facing port (UFP).

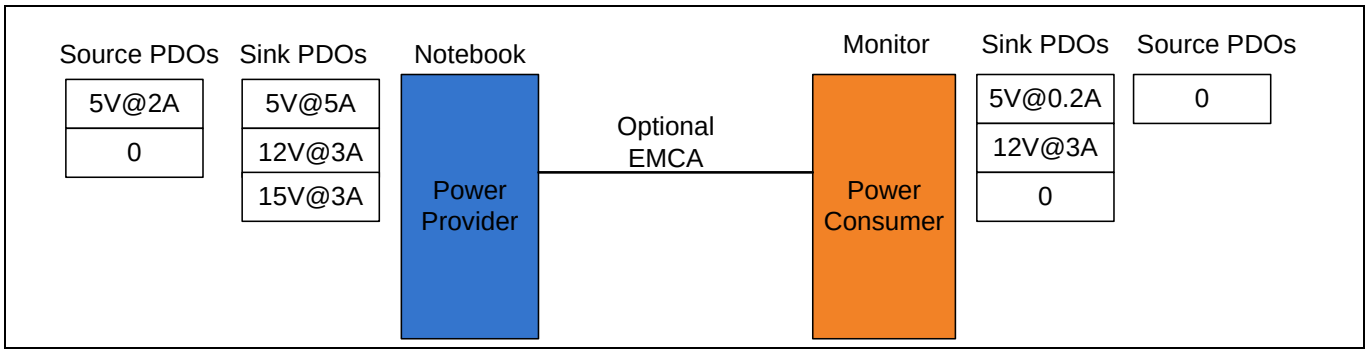

Figure 10 Notebook and Monitor Connected with USB Type-C Cable

Upon initial connection, the notebook becomes a DFP and a power source, and the monitor becomes a UFP and a power sink. The monitor, being AC plugged, can provide more power to the notebook, while the notebook, being battery powered, prefers to be the power sink to charge its battery. This condition triggers a power role swap from the monitor, which results in the monitor charging the notebook.

However, their data roles remain unchanged with the notebook still operating as the data provider. After the power role swap, the monitor supplies 20 V at 3 A to the notebook. From a USB PD perspective, a new contract has been established between the notebook and the monitor as shown in **Figure 11**.

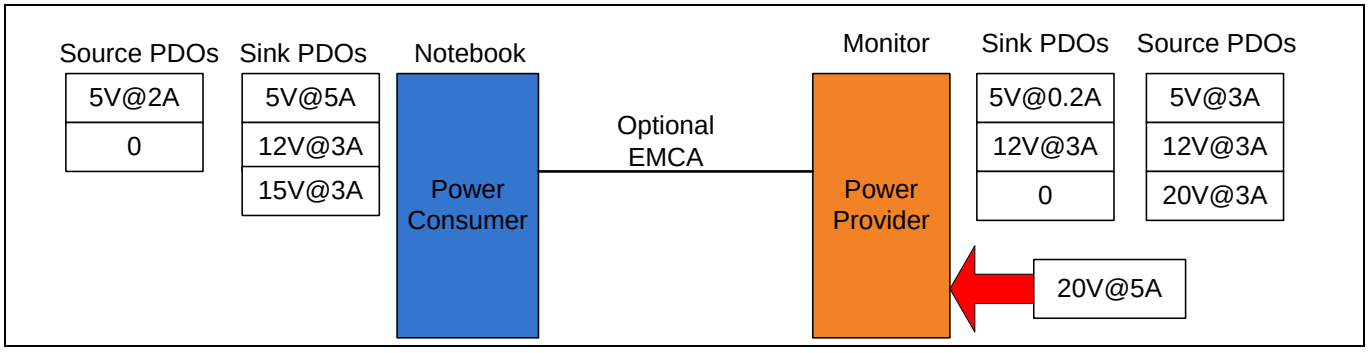

Figure 11 Power Role Swap between Notebook and Monitor

The EC then retrieves information about this PD contract over the HPI from the CCG3/4 device (updated source or sink PDOs) and updates the battery charger with this information. The status and command registers that are used for the exchange of this information are described in the following section. In the interest of clarity, this section describes the registers as well as the particular use case while establishing the PD contract.

3.2.1 Status Registers

The CCG3/4 device contains many status registers that can be read by the EC. They are updated by the CCG3/4 device during run time. The order of the registers defined here is not in numerical order; it is in a logical order for easier understanding.

3.2.1.1 TYPE_C_STATUS

This register holds the current Type-C port status information, as described in the following table.

Table 23 TYPE_C_STATUS Register

TYPE_C_STATUS: 1 Byte
CCG3/CCG4 Port-0 Address: 0x100C
CCG4 Port-1 Address: 0x200C

Field	Field Name	R/W	Description
Byte[0]	Status Bit Map	R	b0: Port partner connection status 0: Port not connected to partner 1: Port connected to partner Following fields are valid only if Type-C port is in connected status. b1: CC polarity 0: CC1 1: CC2 b2:b4: Type of device attached 000: Nothing attached 001: Sink attached 010: Source attached 011: Debug Accessory attached

TYPE_C_STATUS: 1 Byte CCG3/CCG4 Port-0 Address: 0x100C CCG4 Port-1 Address: 0x200C			
Field	**Field Name**	**R/W**	**Description**
			100: Audio Accessory attached 101: Powered Accessory attached 110: Unsupported Accessory attached b5: Ra Status 0b0: If CCG detects Ra 0b1: If CCG detects Ra Note that CCG is expected to detect Ra only when CCG is a source. When CCG is a sink, this field is not valid. b6b7: Type-C current level This is the Type-C current level in both source and sink modes. In source mode, this value reflects the value of Rp (Type-C current level) asserted by CCG3/4. In sink mode, this value reflects the value of Rp asserted by port partner (source). When CC line voltage is invalid, the current advertised field will reflect default current value. Voltage above vRd-3.0 (max) – 2.04 V will be considered invalid. 00: Default 01: 1.5 A 10: 3 A Current Port Power Role status bit in PD_STATUS register defines CCG3/4's current power role (source/sink).

3.2.1.2 PD_STATUS

This register contains the default PD profile and current PD status of the device, as described in the following table.

Table 24 PD_STATUS Register

PD_STATUS: 4 Bytes CCG3/CCG4 Port-0 Address: 0x1008 CCG4 Port-1 Address: 0x2008			
Field	**Field Name**	**R/W**	**Description**
Byte[0:3]	Status Bit Map	R	b1b0 – Default Port Data Role 00: UFP 01: DFP

PD_STATUS: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1008
CCG4 Port-1 Address: 0x2008

Field	Field Name	R/W	Description
			10: DRP b2: Default Port Data Role in case of DRP 0: UFP 1: DFP b3b4: Default Port Power Role 00: Sink 01: Source 10: Dual Role (Source/Sink or Sink/Source) b5: Default Port Power Role in case of Dual Role 0: Sink 1: Source b6: Current Port Data Role This field is updated with the current data role when Type-C port is in connected state. When Type-C port is not in connected state, this field is not valid. This field is also updated when CCG3/4 undergoes a successful DR_SWAP. 0: UFP 1: DFP b7: Reserved b8: Current Port Power Role This field is updated with the current power role when Type-C port is in connected state. When Type-C port is not in connected state, this field is not valid. This field is also updated when CCG3/4 undergoes a successful PR_SWAP. 0: Sink 1: Source b9: Reserved b10: Contract State This bit is set when CCG3/4 established a PD contract with port partner. It is cleared when no contract exists. 0: No contract exists with port partner 1: Contract exists with port partner

PD_STATUS: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1008
CCG4 Port-1 Address: 0x2008

Field	Field Name	R/W	Description
			Current Port Data Role and Current Port power role fields are valid only if contract state bit is set. b11: EMCA Present This bit is set if CCG discovers an EMCA. CCG3/4 is expected to send Discover ID command whenever it starts or changes to DFP role. If EMCA sends an ACK response to Discover ID command, this bit is set. b12: VCONN Supplier This bit is set when CCG3/4 is supplier of VCONN. CCG3/4 is expected to be supplier of VCONN in following cases: When CCG is DFP When CCG is UFP but becomes source of VCONN after a successful VCONN SWAP with port partner b13: VCONN status This bit is set if CCG3/4 is sourcing VCONN. CCG3/4 turns on VCONN in DFP mode when CCG detects Ra. VCONN is not turned on if Ra is not detected. If Ra is detected, CCG3/4 starts EMCA discovery process by sending DISCOVER ID Structured VDM command. If EMCA does not respond to DISCOVER ID command initiated by CCG3/4, VCONN is not turned off. If EMCA responds to DISCOVER ID command and CABLE VDO indicates that EMCA does not need VCONN, CCG3/4 turns Off VCONN. EC can control VCONN through PD_CONTROL register.

3.2.1.3 CURRENT_PDO

This register holds the active PDO, as described in **Table 25**. Depending on whether CCG3/4 device is a source or sink, if CCG3/4 is the source, this register holds the PDO selected by the attached sink device. If the CCG3/4 device is the sink, this register holds the PDO that CCG3/4 requested from the source device.

Table 25 CURRENT_PDO Register

CURRENT_PDO: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1010
CCG4 Port-1 Address: 0x2010

Field	Field Name	R/W	Description
Byte[0:3]	PDO	R	Active PDO. Register's contents are valid only if contract state bit in PD_STATUS register is set.

3.2.1.4 CURRENT_RDO

The RDO (Request Data Object) is a 32-bit object sent by the sink device to a source indicating the PDO it prefers. Refer to the **USB PD Specification Rev 2.0, version 1.1** or later for more details about the RDO. This register holds the active RDO, as described in **Table 26**. It will hold specific values depending on whether the CCG3/4 device is a source or a sink. If CCG3/4 is a source, this register contains the RDO sent by the attached sink device in its request message. If CCG3/4 is a sink, this register contains the RDO it requested.

Table 26 CURRENT_RDO Register

CURRENT_RDO: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1014
CCG4 Port-1 Address: 0x2014

Field	Field Name	R/W	Description
Byte[0:3]	RDO	R	Active RDO. Register's contents are valid only if contract state bit in PD_STATUS register is set.

3.2.1.5 CURRENT_CABLE_VDO

The cable VDO is a 32-bit object sent by the cable describing its properties. This register contains the cable VDO returned by the cable in response to the Discover Identity request sent by the CCG3/4 device, as described in the following table.

Table 27 Current_Cable_VDO Register

CURRENT_CABLE_VDO: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1018
CCG4 Port-1 Address: 0x2018

Field	Field Name	R/W	Description
Byte[0:3]	VDO	R	Cable VDO

3.2.1.6 EFFECTIVE_SOURCE_PDO_MASK

The EC reads this register to determine the PDOs that are being used by CCG3/4 device in source mode, as described in **Table 28**. At initialization, this register holds the default PDO mask used by the CCG3/4 device for power negotiation. This register is updated when the EC chooses a new source PDO mask at run time using the SELECT_SOURCE_PDO register.

Table 28 EFFECTIVE_SOURCE_PDO_MASK Register

EFFECTIVE_SOURCE_PDO_MASK: 1 Byte
CCG3/CCG4 Port-0 Address: 0x1002
CCG4 Port-1 Address: 0x2002

Field	Field Name	R/W	Description
Byte[0]	PDO Mask	R	b0: PDO 0 enabled b1: PDO 1 enabled b2: PDO 2 enabled b3: PDO 3 enabled b4: PDO 4 enabled b5: PDO 5 enabled b6: PDO 6 enabled b7: Externally powered bit. If the PDO is being used, the corresponding bit is 1. Otherwise, it is 0. If Externally Powered status is being advertised, bit 7 is set. Otherwise it is 0.

3.2.1.7 EFFECTIVE_SINK_PDO_MASK

The EC reads this register to determine the PDOs that are being used by CCG3/4 device in sink mode, as described in **Table 29**. At initialization, this register holds the default PDO mask used by the CCG3/4 device for power negotiation. The register is updated when the EC chooses a new sink PDO mask at run time using the SELECT_SINK_PDO register.

Table 29 EFFECTIVE_SINK_PDO_MASK Register

EFFECTIVE_SINK_PDO_MASK: 1 Byte
CCG3/CCG4 Port-0 Address: 0x1003
CCG4 Port-1 Address: 0x2003

Field	Field Name	R/W	Description
Byte[0]	PDO Mask	R	b0: PDO 0 enabled b1: PDO 1 enabled b2: PDO 2 enabled b3: PDO 3 enabled b4: PDO 4 enabled b5: PDO 5 enabled b6: PDO 6 enabled b7: Externally powered bit.

EFFECTIVE_SINK_PDO_MASK: 1 Byte CCG3/CCG4 Port-0 Address: 0x1003 CCG4 Port-1 Address: 0x2003			
Field	**Field Name**	**R/W**	**Description**
			If the PDO is being used, the corresponding bit is 1. Otherwise, it is 0. If Externally Powered status is being advertised, bit 7 is set. Otherwise it is 0.

3.2.1.8 PORT_INTR_STATUS

This 4-byte register reports the PD port status in the form of interrupt status bits, so that EC can identify all events in all cases, including event queue overflow, as described in **Table 30**. This register is a write one to clear (W1C) register, where the EC has to write 1 into any bits that it wants to clear.

Note: There is no direct relationship between this register and the INTR# triggered by the CCG3/4 device. The EC_INT operation is based on the event queue and RESPONSE register access.

Table 30 PORT_INTR_STATUS Register

PORT_INTR_STATUS: 4 Bytes CCG3/CCG4 Port-0 Address: 0x1034 CCG4 Port-1 Address: 0x2034			
Field	**Field Name**	**R/W**	**Description**
Bit 0	CC_ATTACH	R / W1C	Type-C attach event notification
Bit 1	CC_DETACH	R / W1C	Type-C detach event notification
Bit 2	CONTRACT_CMPLT	R / W1C	PD contract complete
Bit 3	PRSWAP_CMPLT	R / W1C	PR swap complete
Bit 4	DRSWAP_CMPLT	R / W1C	DR swap complete
Bit 5	VCONNSWAP_CMPLT	R / W1C	VCONN swap complete
Bit 6	HARDRESET_RCVD	R / W1C	HARD RESET received
Bit 7	HARDRESET_SENT	R / W1C	HARD RESET sent
Bit 8	SOFTRESET_SENT	R / W1C	SOFT RESET sent
Bit 9	CABLERESET_SENT	R / W1C	CABLE RESET sent
Bit 10	CC_ERROR_RCVRY	R / W1C	Type-C error recovery initiated
Bit 11	SRC_DISABLED	R / W1C	CCG3/4 entered Source Disabled state.
Bit 12	EMCA_DETECT	R / W1C	EMCA detected by CCG3/4
Bit 13	CBLDISC_FAIL	R / W1C	Cable discovery by CCG3/4 failed.
Bit 16	ALTMODE_ENTRY	R / W1C	Entered any alternate mode
Bit 17	ALTMODE_EXIT	R / W1C	Exited any alternate mode
Bit 29	UNEXP_VOLTAGE	R / W1C	Unexpected voltage detected
Bit 30	OVP_EVT	R / W1C	VBUS overvoltage
Bit 31	OCP_EVT	R / W1C	VBUS overcurrent

3.2.2 Control Registers

3.2.2.1 SWAP_RESPONSE

This register is used by the EC to determine how the CCG3/4 device responds to SWAP commands, as described in **Table 31**. The CCG3/4 device initializes this register with the Swap Response parameter (in the configuration table) after power up. The EC can update this register based on operational conditions. It can also read this register to determine the selected responses for each SWAP command.

Table 31 SWAP_RESPONSE Register

SWAP_RESPONSE: 1 Byte **CCG3/CCG4 Address: 0x1028** **CCG4 Port-1 Address: 0x2028**			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Response	R/W	Two bits are reserved for each SWAP command: b0:b1: DR_SWAP b2:b3: PR_SWAP b4:b5: VCONN_SWAP The value of these bit fields determines the behavior of CCG3/4 when a SWAP command is sent by port partner: 0b00: Accept: Send ACCEPT 0b01: Reject: Send REJECT 0b10: Wait: Send WAIT 0b11: Illegal value. CCG3/4 will send REJECT.

3.2.2.2 SELECT_SOURCE_PDO

This register is used by the EC to select the desired source PDO to be used during a PD contract negotiation, as described in **Table 32**. By default, the CCG3/4 device uses the default source PDO mask in the configuration table to choose PDOs for power contract negotiations. The configuration table contains a list of up to seven source PDOs supported by the CCG3/4 device. The default PDO mask (in the configuration table) determines the PDOs, which are used by the CCG3/4 device to negotiate the power contract by default. After power up, CCG3/4 updates the EFFECTIVE_SOURCE_PDO_MASK register with this default source PDO mask. The EC can select PDOs for negotiating power at run time by setting the corresponding bits of the PDOs in this register.

Note that the EC has to ensure the following conditions while updating this register:

- Voltage level of all selected PDOs is different.
- One 5-V PDO is always selected.
- EC selects at least one PDO from the default Source PDO list while updating this register. Otherwise, the CCG3/4 device returns INVALID_ARGUMENT as a response and does not handle the request.

The MSB of the PDO mask is used by the EC to indicate if the Externally Powered status needs to be advertised in the PDOs.

Table 32 SELECT_SOURCE_PDO Register

SELECT_SOURCE_PDO: 1 Byte CCG3/CCG4 Port-0 Address: 0x1004 CCG4 Port-1 Address: 0x2004			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Command	R/W	b0: Select PDO 0. b1: Select PDO 1. b2: Select PDO 2. b3: Select PDO 3. b4: Select PDO 4. b5: Select PDO 5. b6: Select PDO 6. b7: Externally powered bit A read of this field always returns 0.

Following are the conditions under which the EC updates the source PDO mask. The CCG3/4 device's response for each condition is also given.

- **Type-C Port is not connected or PD contract does not exist:** CCG3/4 updates the EFFECTIVE_SOURCE_PDO_MASK register with the PDO mask written by the EC and returns a Success response message to the EC. The CCG3/4 device uses the updated PDO mask after a connection is established on the Type-C Port.
- **CCG3/4 is currently operating in sink mode:** The CCG3/4 device updates the EFFECTIVE_SOURCE_PDO_MASK register with the PDO mask, returns Success response code, and uses it when it enters source (provider) mode.
- **CCG3/4 is operating in source mode, and PDO mask sent by EC is same as the EFFECTIVE_SOURCE_PDO_MASK:** The CCG3/4 device returns Success response code to the EC and does not initiate power contract renegotiation with the port partner.
- **CCG3/4 is operating in source mode, and the PDO mask sent by EC differs from EFFECTIVE_SOURCE_PDO_MASK:** The CCG3/4 device returns PD Command Failed response code if the CCG3/4 device is actively receiving or transmitting a PD message on the Type-C interface (Type-C interface is not "idle").

CCG3/4 returns Success response code if the Type-C interface is idle. CCG3/4 initiates a power contract renegotiation process with the port partner using updated PDOs. This process is explained in the next section.

3.2.2.3 Source Mode Power Renegotiation Flow

Figure 12 depicts the power contract renegotiation flow once the EC successfully updates the SOURCE PDO mask. It is assumed that CCG3/4 is operating in source mode and has previously entered into a power contract with the port partner. It is also assumed that the EC has unmasked relevant HPI events using the EVENT_MASK register.

1. EC updates the SOURCE_PDO_MASK register. CCG3/4 updates the EFFECTIVE_SOURCE_PDO_MASK register and sends a Success response to the EC.
2. The CCG3/4 device transmits a Source Capability message based on the current PDO mask to the port partner. It retries for nCapsCount (50) until GOOD_CRC is received. If GOOD_CRC is not received, CCG3/4 moves to the Source Disabled state and notifies the EC through the Source Disabled State Entered event.

3. If GOOD_CRC is received in response to the SRC CAP (Source Capability) message, the CCG3/4 device waits for the RDO from the port partner. If the port partner does not send the RDO in the appropriate time, the CCG3/4 device notifies the EC with a Sender Response Timer Timeout event and then sends a hard reset to the port partner.
4. If a valid RDO is not received, the CCG3/4 device sends a REJECT followed by a hard reset if an explicit contract exists but the previous contact is invalid
   - The CCG3/4 device sends a REJECT if an explicit contract exists and the previous contract is still valid
   - The CCG3/4 device returns this information to the EC through a PD Contract Negotiation Complete event
5. If a valid RDO is received, CCG3/4 sends an ACCEPT message to the port partner. It adjusts the power supply to the negotiated output and sends a PS_RDY. Subsequently, the CCG3/4 device notifies the EC with a PD Contract Established event. If the port partner indicates a Capability Mismatch in the RDO, the CCG3/4 device notifies the EC through a PD Contract Negotiation Complete event. In such cases, the EC is expected to retrieve the port partner's sink capabilities and adjust the CCG3/4 device's source capabilities accordingly, if required.

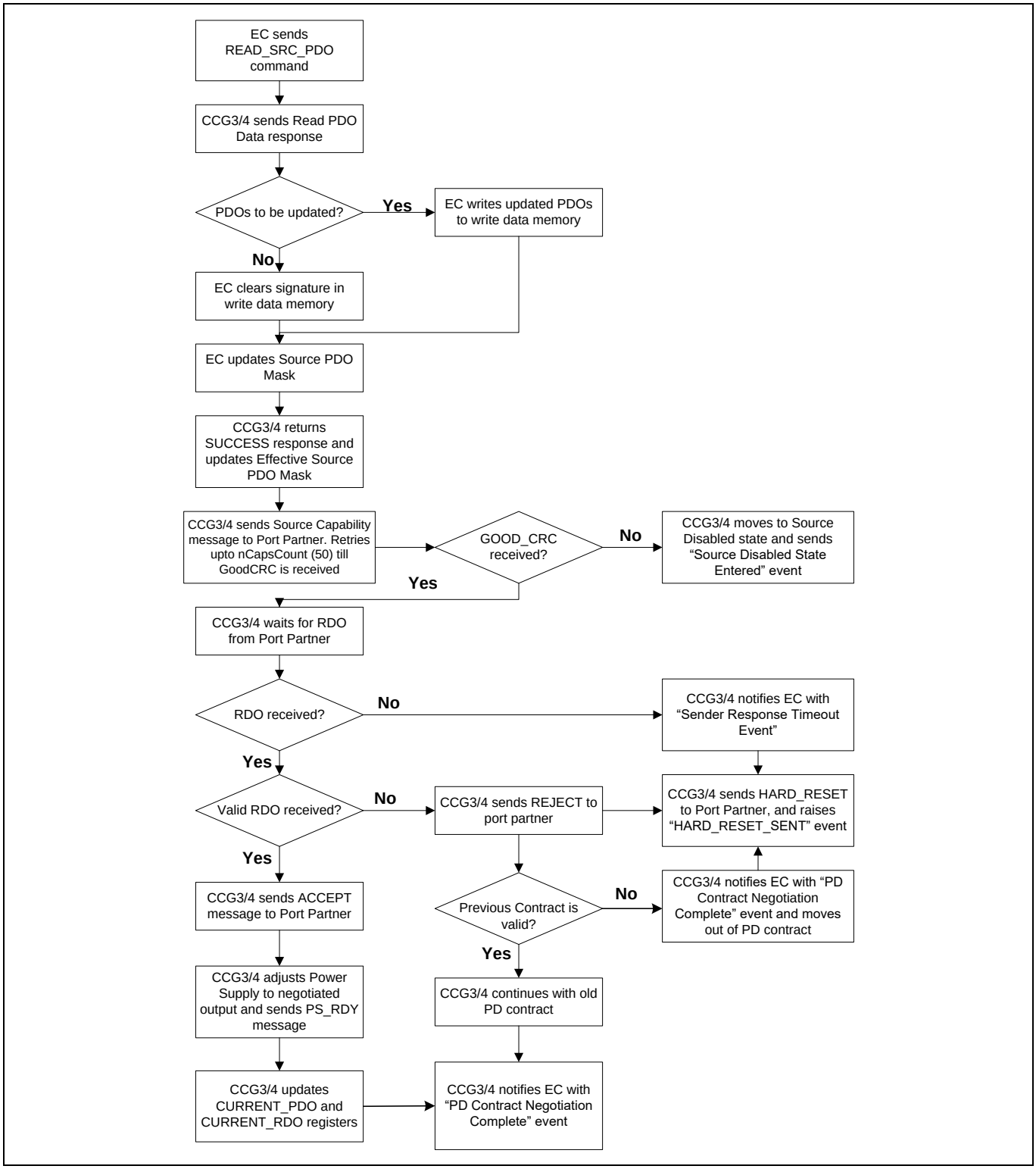

Figure 12 Source Mode Power Contract Renegotiation Process

3.2.2.4 SELECT_SINK_PDO

This register is used by the EC to select sink mode PDOs at run time, as described in **Table 33.** By default, the CCG3/4 device uses the Default Sink PDO mask present in the configuration table to choose sink PDOs. The configuration table contains a list of sink PDOs (maximum s PDOs) to be used in sink mode. The default sink PDO mask determines which PDOs are used by CCG3/4 to determine its sink capabilities. After power-up or coming out of reset, the CCG3/4 device updates the EFFECTIVE_SINK_PDO_MASK register with the default sink PDO mask value. The EC can then select sink PDOs at run time if the sink capabilities change by setting the corresponding bits of PDOs required in this register.

Note that the EC has to ensure the following conditions while updating this register:

- Voltage level of all selected PDOs is different.
- There is exactly one 5-V PDO.
- EC selects at least one PDO from the default Sink PDO list while updating this register. Otherwise, the CCG3/4 device returns INVALID_ARGUMENT as a response and does not handle the request.

The MSB of the PDO mask is used by the EC to indicate if the Externally Powered status needs to be advertised in the PDOs.

Table 33 SELECT_SINK_PDO Register

SELECT_SINK_PDO: 1 Byte
CCG3/CCG4 Port-0 Address: 0x1005
CCG4 Port-1 Address: 0x2005

Field	Field Name	R/W	Description
Byte[0]	Command	R/W	b0: Select PDO 0. b1: Select PDO 1. b2: Select PDO 2. b3: Select PDO 3. b4: Select PDO 4. b5: Select PDO 5. b6: Select PDO 6. b7: Externally powered bit A read of this field always returns 0.

The EC updates the sink PDO mask when:

- **Type-C port is not connected or PD contract does not exist.**

  CCG3/4 updates the EFFECTIVE_SINK_PDO_MASK register with the PDO mask written by the EC and returns Success response code to the EC. CCG3/4 uses the updated PDO mask to determine its sink capabilities when source capabilities are received from the source port.

- **CCG3/4 device is operating in source (provider) mode.**

  The CCG3/4 device updates the EFFECTIVE_SINK_PDO_MASK register with the PDO mask, returns Success response code, and uses it when it enters sink (consumer) mode.

- **CCG3/4 is operating in sink mode, and PDO mask sent by EC is same as EFFECTIVE_SINK_PDO_MASK.**

CCG3/4 returns Success response code to the EC and does not initiate power contract renegotiation with the port partner.

- **CCG3/4 is operating in sink mode, and PDO mask sent by EC differs from EFFECTIVE_SINK_PDO_MASK.**

  CCG3/4 returns PD Command Failed response code if CCG3/4 is actively receiving or transmitting a PD message on the Type-C interface (Type-C interface is not idle).

CCG3/4 returns Success response code if the Type-C interface is idle. CCG3/4 initiates a power contract renegotiation process with the port partner using updated PDOs. This process is explained in the next section.

3.2.2.5 Sink Mode Power Renegotiation Flow

The following sequence, shown in **Figure 13**, describes the power contract renegotiation flow once the EC successfully updates the sink PDO mask. It is assumed that CCG3/4 is operating in sink mode and has entered into a power contract with the port partner. It is also assumed that the EC has unmasked relevant HPI events using the EVENT_MASK register.

1. The EC updates the Sink PDO Mask register. CCG3/4 updates the EFFECTIVE_SINK_PDO_MASK register and sends a Success response to the EC.
2. CCG3/4 sends a GET_SOURCE_CAP to the source to determine the latest capabilities that the source has to offer.
3. After source capabilities are received, CCG3/4 evaluates the capabilities and compares them with the current sink capabilities, as determined by the PDO mask. Based on this comparison, CCG3/4 makes a request (sends an RDO) from the offered source capabilities. If the offered source capabilities are not able to meet the CCG3/4 sink requirements, CCG3/4 sets a “capability mismatch” bit in the RDO.
4. If the Accept message is received from the source, CCG3/4 notifies the EC with an Accept Message Received event. When the source sends a PS_RDY message, CCG3/4 notifies the EC with a PS_RDY message received event and transitions the power supply to the new power level. When power transition is complete, CCG3/4 notifies the EC with a PD Contract Negotiation Complete event.
5. If a Reject message is received, CCG3/4 notifies the EC with a Reject Message Received and PD Contract Negotiation Complete event. The EC should readjust the sink PDO mask, if required.
6. If a Wait message is received, CCG3/4 notifies the EC with a Wait Message Received event and starts the Sink Request Timer. CCG3/4 resends the RDO to the source after the Sink Request Timer times out. This timer is stopped if any other PD message or hard reset is received.
7. If no response is received (Accept/Reject/Wait) for the RDO, CCG3/4 notifies the EC with a Sender Response Timer Timeout event and sends a HARD_RESET to the port partner. CCG3/4 notifies the EC with a Hard Reset Sent event.

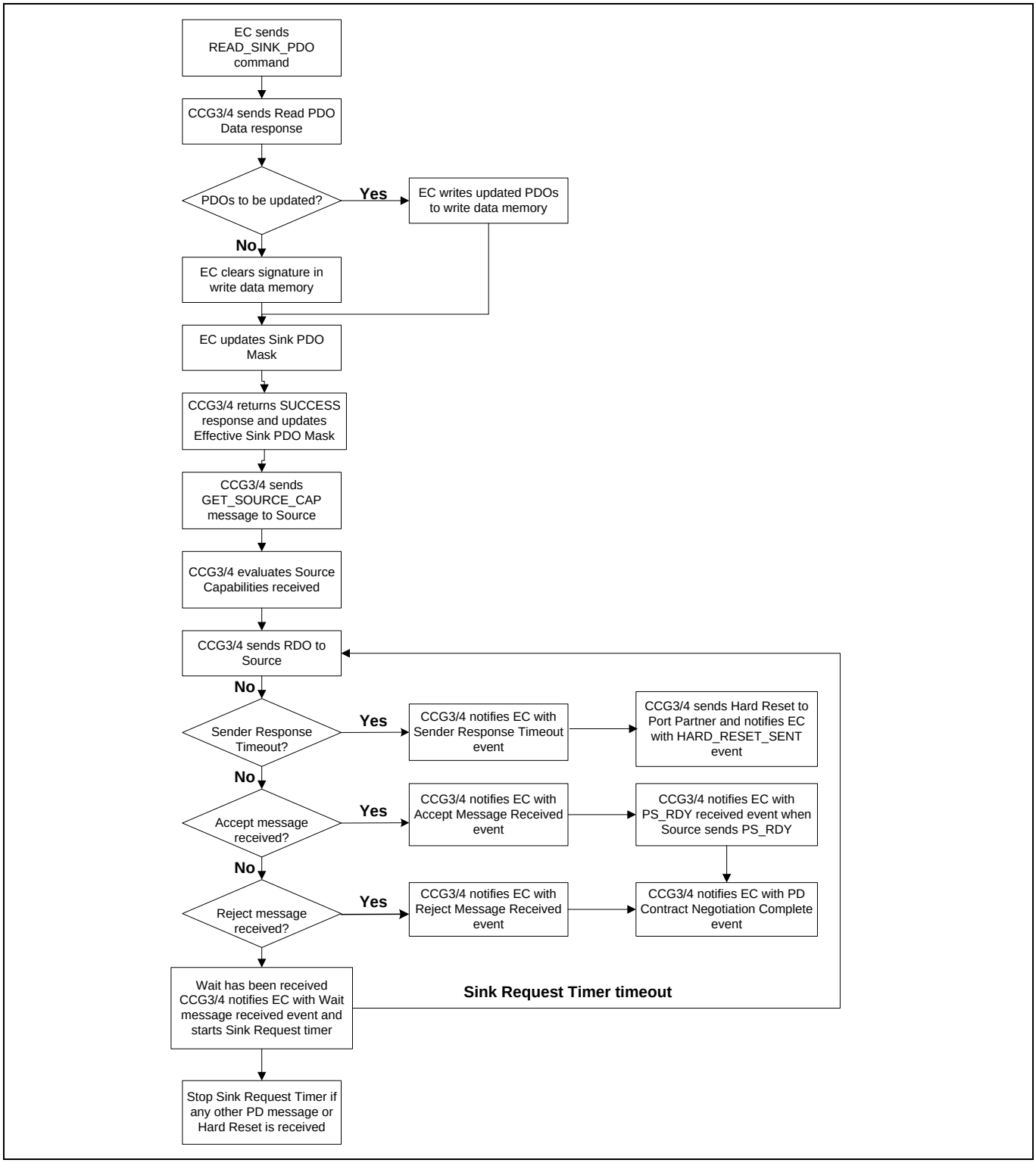

Figure 13 Sink Mode Power Contract Renegotiation Process

3.2.2.6 PD_CONTROL

This register is mainly used by the EC to request the CCG3/4 device to send PD-related commands to the port partner, as described in **Table 34**. It is also used to set up the Type-C current advertisement (Rp) and initialize and disable the Type-C port. The following sections describe possible values that the register can contain.

Table 34 PD_CONTROL Register

PD_CONTROL: 1 Byte
CCG3/CCG4 Port-0 Address: 0x1006
CCG4 Port-1 Address: 0x2006

Field	Field Name	R/W	Description
Byte[0]	Command	R/W	0x00: Set Type-C default profile 0x01: Set Type-C 1.5A profile 0x02: Set Type-C 3A profile These commands update Rp advertised by CCG. 0x03: Reserved 0x04: Reserved 0x05: Trigger Data Role Swap 0x06: Trigger Power Role Swap 0x07: Switch On VCONN EC can use this command to turn on VCONN. If Type-C port is not connected, CCG responds with PD Command Failed response code. Otherwise, CCG turns on VCONN and responds with Success response code. 0x08: Switch Off VCONN EC can use this command to turn off VCONN. CCG turns off VCONN and responds with Success response code. 0x09: Trigger VCONN Role Swap 0x0A: Retrieve Source Capabilities 0x0B: Retrieve Sink Capabilities 0x0C: Send GotoMin Message 0x0D: Send HARD RESET 0x0E: Send SOFT RESET 0x0F: Send CABLE RESET 0x10: EC Initialization complete

PD_CONTROL: 1 Byte
CCG3/CCG4 Port-0 Address: 0x1006
CCG4 Port-1 Address: 0x2006

Field	Field Name	R/W	Description
			EC can use this command to indicate to CCG3/4 that it has configured PDO mask and event mask once CCG3/4 enters normal operation mode. CCG3/4 enables Type-C interface once EC sends this command. After CCG3/4 boots up in normal operation mode, which is notified through a Reset Complete event, CCG3/4 starts a timer of 100 ms. This time period is provided to EC to configure PDO mask and event mask and then send this command. If EC does not send this command within 100 ms, CCG3/4 initializes Type-C machine and configures itself for PD negotiations once Type-C connect is detected. If EC sends this command after timeout period of 100 ms, CCG3/4 responds with PD Command Failed response code. 0x11: Port Disable EC can use this command to disable PD and Type-C interface of CCG3/4 to recover from error scenarios. CCG3/4 moves to lower power mode (deep sleep) and disables Type-C and PD interface. CCG3/4 responds with Success response code after handling this command. EC should send Reset command to CCG3/4 to re-enable Type-C and PD interface. On CCG3/4, port can be re-enabled using the PDPORT_ENABLE register. 0x12: Send Soft Reset SOP_PRIME 0x13: Send Soft Reset SOP_DPRIME 0x14: Change PD Port Parameters 0x15: Abort Pending PD Command 0x20: READ_SRC_PDO 0x21: READ_SINK_PDO A read of this field always returns 0.

3.2.2.7 EVENT_MASK

This register is used by the EC to choose the events notified by the CCG3/4 device, as described in **Table 35**. By default, CCG3/4 handles all events and asynchronous PD messages autonomously and does not notify the EC. Depending on the mask value in this register, the EC can choose which events and messages will be notified. The corresponding bit in the event mask bitmap of this register should be set to enable notification of events and messages.

Table 35 EVENT_MASK Register

EVENT_MASK: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1024
CCG4 Port-1 Address: 0x2024

Field	Field Name	R/W	Description
Byte[0:3]	Response Filter	R/W	b0: Reserved. b1: Overcurrent Detected b2: Overcoltage Detected b3: Type-C Port Connect Detected b4: Type-C Port Disconnect Detected b5: PD Contract Negotiation Complete b6: PD Control Message Received This event mask bit corresponds to following events: SWAP Complete PS_RDY Message Received GotoMin Message Received Accept Message Received Reject Message Received Wait Message Received Hard Reset Received b7: VDM Received b8: Source Capability Message Received b9: Sink Capability Message Received b10: DP Alternate Mode Related Events DP Mode Entered DP Device Connected at UFP_U DP Device Not Connected at UFP_U DP SID Not Discovered Multiple SVIDs Discovered along with DP SID DP Mode Not Supported by Cable DP Mode Not Supported by UFP b11: Error and Timeout Related Events Hard Reset Sent Soft Reset Sent Cable Reset Sent

EVENT_MASK: 4 Bytes CCG3/CCG4 Port-0 Address: 0x1024 CCG4 Port-1 Address: 0x2024			
Field	**Field Name**	**R/W**	**Description**
			Source Disabled State Entered Sender Response Timer timeout No VDM Response Received Unexpected Voltage on VBUS Type-C Error Recovery b12: EMCA Related Events EMCA Detected b13: Miscellaneous Events Rp Change Detected

3.2.2.8 CMD_Timeout

The CCG3/4 device is capable of holding a vendor defined mode (VDM) or a PD command such as HARD_RESET, DR_SWAP, and so on from the EC and retrying it if the USB PD PHY was busy when the command was queued.

The default behavior is not to queue the requests and return a PORT BUSY response immediately. If a non-zero timeout value is programmed into this register, CCG3/4 will store the command and retry it until the programmed timeout period (in ms) has elapsed, as described in **Table 36**.

Table 36 CMD_TIMEOUT Register

CMD_TIMEOUT: 1 Byte CCG3/CCG4 Port-0 Address: 0x1030 CCG4 Port-1 Address: 0x2030			
Field	**Field Name**	**R/W**	**Description**
Byte 0	Timeout	R/W	Timeout period for VDMs and PD commands in milliseconds

3.2.3 Events and Responses

3.2.3.1 RESPONSE_Register

The RESPONSE_REGISTER is used by the CCG3/4 device to indicate the need for service (as signaled by the INTR# pin). The CCG3/4 device can indicate that a response to a command has been generated, or it can indicate that an enabled event (as indicated in the EVENT_MASK register) has been received. The CCG3/4 device does not report any event except a Reset Complete (0x80) event at power up. All other events must be enabled by writing the EVENT_MASK register. **Table 37** lists the current response codes.

Table 37 RESPONSE_REGISTER

RESPONSE_REGISTER: 2 Bytes CCG3/CCG4 Address: 0x007E			
Field	**Field Name**	**R/W***	**Description**
Byte[0]	Response/ Message Code	R	b7 – Type of Message 0 – Response to command issued by EC 1 – Event or Asynchronous Message b6:b0 – Message Code The codes for responses, events, and asynchronous PD messages are listed in **Table 38**.
Byte[1]	Length	R	Length of response/message.

* The R/W column specifies the EC's register access permissions.

3.2.3.2 CCG3/4 Response and Event Codes

This table describes the CCG3/4 response and event codes.

Table 38 CCG3/4 Response and Event Codes

Code	Description
0x00	No Response. No pending command, event, or asynchronous message.
0x01	Reserved.
0x02	Success. Command handled successfully.
0x03	Flash Data Available. Flash row data requested by EC is available in data memory.
0x04	Reserved
0x05	Invalid Command. Partial, unaligned register write or invalid command.
0x06	Reserved
0x07	Flash Update Failed. Flash write operation failed.
0x08	Invalid Firmware. Firmware validity check failed. Refer to the VALIDATE_FW command.
0x09	Invalid Arguments. Command handling failed due to invalid arguments.
0x0A	Not Supported. Command not supported in the current mode.
0x0B	Reserved
0x0C	Transaction Failed. GOOD_CRC was not received for the PD message transmitted.
0x0D	PD Command Failed. CCG3/4 was not able to handle the command issued.
0x0F	Undefined Error
0x10-0x7F	Reserved
Device-Specific Events	
0x80	Reset Complete. Device was reset and is back in operation mode.
0x81	Message Queue Overflow. Message queue overflow detected.

Code	Description
Type-C Specific Events	
0x82	Overcurrent Detected
0x83	Overvoltage Detected
0x84	Type-C Port Connect Detected
0x85	Type-C Port Disconnect Detected
PD Control Message Specific Events	
0x86	PD Contract Negotiation Complete
0x87	SWAP Complete
0x88	Reserved
0x89	Reserved
0x8A	PS_RDY Message Received
0x8B	GotoMin Message Received.
0x8C	Accept Message Received
0x8D	Reject Message Received
0x8E	Wait Message Received
0x8F	Hard Reset Received
PD Data Message Specific Events	
0x90	VDM Received
Capability Message Specific Events	
0x91	Source Capabilities Message Received
0x92	Sink Capabilities Message Received
DP and Alternate Mode Specific Events	
0x93	Display Port Alternate Mode entered
0x94	Display Port device connected at UFP_U
0x95	Display port device not connected at UFP_U
0x96	Display port SID not found in Discover SID process
0x97	Multiple SVIDs discovered along with DisplayPort SID
0x98	DP Functionality not supported by Cable
0x99	Display Port Configuration not supported by UFP
Resets and Error Scenario Events	
0x9A	Hard Reset Sent to Port Partner
0x9B	Soft Reset Sent to Port Partner
0x9C	Cable Reset Sent to EMCA
0x9D	Source Disabled State Entered
0x9E	Sender Response Timer Timeout
0x9F	No VDM Response Received
0xA0	Unexpected Voltage on Vbus
0xA1	Type-C Error Recovery
0xA2-A5	Reserved

Code	Description
0xA6	EMCA Detected
0xA7-0XA9	Reserved
0xAA	Rp Change Detected

Transaction Failed

If a GoodCRC message was not received for a USB PD command sent by the CCG3/4 device, it retries the command as described in the USB PD Specification. If the retry counter runs down to 0 (that is, port partner never responds), the EC is notified with this status code.

This status code means that the port partner is unresponsive. The EC is therefore required to retry the previous request after some time. If this retry also fails, the EC can take any recovery action.

PD Command Failed

The CCG3/4 device responds with this response code under the following conditions:

- If the EC requests to transmit a PD control/data message when the Type-C port is not connected or if a PD contract does not exist
- If the EC issues a PD Policy command (for example, a Data Role Swap) when the CCG3/4 device is actively communicating on the Type-C interface
- If CCG3/4 is not in the PE SOURCE/SINK READY states as defined in Chapter 8, Section 3, of the USB PD Specification. CCG3/4 can accept a PD command from the EC only in the SOURCE/SINK READY states. Refer to the USB PD Specification Rev. 2.0 Ver. 1.1 Section 8.3.3, “State Diagrams” for details on the PE states of a USB PD port.

This applies to all PD messages sent by the CCG3/4 device autonomously and messages triggered by the EC. When the EC triggers a PD message, the CCG3/4 device transitions to an appropriate state and executes the requirements of that state. The EC is expected to wait until CCG3/4 transitions back to the Ready state and then trigger a new message. The CCG3/4 device provides all relevant events to the EC while transitioning between states so that the EC knows when CCG3/4 has transitioned to the Ready state.

PD Contract Negotiation Complete

This event contains information relevant to the last negotiated PD contract.

The CCG3/4 device uses the source and sink power capabilities programmed in its configuration table and selected by the EC using the SELECT_SOURCE_PDO and SELECT_SINK_PDO registers to negotiate the PD contract with the port partner after a Type-C connect is detected.

The CONTRACT_INFO status bitmap is written into the read data memory once the PD contract negotiation event is complete. **Table 39** describes the CONTRACT_INFO status bitmap.

Table 39 CONTRACT_INFO Bitmap

Field	Field Name	Description
Byte[0]	CONTRACT_STATUS	b0: Set if PD contract negotiation is successful. b1: Set if port partner indicates Capability Mismatch. b0 is set if this bit is set. b4:b2: Reason for PD contract negotiation failure. EC can refer to this bit field if b0 is not set. It can take the following values:

Field	Field Name	Description
		0b000: CCG3/4 (source) rejects RDO, and previous PD contract is valid. Received RDO is present in RDO field. 0b001: CCG3/4 (source) rejects RDO, and previous PD contract is not valid. Received RDO is present in RDO field. CCG3/4 comes out of PD contract by sending Hard Reset to port partner. 0b010: CCG3/4 (source) rejects RDO, and there was no previous explicit PD contract. Received RDO is present in RDO field. 0b011: Port partner rejects CCG3/4's (sink) RDO, and there was a previous explicit PD contract. 0b100: Port partner rejects CCG3/4's (sink) RDO, and there was no previous explicit PD contract. 0b101: Port partner accepts CCG3/4's (sink) RDO but fails to send PS_RDY. CCG3/4 moves out of PD contract and sends HARD_RESET to port partner. 0b110: CCG3/4 in source mode fails to send PS_RDY because voltage does not reach the expected negotiated level. CCG3/4 goes through Type-C error recovery. All other values are reserved. b7:b5: Reserved
Byte[2:3]	Reserved	–
Byte[4:7]	RDO	When PD contract negotiation fails because CCG3/4 in source mode rejects port partner's RDO, this field contains the received RDO.

Note that when the RDO field is not applicable, the size of CONTRACT_INFO is 1 byte, and only the CONTRACT_STATUS byte is returned by the CCG3/4 device. If RDO is applicable, the complete 8 bytes are returned. The CONTRACT_INFO status is placed in the read buffer (offset: 0x80).

SWAP Complete

This event contains information related to the DR_SWAP, PR_SWAP, and VCONN_SWAP commands. The EC can trigger a SWAP command through the PD_CONTROL register .The response received from the port partner is notified through this event.

If the CCG3/4 device receives a SWAP command from the port partner, it responds based on SWAP_RESPONSE and provides SWAP-related information through this event.

The SWAP_STATUS bitmap is written into the read data memory, once the Swap Complete event is complete. **Table 40** shows the SWAP_STATUS bitmap.

Table 40 SWAP_STATUS Bitmap

Field	Field Name	Description
Byte[0]	SWAP_ STATUS	b3:b0: Type of SWAP 0b0000: DR_SWAP 0b0001: PR_SWAP 0b0010: VCONN_SWAP b7:b4: Response to SWAP command 0b000: ACCEPT: Command accepted and SWAP complete

Field	Field Name	Description
		0b001: REJECT: SWAP command rejected 0b010: WAIT: SWAP command should be retried later. 0b011: NO RESPONSE: SWAP command ignored 0b100: HARD_RESET sent Note that a response is generated either by CCG3/4 or by ort partner based on initiator of command. If EC triggers a SWAP command, response contains response sent by port partner, as described in section **3.2.2.1**. If port partner is the initiator of SWAP command, swap response contains the response sent by CCG3/4 based on SWAP_RESPONSE register, as described in section **3.2.2.1**.

If the port partner is the initiator of a SWAP command, CCG3/4 is expected to respond based on the SWAP_RESPONSE register. CCG3/4 will override the responses in the SWAP_RESPONSE register in the following scenarios:

- In DFP/UFP-only solutions such as a controller power adapter or a display dongle/adapter, the CCG3/4 device may respond with REJECT to the DR_SWAP and PR_SWAP commands. It then notifies the EC with the SWAP_COMPLETE event, and the SWAP response will be REJECT.
- If the Type-C port supports alternate modes (such as DisplayPort) and if a Data Role Swap command is received from the port partner, the CCG3/4 device will respond with a HARD_RESET if the SWAP_RESPONSE is configured as NO_RESPONSE. This is because Type-C ports are not expected to send DR_SWAP when alternate modes are active. The CCG3/4 device notifies the EC with the SWAP_COMPLETE event, and the SWAP response is HARD_RESET.

Source Capabilities Message Received

CCG3/4 device raises this event when a source capabilities message is received from the port partner. The CCG3/4 device writes the received source capabilities message into the read data memory. It updates the length field of the RESPONSE register with the length of the source capabilities message in bytes and asserts the INTR# pin. The format of the source capabilities message in the read data memory is as follows:

- First two bytes is the message header.
- Next two bytes is SOP type: SOP (0). (Source capability message is always sent with SOP.)
- 32-bit source PDOs received from port partner

Sink Capabilities Message Received

The CCG3/4 device raises this event when a sink capabilities message is received from the port partner. The CCG3/4 device writes the sink capabilities message into the read data memory. It updates the length field of the RESPONSE register with the length of the sink capabilities message in bytes and asserts the INTR# pin. The format of the sink capabilities message in the read data memory is as follows:

- First two bytes is the message header.
- Next two bytes is SOP type: SOP (0). (Sink capability message is always sent with SOP.)
- Sink PDOs received from port partner

Unexpected Voltage on VBUS

The CCG3/4 device notifies the EC with this event if CCG3/4 is a source and an unexpected voltage is detected on VBUS before CCG3/4 turns on VBUS. CCG3/4 does not continue with Type-C connect tasks in this case and does not start PD tasks. CCG3/4 stays in this state until a Type-C disconnect happens or the VBUS voltage goes to a safe level. The EC can choose to disable the CCG3/4 Type-C interface using the PORT_DISABLE command in the PD_CONTROL register.

Type-C Error Recovery

The CCG3/4 device notifies the EC with this event when it enters the Type-C Error Recovery state. Refer to section 4.5.2.2.2 of the Type-C Specification v1.1 to read more about it.

EMCA Detected

CCG3/4 in DFP mode discovers EMCAs (SOP') by sending the Discover Identity Structured VDM command. If the EMCA responds with an ACK, the CCG3/4 device notifies the EC with this event.

3.2.4 Summary

Table 41 gives a summary of the PD policy/status registers.

Table 41 Summary of PD Policy/Status Registers

Register	Address*	Size	TYPE
EFFECTIVE_SOURCE_PDO_MASK	0x24 0x1002 0x2002	1	Status: Current source PDO mask
EFFECTIVE_SINK_PDO_MASK	0x25 0x1003 0x2003	1	Status: Current sink PDO mask
SELECT_SOURCE_PDO	0x26 0x1004 0x2004	1	Control: Mask to select source PDOs
SELECT_SINK_PDO	0x27 0x1005 0x2005	1	Control: Mask to select sink PDOs
PD_CONTROL	0x28 0x1006 0x2006	1	Control: Trigger PD control messages
PD_STATUS	0x2C 0x1008 0x2008	4	PD port status
TYPE_C_STATUS	0x30 0x100C 0x200C	1	Type-C interface status.
CURRENT_PDO	0x34 0x1010 0x2010	4	Status: Currently active PDO

Register	Address*	Size	TYPE
CURRENT_RDO	0x38 0x1014 0x2014	4	Status: RDO used to establish PD contract
CURRENT_CABLE_VDO	0x3C 0x1018 0x2018	4	Status: Currently active cable VDO
EVENT_MASK	0x48 0x1024 0x2024	4	Control: Select events to be reported
SWAP_RESPONSE	0x4C 0x1028 0x2028	1	Control: Select response to SWAP requests
CMD_TIMEOUT	N/A 0x1030 0x2030	1	Control: VDM and PD command timeout setting
PORT_INTR_STATUS	N/A 0x1034 0x2034	4	Status: Port status notifications
EXTERNAL_POWER_CONTROL	0x60 N/A N/A	1	Status: Power control request from CCG to EC
PD_RESPONSE	0x7E 0x1400 0x2400	2 4	Status: Response type and length

*First row in Address column denotes CCG1/2 address. Second row in address denotes CCG4 Port1 address. Third row in address denotes CCG4 Port2 address.

3.3 VDM Registers

The EC needs to use the registers described in this section to send unstructured and structured VDMs to the port partner.

3.3.1 VDM_CONTROL

This register is used by the EC to request the CCG3/4 device to send one VDM, as described in **Table 42**. The EC subsequently updates the write memory with the contents of the VDM (with each 32-bit object in little-endian format) and then updates this register. The CCG3/4 device then tunnels this VDM to the port partner. If the VDM is transmitted successfully (GoodCRC received from port partner), the CCG3/4 device updates the RESPONSE register with Success code and asserts the INTR# pin. If the transaction fails, the RESPONSE register is updated with Transaction Failed response code, and the INTR# pin is asserted. If the Type-C port is not connected or if CCG3/4 is not in PE READY state, CCG3/4 responds with a PD Command Failed response code.

Table 42 VDM_CONTROL Register

VDM_CONTROL: 2 Bytes at Address: 0x20			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	VDM mode	R*/W	0x00 – SOP 0x01 – SOP' 0x02 – SOP" If this field contains any other value, CCG3/4 responds with Invalid Argument response code.
Byte[1]	Length	R*/W	Total length of VDM message in bytes. Should be a multiple of 4 and less than 28. If this field contains any other value, CCG3/4 responds with Invalid Argument response code.

The PD_SPEC defines a set of structured VDMs that are used to select alternate mode operation within a USB Type-C link. Refer to Chapter 6, "Protocol Layer," of the PD Specification for more details.

3.3.2 VDM_EC_CONTROL

This register is used by the EC to indicate if it is ready to handle structured VDMs, as described in **Table 43**. The EC can respond to VDM requests by tunneling VDMs using the VDM_CTRL register. The CCG3/4 device uses this information to respond to structured VDM commands.

Table 43 VDM_EC_CONTROL Register

VDM_EC_CONTROL: 1 Byte CCG3/CCG4 Port-0 Address: 0x102A CCG4 Port-1 Address: 0x202A			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Control	R/W	b0: EC_CTRL_EN Value of this bit determines how CCG3/4 handles any VDM request received. 0b0: EC control disabled. CCG3/4 will handle all VDMs (NACK unknown / unsupported VDMs). 0b1: CCG will ignore all VDMs, and EC is expected to handle them. b1:b7: Reserved and should be zero

The CCG3/4 device notebook firmware can be configured to implement an automatic discovery process to identify if the attached device supports a DisplayPort alternate mode. If this support is not configured or if the peer device presents other alternate modes, then the EC must handle all alternate mode processing using structured VDMs.

3.4 Alternate Mode (DisplayPort) Registers

3.4.1 ALT_MODE_CMD

This register allows the EC to initiate commands relating to alternate mode implementation on the CCG3/4 device, as described in **Table 44**.

Table 44 ALT_MODE_CMD Register

ALT_MODE_CMD: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x101C
CCG4 Port-1 Address: 0x201C

Field	Field Name	R/W	Description
Bytes 3:2	SVID	R/W	Unique 16-bit unsigned integer, assigned by USB-IF (VID of alternate mode) 0xFF01 for DisplayPort 0x8086 for Thunderbolt
Byte 1	Alternate mode ID	R/W	Index of supported alternate modes array maintained by firmware: 0 – DisplayPort 1 – Thunderbolt Other values reserved
Byte 0 Bit 0	Alt mode data role	R/W	Data role played by CCG3/4 0 – UFP 1 – DFP
Byte 0 Bits 7:1	Alt mode command ID	R/W	Alternate mode specific command opcode: 0 – Reserved 1 – Enable EC trigger of alternate mode 2 – Disable EC trigger of alternate mode 3 – Initiate alternate mode entry (requires EC trigger of alternate mode to be enabled) 4 – Initiate alternate mode exit (requires EC trigger of alternate mode to be enabled) 5 – Alternate mode specific command. Command details to be sent through write data memory. Other opcodes reserved

See section **4.3** for more details about how these commands are handled and responded to by CCG3/4.

3.4.2 APP_HW_CMD

This register allows the EC to control the DisplayPort hardware such as the display multiplexer, as described in **Table 45**. It also controls the HPD signal of the multiplexer.

Table 45 APP_HW_CMD Register

APP_HW_CMD: 4 Bytes
CCG3/CCG4 Port-0 Address: 0x1020
CCG4 Port-1 Address: 0x2020

Field	Field Name	R/W	Description
Byte 3	CCG data role	R/W	Represents the CCG3/4 device data role: 0 – UFP 1 – DFP
Byte 2	Hardware type	R/W	Type of hardware to which the command is addressed: 1 – MUX 2 – HPD Other values reserved
Bytes 1:0	Command ID	R/W	Command ID. This takes different meanings for different hardware types. **MUX Commands:** 0 – Set data MUX to isolate mode (no connection from Type-C data pins) 2 – Set data MUX for USB 3.1 connection. 3 – Set data MUX for multifunction DisplayPort (2-lane DP + USB) connection. 4 – Set data MUX for 4-lane DisplayPort connection. Other values reserved **HPD Commands:** 0 – Enable HPD signal output 1 – Signal HPD low 2 – Signal HPD high 3 – Signal HPD IRQ 4 – Disable HPD signal output Other values reserved

3.4.3 ACTIVE_EC_MODES

This register can be used by the EC to indicate if it is maintaining alternate modes when CCG3/4 is UFP, as described in **Table 46**. The EC can enter proprietary alternate modes by tunneling VDMs to the part partner using the VDM_CTRL register. CCG3/4 uses this information to respond to the Exit Mode Structured VDM command.

Table 46 Active_EC_Modes

ACTIVE_EC_MODES: 1 Byte CCG3/CCG4 Port-0 Address: 0x1029 CCG4 Port-1 Address: 0x2029			
Field	**Field Name**	**R/W**	**Description**
Byte[0]	Status	R/W	0x00: EC has no active alternate modes. Any other value: EC has active alternate modes.

3.4.4 Alternate Mode Events

The CCG3/4 device's HPI implementation provides a generic alternate mode events infrastructure that covers all alternate modes implemented by the CCG3/4 device.

The Alternate Mode event (event code = 0xB0) is used to report any events relating to alternate modes to the EC, as described in the following table.

Table 47 Alternate Mode Event Definition

Byte	Field	Description
0	Bit 0	CCG3/4 data role 0 – UFP 1 – DFP
0	Bits 7:1	Event type. These are generic event types that are applicable to all alternate modes. 0x01 – UFP does not support any alternate modes. 0x02 – Alt mode entered. Refer to bytes 3:1 for details of mode. 0x03 – Alt mode exited. Refer to bytes 3:1 for details of mode. 0x04 – Mode discovery completed. Allows EC to trigger enter mode. 0x05 – UFP reports SVID not supported by CCG. Refer to bytes 3:1 for details of SVID. 0x06 – UFP reports SVID supported by CCG. Refer to bytes 3:1 for details of SVID. 0x07 – CCG supports alternate mode reported by UFP. Refer to bytes 3:1 for details of mode. 0x08 – UFP failed to respond to VDM. 0x09 – Cable failed to respond to VDM. 0x0A – Cable does not support alternate mode. 0x0B – Mismatch between CCG and port partner capabilities (such as DP pin configuration) 0x0C – Alternate mode specific event. More information is in bytes 7:4 of event data. Other values are reserved.
1	Bits 7:0	Alternate mode ID 0 – DisplayPort 1 – Thunderbolt

Byte	Field	Description
		Other values reserved
3:2		SVID corresponding to the event 0xFF01 for DisplayPort 0x8086 for Thunderbolt Other values reserved
6:4		Event-specific data. See description for byte 7.
7	Bits 7:0	Mode-specific event type. These events are defined per alternate mode. The following event types are defined for DisplayPort: 0x01 – DisplayPort pin configuration event. Bytes 6:4 will contain a bitmap that shows which pin configurations are supported by CCG3/4 as well as the port partner. 0x02 – DisplayPort status update event. Bytes 6:4 will contain the lower 24 bits of the status update VDO.

3.4.5 Alternate Mode Hardware Events

The alternate mode hardware event (0xB1) is used by the CCG3/4 device to notify the EC about changes relating to the hardware blocks monitored by CCG3/4, as described in **Table 48**. The only current use for this event is to report changes that are detected on the HotPlugDetect (HPD) input/output pin used for DisplayPort source or sink operation.

Table 48 Alternate Mode Hardware Event Definition

Byte	Field	Description
3	Bits 7:0	CCG3/4 data role 0 – UFP 1 – DFP
2	Bits 7:0	Hardware type: Identifies the hardware block that is triggering the event 1 – Data mux 2 – HPD signal
1:0		Event type: Currently events are only defined for HPD signal. HPD signal-specific events: 1 – HPD unplug/low detected (DP sink only) 2 – HPD plug/high detected (DP sink only) 3 – HPD IRQ detected (DP sink only) 4 – HPD status update completed (DP source only)

3.4.6 Summary

A summary of the alternate mode and DisplayPort registers is given in **Table 49**.

Table 49 Summary of PD Policy/Status Registers

Register	Address*	Size	TYPE
VDM_CTRL	0x20 0x1000 0x2000	2	Control: Command to initiate VDM transmission
ALT_MODE_CMD	N/A 0x101C 0x201C	4	Control: Alternate mode command register
APP_HW_CMD	N/A 0x1020 0x2020	4	Control: Hardware (MUX/HPD) command register
ACTIVE_EC_MODES	0x4D 0x1029 0x2029	1	Control: Enable alternate modes
VDM_EC_CONTROL	0x4E 0x102A 0x202A	1	Control: Select method of VDM handling

Note *: First row in Address column denotes CCG1/2 address. Second row in address denotes CCG4 Port1 address. Third row in address denotes CCG4 Port2 address.

4 Application Examples

This section describes several application scenarios in a system where interaction between the CCG3/4 device and EC is required over the HPI interface.

4.1 CCG3/4 Firmware Update

Update of the firmware or configuration table of the CCG3/4 device can be done in the alternate firmware flashing mode. CCG3/4 has two copies of firmware, and each copy can update the other copy. The EC can read the status registers and write to the command registers of the CCG3/4 device to update the device firmware.

4.1.1 CCG3/4 Device Firmware Update Approach

The CCG3/4 firmware implements the following flash update mechanism.

4.1.1.1 Dual Firmware Mode

In the dual firmware mode, the device flash contains two copies of the firmware in addition to the bootloader. The configuration tables are present along with each copy of the firmware. The bootloader is only responsible for validating the firmware binaries, identifying the correct firmware to start, and then starting the firmware application. Flash update is supported by the firmware itself, with each copy of the firmware having the capability to update the other copy.

This implementation allows the CCG3/4 device to stay in a USB PD contract and support USB PD and Type-C functionality while a firmware update is in progress.

4.1.2 CCG3/4 Notebook Firmware Flash Map

This section provides the firmware flash map of the CCG3/4 devices in dual firmware mode for a DRP application.

4.1.2.1 Dual Firmware Mode

Figure 14 shows the flash map for the notebook DRP implementation based on dual firmware mode on the CCG3/4 device.

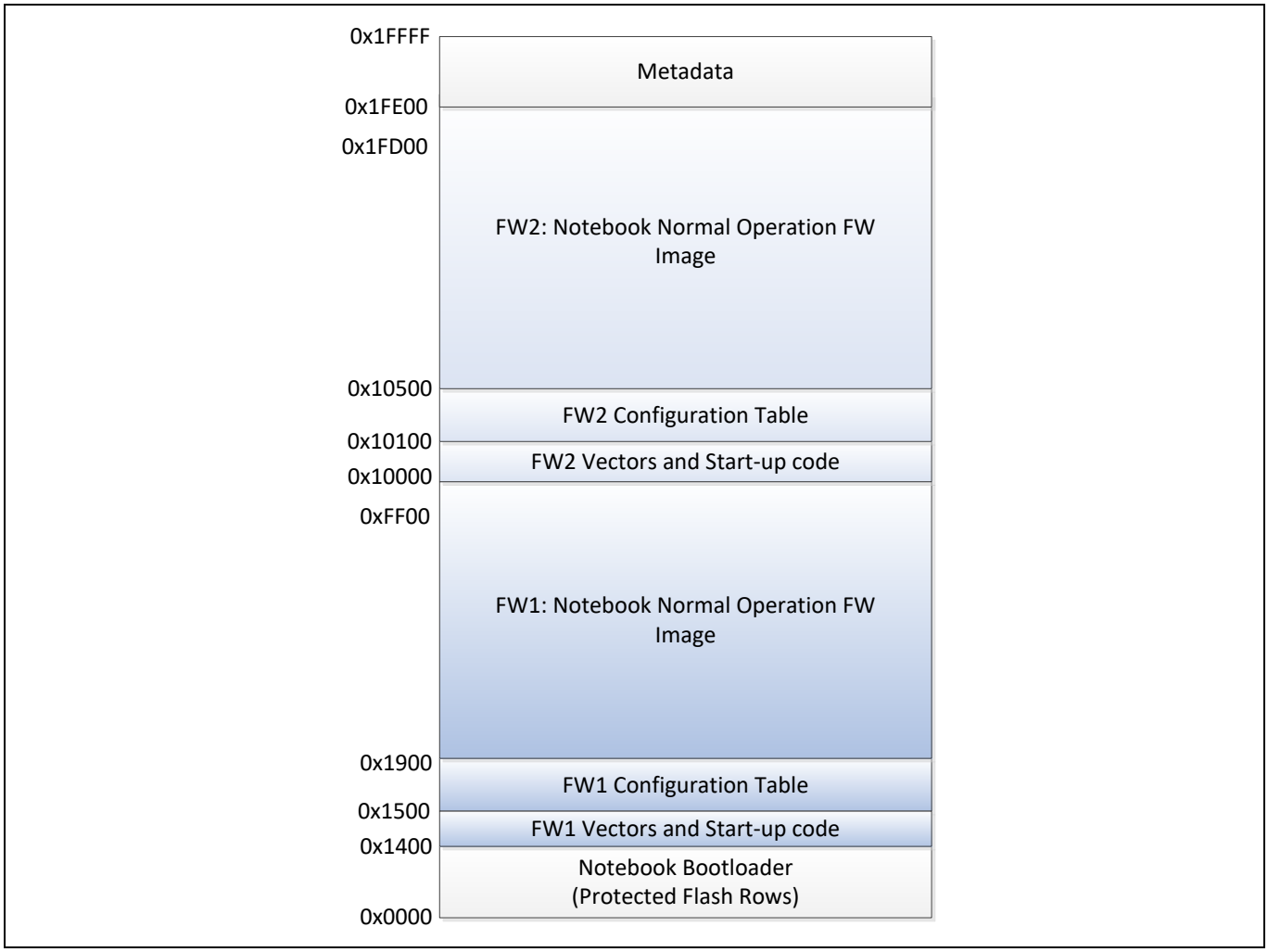

Figure 14 CCG3/4 Firmware Flash Map in Dual Firmware Mode

- Bootloader begin: 0x0000
- Bootloader end: 0x13FF
- Firmware 1 starts from 0x1400.
  - Configuration table is from 0x1500 to 0x18FF (1024 bytes).
  - Firmware code is from 0x1900 to 0xFEFF (57 KB).
- Firmware 2 starts from 0x10000.
  - Configuration table is from 0x10100 to 0x104FF (1024 bytes).
  - Firmware code is from 0x10500 to 0x1FCFF (62 KB).
- Firmware metadata is stored from 0x1FE00 to 0x1FFFF (512 bytes).
- Pseudo-metadata is used for CCG3 to facilitate non-blocking flash writes during PD operation.

Note: The sizes of FW1 and FW2 are subject to change based on the features to be supported. The start address for FW1 and FW2 can always be located from the metadata, and the metadata location is fixed. The configuration table will always start at an offset of 0x100 from the firmware vectors and start-up code.

4.1.3 Bootloader Registers

This section provides details about the status and command registers, which need to be used by the EC to update the firmware of the CCG3/4 device.

4.1.3.1 Status Registers

Refer to **Table 6** for BOOT_MODE_REASON register, **Table 8** for BOOT_LOADER_LAST_ROW register, **Table 7** for READ_SILICON_ID register.

4.1.3.2 Command Registers

Refer to **Table 10** for JUMP_TO_BOOT register, **Table 12** for ENTER_FLASHING_MODE register, **Table 14** for FLASH_ROW_READ_WRITE register, **Table 13** for VALIDATE_FW register.

4.1.4 Firmware Update in Dual Firmware Mode

The bootloader is a launcher application that will check for a valid firmware application and will then launch the application in the CCG3/4 device, as shown in **Figure 15**.The bootloader proceeds with the I^2C and HPI initialization steps only if both FW1 and FW2 are found to be invalid. If either firmware copy is valid, CCG3/4 can update the other copy of the firmware. For example, if the device is currently running FW1, FW2 can be updated with the new version. The complete firmware update procedure is as follows:

1. Check the DEVICE_MODE register.
2. Identify the firmware binary corresponding to the inactive firmware application (FW2 if FW1 is running and vice versa).
3. Initiate flashing mode entry using the ENTER_FLASHING_MODE register.
4. Clear the metadata corresponding to the firmware being updated:
   - Fill the data memory with zeroes.
   - Use the FLASH_ROW_READ_WRITE register to trigger a write of the “zero” buffer into the metadata flash row.
   - Wait for a Success response.
5. For each flash row to be updated:
   - Copy the data into the flash read/write memory.
   - Use the FLASH_ROW_READ_WRITE register to trigger writing of data to the desired flash row.
   - Wait for a Success response.
   - If read-verify is required:
     - Use the FLASH_ROW_READ_WRITE register to trigger reading of the data from the desired flash row.
     - Wait for a FLASH_DATA_AVAILABLE response from the CCG.
     - Read the data from the flash read/write memory and verify.
   - Use the VALIDATE_FW register to request the new firmware to be validated.
   - If new firmware has to be activated:
     - Use the PDPORT_ENABLE register to disable the PD ports.
     - Wait for a SUCCESS response.
     - Use the RESET register to go through a fresh start-up cycle, which will load the new firmware binary.
     - Wait for the RESET_COMPLETE event.

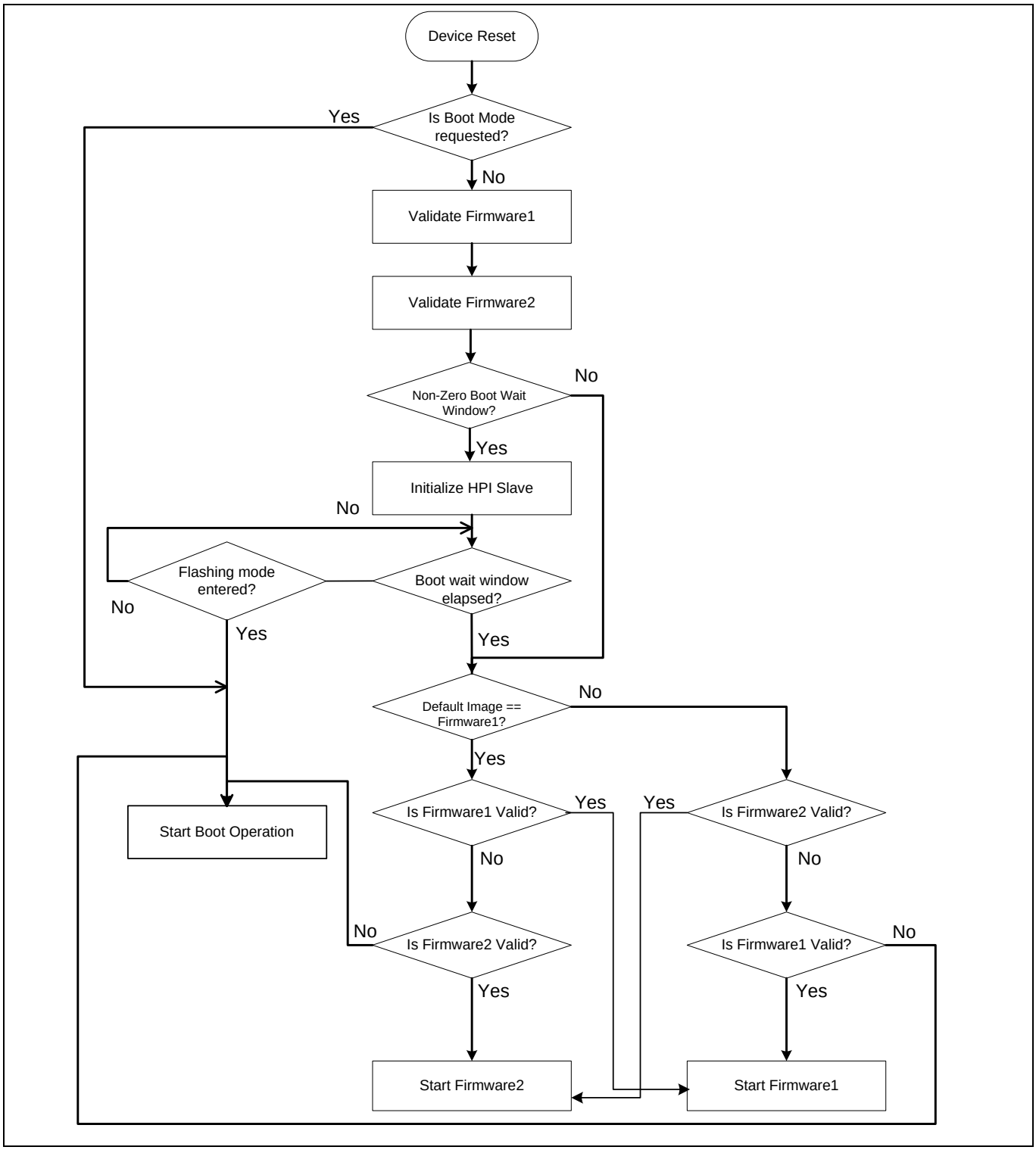

Figure 15 Bootloader Operation in Dual Firmware Mode

4.1.5 Pseudo-Code to Update CCG3/4 Firmware by EC

The following pseudo-code shows the procedure to update the firmware of the CCG3/4 device.

```
BOOL jump_to_boot (void)     /* Write to JUMP_TO_BOOT register. */
{
    UINT8 jump_sig = 0x4A;

    /* hpi_write_reg(register address, data, size); */
    return hpi_write_reg(CY_PD_REG_JUMP_TO_BOOT_ADDR, &jump_sig, 1);

        }
BOOL get_device_mode (UINT8 *mode)     /* Read DEVICE_MODE register. */
{
            hpi_read_reg(CY_PD_REG_DEVICE_MODE_ADDR, data, 1);
            /* hpi_read_reg(register address, data, size); */
     if((data[0] != 0x0)
         {
         /* Device is not in boot mode. Firmware can't be updated */
           return STATUS_ERROR;
         }
}
/* Write to ENTER_FLASHING_MODE register. */
BOOL enter_flashing_mode (void)
{
   UINT8 enter_flash_sig = 0x50;
return hpi_write_reg(CY_PD_REG_ENTER_FLASHING_MODE_ADDR,   &enter_flash_sig,
1);
        }
BOOL flash_write (UINT16 row_num, UINT16 size, UINT8 *data)
{
/* First fill the data memory with data. When EC requests a read or     write
of flash, the upper 128 bytes of the CCG3/4 Register Space address 0x80 to
0xFF) is used */
    if (STATUS_SUCCESS != hpi_write_reg(0x80, data, size))
    {
        return STATUS_ERROR;
    }
```

```
    else
    {
        /* Write to FLASH_READ_WRITE Register. */
        UINT8 flash_cmd[4];
        flash_cmd[0] = 0x46;
        flash_cmd[1] = 0x01;
        flash_cmd[2] = row_num & 0xFF;
        flash_cmd[3] = (row_num >> 8);
        return hpi_write_reg(CY_PD_REG_FLASH_ROW_READ_WRITE_ADDR, flash_cmd,
4);
    }
}
BOOL validate_fw (UINT8 fw_index)    /* Write to VALIDATE_FW register. */
{
    return hpi_write_reg(CY_PD_REG_VALIDATE_FW_ADDR, &fw_index, 1);
        }
BOOL send_device_reset (void)    /* Write to RESET register. */
{
    UINT8 reset_cmd[2];
    /* Signature Byte. */
    reset_cmd[0] = 0x52;
    reset_cmd[1] = 0x01;
    if (STATUS_SUCCESS == hpi_write_reg(CY_PD_REG_DEVICE_RESET_ADDR,
reset_cmd, 2))
     {
          /* Wait for some time. */
          Sleep (50);
          return STATUS_SUCCESS;
     }
     else
     {
          return STATUS_ERROR;
     }
        }
```

4.1.6 Error Handling

The EC can read the BOOT_MODE REASON register if the CCG3/4 device stays in boot mode. CCG3/4 stays in boot mode in the following scenarios:

- FW1 and FW2 firmware binaries are both invalid.
- The configuration table for any valid firmware binary is invalid. The firmware validates (checksum validation) the configuration table after each reset. If the configuration table is invalid, the firmware switches control back to the bootloader.

4.1.7 Configuration Table Update Procedure

The configuration table of a CCG3/4 device can be updated by itself or the configuration table data can be embedded in the firmware image and updated along with the firmware binary. The flash row read/write commands listed in section **4.1.4** can be used for this operation.

4.1.8 Reading Firmware Version from .cyacd File

The 8-byte firmware version can be read from the firmware image file. This file is provided in *.cyacd* format, which can be used by the EC to update CCG3/4's firmware. This file is a header followed by lines of flash data. Excluding the header, each line in the *.cyacd* file represents an entire row of flash data. The data is stored as ASCII data in big-endian format. The *.cyacd* file format is as follows.

The header record has this format:

[4-byte SiliconID][1-byte SiliconRev][1-byte Checksum Type]

The data records have this format:

[1-byte ArrayID][2-byte RowNumber][2-byte DataLength][N-byte Data][1-byte Checksum]

The 8-byte firmware version is stored at an offset of 0x0100 from the start address of the firmware. As each data record contains 128/256 bytes of data (CCG4 device's flash row size is 256), the firmware version is stored in the first 8 bytes of the third row of the *.cyacd* image file. The EC reads this version from the image file and determines if a firmware update is required on the CCG3/4 device.

4.2 Initialization of PD Commands over HPI

In a Type-C system, the EC needs to communicate with the CCG3/4 device while establishing the power contract with the attached Type-C device. For example, in a typical notebook design, the EC controls the Battery Charger Controller (BCC) and then communicates with the CCG3/4 device to become the power provider or consumer based on the charge present in the battery. Typically, the EC initiates the following events with the CCG3/4 device.

1. Update source/sink PDOs to vary power profiles based on the connected Type-C device
2. Data role swap (DFP or UFP)
3. Power role swap (power provider or power consumer)
4. VCONN swap
5. EC behavior whenever the barrel charger is connected or disconnected from a CCG3/4 enabled notebook

Subsequent sections explain the EC-CCG3/4 initialization sequence and handling of these events by the EC and CCG3/4.

4.2.1 EC-CCG3/4 Initialization Sequence

Figure 16 shows the EC-CCG3/4 initialization sequence after power on, device reset, or JUMP_TO_BOOT mode command.

1. EC starts communication with CCG3/4 after receiving Reset Complete (0x80) event message code.
2. If tBootWait (100 ms) timeout value is non-zero or if normal firmware image is invalid or nonexistent, CCG3/4 device enters boot mode, initializes HPI, and notifies EC with Reset Complete event. Otherwise, CCG3/4 device enters normal firmware operation after tBootWait timeout.
3. EC has to check if CCG3/4's normal firmware is operational or if it is in boot mode by reading DEVICE_MODE register.
4. If EC needs to change the PDO mask or provide barrel connect/disconnect status, it sets the SELECT_SOURCE_PDO and SELECT_SINK_PDO registers.
5. EC should set the EVENT_MASK register to enable CCG3/4 to send relevant events to EC during operation. By default, no events are sent to EC, except Reset Complete event and Message Queue Overflow events. These events are not maskable.
6. After setting required registers, EC sends EC Initialization Complete command using PD_CONTROL register to inform CCG3/4 to proceed with its initialization.
7. CCG3/4 initializes Type-C interface only after receiving this command. If this command is not sent within 100 ms, CCG3/4 autonomously initializes Type-C interface.
8. CCG3/4 initiates and handles all PD-related negotiations by itself, based on PD profile configuration. When the negotiations are complete, the PD Contract Established event is sent to EC, if enabled.
9. The contract state bit in PD_STATUS register indicates if contract exists. Subsequently, EC initiates alternate mode negotiations as described in section **4.3**.

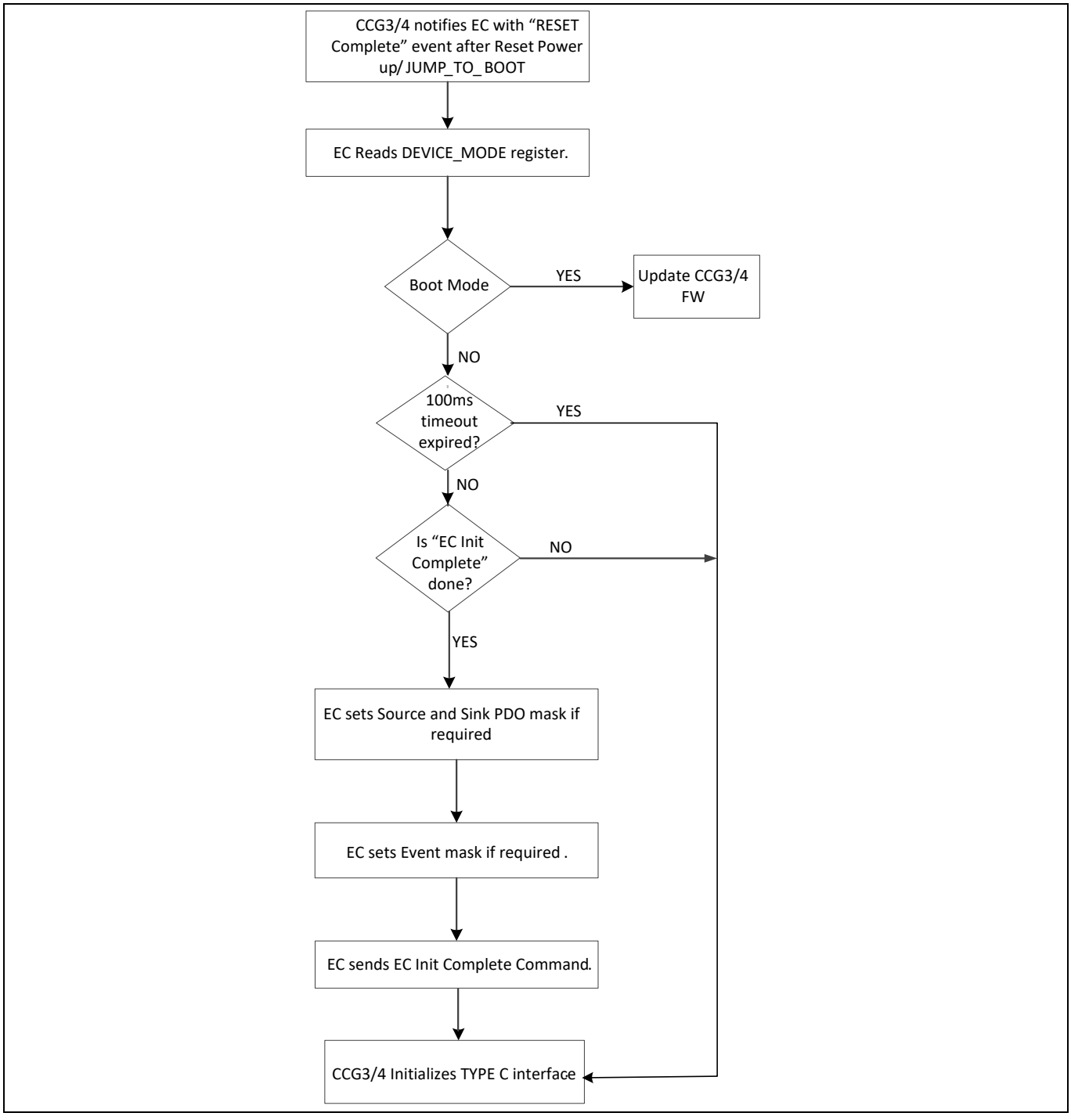

Figure 16 EC-CCG3/4 Initialization Sequence

The pseudo-code for the EC-CCG3/4 initialization sequence is as follows. Refer to the attached Visual Studio based reference code (*HPI Reference Code**HPI*\) for more details.

hpi_read_reg and hpi_write_reg are the functions to read and write a CCG3/4 register.

```
BOOL ec_init (void)

{

/* Check the active Device mode of CCG3/4 by reading DEVICE_MODE
register.*/
```

```
            hpi_read_reg(CY_PD_REG_DEVICE_MODE_ADDR, data, 1);
             /* hpi_read_reg(register address, data, size); */
      if(data[0] == 0x0)
      {
             /* Device in boot mode. Application can't execute */
            return STATUS_ERROR;
      }
printf ("Setting EVENT_MASK to enable DP and PD negotiation complete
events.\n");
      /* Setting EVENT_MASK to enable all events */
      data[0] = 0xFF;
      data[1] = 0xFF;
      data[2] = 0xFF;
      data[3] = 0xFF;
            if (STATUS_SUCCESS != hpi_write_reg
      (CY_PD_REG_EVENT_MASK_MASK,data,1)
      {
           /* EVENT_MASK register write failed */
                printf ("EVENT_MASK register write failed.\n");
          return STATUS_ERROR;
      }
      else
      {
          /* Read response. */
          if (CY_PD_RESP_SUCCESS != read_event())
    /* read_event() is to read the RESPONSE_REGISTER after the INTR
    Line is asserted. */
           /* Command failed: Set EVENT_MASK */
            {
                 return STATUS_ERROR;
            }
      }
```

```
/* Setting the  SOURCE_PDO_MASK as 0x01 by writing to SELECT_SOURCE_PDO
register. Default only   fixed 5V PDO  */
      data[0] = 0x01;
if (STATUS_SUCCESS != hpi_write_reg(CY_PD_REG_SOURCE_PDO_MASK _ADDR, data,
1))
      {
         /* SELECT_SOURCE_PDO register write failed */
          printf ("Source_PDO_Mask register write failed.\n");
         return STATUS_ERROR;
      }
      else
      {
            /* Read response. */
            if (CY_PD_RESP_SUCCESS != read_event())
/* read_event() is to read the RESPONSE_REGISTER after the INTR line is
asserted */
            {
              /* Command failed: SELECT_SOURCE_PDO */
                  return STATUS_ERROR;
            }
      }

      /* Send EC_INIT_COMPLETE command. */
      data[0] = 0x10;
      if (STATUS_SUCCESS != hpi_write_reg(CY_PD_REG_PD_CTRL_ADDR, data, 1))
      {
           /* PD_CONTROL register write failed */
           printf ("PD Control register write failed.\n");
          return STATUS_ERROR;
      }
      else
      {
            /* Read response. */
            if (CY_PD_RESP_SUCCESS != read_event ())
                /* read_event() is to read the RESPONSE_REGISTER after the
                INTR line is asserted*/
```

```
            {
                /* Command failed: EC_INIT_COMPLETE. Device already
                Initialized */
                return STATUS_ERROR;
            }
      }

      /* Wait for device to enter a PD contract with the Port Partner. */

      Sleep(500);

      /* Check if PD contract established by reading PD_STATUS register. */
            hpi_read_reg (CY_PD_REG_PD_STATUS, data, 4);
/* Check the PD Contract Established bit (10th) of PD_STATUS register.*/
      if((data[1] & 0x04)== 0)
      {
            /* PD contract not Established */
      printf("PD contract not established \n");
      }
      else
      {
            /* PD contract Established */
            printf("PD contract is established \n");

      }

      /* Read all events from CCG3/4. */
            read_all_events ();
/* read_all_events() is to read all the events from RESPONSE_REGISTER after
the INTR line is asserted, until the INTR line is de-asserted. */
      return STATUS_SUCCESS;
}

UINT8 read_event (void)
{
```

```
      if(STATUS_SUCCESS != wait_for_event(1000))
      {
            printf ("Response Register Read Failed.\n");
            return CY_PD_RESP_NO_RESPONSE;
      }
      else
      {
            /* Return Response/Event Opcode. */
            return received_event_code;
      }
}

BOOL wait_for_event (UINT16 timeout_ms)
{
      UINT8 state;
      /* Get the start time. */
      start_time = clock();
      while(1)
      {
        if(gl_override_gpio)
           Sleep(500);
           else
            {
             elapsed_time = clock() - start_time;
             if(elapsed_time > timeout_ms)
             {
              printf("Timeout in response\n");
              return STATUS_ERROR;
             }
            /* Check for INTR GPIO status. */
             if (STATUS_SUCCESS != hpi_comm->get_gpio_status (&state))
             {
              /* GPIO Status Read Failed. */
              printf ("GPIO Status Read Failed");
```

```
            return STATUS_ERROR;
          }
        }

      /* Read response if INTR GPIO is asserted. */
        if (gl_override_gpio || !state)
                {
                  if (STATUS_SUCCESS != read_event_reg ())
                    {
                      printf ("Response Register read failed.\n");
                return STATUS_ERROR;
                    }
            else
              {
                return STATUS_SUCCESS;
              }
                  }
    }
    return STATUS_SUCCESS;
}

BOOL read_event_reg (void)
{
    tEVENT event;

    /* First read the response register. */
    if (STATUS_SUCCESS != hpi_comm->reg_read (CY_PD_REG_RESPONSE_ADDR,
            (UINT8 *)&event, 2))
    {
        printf("reg_read failed @ 1\n");
      return STATUS_ERROR;
    }
    else
    {
```

```
        /* Check if length field is non zero. */
        if (event.event_length != 0)
        {
        }
    }

    /* Write to CLEAR_INTR register to deassert INTR_GPIO. */
    if (STATUS_SUCCESS != clear_intr ())
    {
            printf("CLEAR_INTR failed\n");
        return STATUS_ERROR;
    }

    /* Invoke event handler. This event handler is registered by main
    Application */
    hpi_comm->event_handler (&event);

    return STATUS_SUCCESS;
}
```

The following sections explain the events that can be initiated by the EC based on the attached port partner to the CCG3/4 device. In a notebook design, the EC makes these decisions based on charge present in the battery and then sends commands to the CCG3/4 device.

4.2.2 Update Source PDO

Refer to section **3.2.2.2** for more details.

4.2.3 Update Sink PDO

Refer to section **3.2.2.4** for more details.

4.2.4 Data Role Swap

The EC sends this command whenever it wants to swap the existing data role of CCG3/4 for an already established PD contract. This results in a DR_SWAP message sent by the CCG3/4 device to the port partner. To initiate a data role swap (DR_SWAP), the EC needs to read the current data power role of the CCG3/4 device (second bit of PD_STATUS register) and initiate a DR_SWAP, if needed. Writing 0x05 to the PD_CONTROL register of CCG3/4 initiates the DR_SWAP.

Here is one example of when an EC initiates a DR_SWAP. Assume CCG3/4 is the Type-C controller of a notebook whose battery is dead (dead battery scenario). In this example, the CCG3/4 device will be advertising Rd by default on a Type-C interface. If a DRP (such as a monitor) connects to the notebook, it will detect Rd and apply 5 V on VBUS. This will power up the CCG3/4 device, which will start power negotiations with the monitor. The

notebook eventually settles into a UFP role and starts sinking power from the monitor. Once PD negotiations are complete, the EC can trigger a DR_SWAP (if the monitor has not already done so) using this command to switch to a DFP role from the UFP role. This swap then results in the notebook and monitor entering the DisplayPort alternate mode to share video.

The EC should check the following conditions before applying this command:

- EC ensures that this command is not sent when CCG3/4 is already in an alternate mode.
- CCG3/4 device responds with PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.
- If CCG3/4 device is configured to be a DFP or a UFP only port, it responds with Not Supported error code and does not handle this request.
- If DR_SWAP message transmission fails (that is, GOOD_CRC is not received), CCG3/4 device responds with Transaction Failed error code.
- If DR_SWAP message is transmitted successfully, CCG3/4 responds with Success response code and starts the Sender Response Timer. If the Accept message is received, CCG3/4 notifies the EC with the Accept Message Received event and then swaps its port data role. If a Reject message is received, CCG3/4 notifies the EC with the Reject Message Received event and retains its existing port data role. If a Wait message is received, CCG3/4 notifies the EC with the Wait Message Received event and retrains its existing port data role.

In all the above cases, the CCG3/4 device notifies the EC with the SWAP_COMPLETE event that holds the result of the swap request. If the Sender Response Timer times out, CCG3/4 notifies the EC with the Sender Response Timer Timeout event.

Pseudo-code for initiating a data role swap (DR_SWAP) is as follows:

```
BOOL check_for_dr_swap (void)

{

/* Check the current data role of CCG3/4 by reading the 6th bit of PD_STATUS
register. If device data role is UFP, initiate a DR_SWAP. */

           hpi_read_reg (CY_PD_REG_PD_STATUS_ADDR, data, 4);

      if (!(data[0] & 0x40))

      {

/* Device is UFP, initiate DR_SWAP by writing value 5 to PD_CONTROL register.
*/

           data[0] = 0x05;

if (STATUS_SUCCESS !=hpi_write_reg(CY_PD_REG_PD_CTRL_ADDR, data, 1))

           {

                /* PD_CONTROL register write failed */

               return STATUS_ERROR;

           }

           else

           {

                /* Check for Success Response. */
```

```
                if (CY_PD_RESP_SUCCESS != read_event ())
                {
                     /* Command failed: TRIGGER_DR_SWAP */
                     printf("Initialization to data role swap is failed");
                    return STATUS_ERROR;
                }
            }

            /* Wait for DR SWAP to complete. */
            Sleep (1000);
/* Check the current data role of CCG3/4 by reading the 6th bit of PD_STATUS register.If not DFP, exit . */
            hpi_read_reg(CY_PD_REG_PD_STATUS_ADDR, data, 4);
            if (!(data[0] & 0x40))
            {
                 /*DR SWAP not successful. Exit the application */
                 printf("data role swap is not successful");
                return STATUS_ERROR;
            }
            else
            {
                 /*DR Swap Successful */
            }
      }
      return STATUS_SUCCESS;
}
```

4.2.5 Power Role Swap

The EC sends this command whenever it wants to swap the existing power role of CCG3/4 for an already established PD contract. This results in a PR_SWAP message sent by the CCG3/4 device to the port partner. To initiate a power role swap (PR_SWAP), the EC needs to read the current port power role of the CCG3/4 device (eighth bit of PD_STATUS register) and initiate PR_SWAP, if needed. Writing 0x06 to the PD_CONTROL register of CCG3/4 initiates the PR_SWAP.

When CCG3/4 is acting as a power source:

- If the PR_SWAP message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with the Transaction Failed error code.

- If the PR_SWAP message is transmitted successfully, the CCG3/4 device responds with the Success response code and starts the Sender Response Timer. CCG3/4 waits for the response from the port partner.
  - If an Accept message is received, CCG3/4 device notifies the EC with the Accept Message Received event. It then turns off the power supply and asserts Rd. Then CCG3/4 sends a PS_RDY message to the port partner. If transmission of the PS_RDY message fails or the port partner does not respond with PS_RDY, CCG3/4 sends a Hard Reset to the port partner and notifies the EC with a Hard Reset Sent event.

    If a PS_RDY message is received from the port partner, CCG3/4 notifies the EC with a PS_RDY Message Received event, which marks successful completion of a power role swap sequence. CCG3/4 notifies the EC with a Swap Complete event, with the Swap Response field set to Accept. Then the CCG3/4 device moves to the Sink Startup state, and an explicit power contract negotiation process starts.
  - If a Reject message is received, CCG3/4 notifies the EC with a Reject Message Received event and a Swap Complete event with the Swap Response field set to Reject.
  - If a Wait message is received, CCG3/4 notifies the EC with a Wait Message Received event and a Swap Complete event with Swap Response field set to Wait.
- If Sender Response Timer times out, CCG3/4 notifies EC with Sender Response Timer Timeout event.

When CCG3/4 is acting as a power sink:

- If the PR_SWAP message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with the Transaction Failed response code.
- If the PR_SWAP message is transmitted successfully, the CCG3/4 device responds with the Success response code and starts the Sender Response Timer. CCG3/4 waits for the response from the port partner.
  - If an Accept message is received, CCG3/4 notifies the EC with the Accept Message Received event. It then stops sinking power.

    When a PS_RDY message is received from port partner, CCG3/4 notifies the EC with a PS_RDY Message Received event and asserts Rp. CCG3/4 starts sourcing power and sends the PS_RDY message to the port partner. This marks successful completion of the power role swap sequence. CCG3/4 notifies the EC with a Swap Complete event with the Swap Response field set to Accept. Then CCG3/4 moves to the Sink Startup state, and explicit power contract negotiation process starts.
  - If port partner does not send a PS_RDY message after sending the Accept in response to a PR_SWAP command, the CCG3/4 device sends a HARD_RESET to the port partner and notifies the EC with a Hard Reset Sent event.
  - If CCG3/4 is not able to successfully transmit the PS_RDY message to the port partner after a PS_RDY message is received, CCG3/4 sends a HARD_RESET to the port partner and notifies the EC with a Hard Reset Sent event.
  - If a Reject message is received, the CCG3/4 device notifies the EC with a Reject Message Received event and a Swap Complete event with the Swap Response field set to Reject.
  - If a Wait message is received, CCG3/4 notifies the EC with a Wait Message Received event and Swap Complete event with the Swap Response field set to Wait.
- If the Sender Response Timer times out, the CCG3/4 device notifies the EC with a Sender Response Timer Timeout event.

4.2.6 Switch On/Off VCONN

The EC can use these commands to turn on/off VCONN. If the Type-C port is not connected, the CCG3/4 device responds with a PD Command Failed response code. Otherwise, the CCG3/4 device turns on/off VCONN and responds with a Success response code. To switch VCONN on or off, the EC needs to read the 13th bit of the

PD_STATUS register of the CCG3/4 device. Writing 0x07/0x08 to the PD_CONTROL register of the CCG3/4 device switches VCONN on/off respectively.

4.2.7 Trigger VCONN Source Swap

The EC uses this command to request an exchange of the VCONN source. The EC needs to read the 12th bit of the PD_STATUS register of the CCG3/4 device. Writing 0x09 to the PD_CONTROL register of the CCG3/4 device triggers a VCONN source swap. The possible responses to this command are as follows:

- CCG3/4 responds with a PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.
- If the VCONN_SWAP message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with a Transaction Failed error code.
- If the VCONN_SWAP message is transmitted successfully, the CCG3/4 device responds with a Success response code and starts the Sender Response Timer. The CCG3/4 device waits for a response from the port partner.
- If an Accept message is received, CCG3/4 device notifies the EC with an Accept Message Received event.

In the case of Accept, if the CCG3/4 device was the VCONN source before sending VCONN_SWAP:

1. CCG3/4 waits for a PS_RDY message from the port partner. When a PS_RDY message is received, the CCG3/4 device turns off VCONN and notifies the EC with a PS_RDY Message Received event. This marks successful completion of the VCONN swap sequence. The CCG3/4 device notifies the EC with a Swap Complete event with the Swap Response field set to Accept.
2. If a PS_RDY message is not received (timeout of VCONN On Timer), the CCG3/4 device sends a HARD_RESET to the port partner and notifies the EC with a Hard Reset Sent event.

In the case of Accept, if CCG3/4 was not the VCONN source before sending VCONN_SWAP:

1. CCG3/4 turns on VCONN and sends PS_RDY to the port partner. If the PS_RDY transmission is successful, the VCONN swap sequence is complete. The CCG3/4 device notifies the EC with a Swap Complete event with the Swap Response field set to Accept.
2. If CCG3/4 is not able to successfully transmit the PS_RDY message to the port partner, the CCG3/4 device sends a HARD_RESET to the port partner and notifies the EC with a Hard Reset Sent event.

a) If a Reject message is received, the CCG3/4 device notifies the EC with a Reject Message Received event and a Swap Complete event with the Swap Response field set to Reject.

b) If a Wait message is received, the CCG3/4 device notifies the EC with a Wait Message Received event and a Swap Complete event with the Swap Response field set to Wait.

c) If the Sender Response Timer times out, the CCG3/4 device notifies the EC with a Sender Response Timer Timeout event.

4.2.8 Retrieve Source Capabilities

The EC uses this command to retrieve the source capabilities of the port partner. The EC needs to write 0x0A to the PD_CONTROL register to retrieve source capabilities from the port partner. If the CCG3/4 device is configured to be a DFP or a UFP only port, it will respond with a Not Supported error code and will not handle this request.

If the CCG3/4 device is acting as a source, possible responses of the CCG3/4 device are as follows:

- If the Get_Source_Cap message is transmitted successfully, the CCG3/4 device responds with a Success response code to the EC and starts the Sender Response Timer. The CCG3/4 device waits for the response from the port partner.

- If the port partner responds with a source capabilities message, the CCG3/4 device notifies the EC with a Source Capabilities Message Received event and places the received source capabilities in the data memory.
- If a Reject message is received, the CCG3/4 device notifies the EC with a Reject Message Received event.
- If the Sender Response Timer times out, the CCG3/4 device notifies the EC with the Sender Response Timer Timeout event.
- If the Get_Source_Cap message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with the Transaction Failed error code.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

If the CCG3/4 device is acting as sink:

- If the Get_Source_Cap message is transmitted successfully, this initiates a power contract renegotiation sequence with the port partner.
- If the Get_Source_Cap message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with Transaction Failed error code.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

4.2.9 Retrieve Sink Capabilities

The EC uses this command to retrieve the sink capabilities of the port partner. The EC needs to write 0x0B to the PD_CONTROL register to retrieve sink capabilities from the port partner. The possible responses to this command are as follows:

- If the Get_Sink_Cap message is transmitted successfully, the CCG3/4 device responds with the Success response code to the EC and starts the Sender Response Timer. CCG3/4 waits for the response from the port partner. If the port partner responds with the sink capabilities message, the CCG3/4 device notifies the EC with a Sink Capabilities Message Received event.
- If a Reject message is received, CCG3/4 notifies the EC with a Reject Message Received event.
- If the Sender Response Timer times out, the CCG3/4 device notifies the EC with a Sender Response Timer Timeout event.
- If the Get_Sink_Cap message transmission fails (GOOD_CRC is not received), the CCG3/4 device responds with the Transaction Failed error code.
- CCG3/4 responds with PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

4.2.10 Send Hard Reset

The EC uses this command to send a Hard Reset packet to the port partner. The EC needs to write 0x0D to the PD_CONTROL register of the CCG3/4 device.

- CCG3/4 notifies the EC with a Hard Reset Sent event after transmitting the Hard Reset packet.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

4.2.11 Send Soft Reset

The EC uses this command to send a Soft Reset packet to the port partner. The EC needs to write 0x0E to the PD_CONTROL register of the CCG3/4 device.

- If the CCG3/4 device successfully transmits the Soft Reset packet, the CCG3/4 device notifies the EC with a Soft Reset Sent event and starts the Sender Response Timer.
- If the Soft Reset packet transmission fails, the CCG3/4 device sends a Hard Reset packet and notifies the EC with a Hard Reset Sent event.
- If the port partner responds with an Accept message, the CCG3/4 device notifies the EC with the Accept Received event message.
- If the port partner does not respond, the CCG3/4 device notifies the EC with a Sender Response Timer Timeout event.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

4.2.12 Send Cable Reset to EMCA

The EC uses this command to send a Cable Reset packet to an EMCA. The EC needs to write 0x0F to the PD_CONTROL register of the CCG3/4 device.

- If CCG3/4 successfully transmits the Cable Reset packet, CCG3/4 notifies the EC with a Cable Reset Sent event.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

The EC should consider the following scenarios before sending this command to CCG3/4:

- CCG3/4 should be in DFP mode, and an explicit PD contract should exist.
- CCG3/4 should be the supplier of VCONN. This information is present in the PD_STATUS register.
- VCONN should be turned on. This information is also present in the PD_STATUS register

4.2.13 Send Soft Reset to EMCA

The EC uses this command to send a Soft Reset packet to an EMCA. The EC needs to write 0x12 to the PD_CONTROL register of the CCG3/4 device.

The CCG3/4 device provides separate commands to send the Soft Reset packet to SOP’ and SOP’’ devices.

- If CCG3/4 successfully transmits the Soft Reset packet, the CCG3/4 device notifies the EC with a Soft Reset Sent event and starts the Sender Response Timer.
- If the EMCA responds with an Accept message, the CCG3/4 device notifies the EC with the Accept Received event message.
- If the EMCA does not respond, the CCG3/4 device notifies the EC with the Sender Response Timer Timeout event.
- If CCG3/4 fails to transmit a Soft Reset packet, the CCG3/4 device responds with the PD Transaction Failed error code and moves back to the Ready state.
- CCG3/4 responds with the PD Command Failed error code if it is in the middle of handling another command. If the EC receives this error code, it needs to wait and retry accessing the CCG3/4 device later.

In general, note that the EC can use these commands to reset an EMCA’s protocol state machine in case of protocol errors in SOP’/SOP’’ communication. If a Soft Reset to the EMCA also fails, the EC is expected to use the PD_CONTROL register to send a Cable Reset command to the EMCA.

4.2.14 Barrel Connect and Disconnect

A notebook PC can get powered via a barrel charger or Type-C port. Barrel charging has a higher precedence than Type-C port charging. If a barrel is connected, the notebook is expected to stop sinking power from the Type-C port. If a barrel is not connected, the notebook is expected to get charged over the Type-C port by a Type-C source.

Barrel charger connection and disconnection are communicated by the EC to CCG3/4 using the SELECT_SOURCE_PDO and SELECT_SINK_PDO registers. The MSB of these registers is used by the EC to indicate the barrel connect status. If this bit is set, CCG3/4 sets the Externally Powered field in the source/sink PDOs when advertising capabilities. If this bit is not set, CCG3/4 clears the Externally Powered field in the source/sink PDOs.

4.2.14.1 Barrel Connect

This section explaines the EC's role and CCG3/4's operational overview when a barrel is connected to a notebook. When the Barrel Connect event is detected by the EC, it writes to the SELECT_SOURCE_PDO and SELECT_SINK_PDO registers to indicate that a barrel is connected to a notebook or PC.

When the SELECT_SOURCE_PDO and SELECT_SINK_PDO registers are updated, the CCG3/4 device's subsequent actions depend on the current state of the Type-C port and its current role. The CCG3/4 device's actions are described in the following three scenarios when a barrel is connected to a notebook:

Type-C Port Is Not Connected

The sequence of operations when a barrel is connected to notebook and a Type-C port is not connected can be summarized as follows.

The EC updates the SELECT_SINK_PDO register and sets the MSB of this register to indicate that a barrel is connected. As the barrel is connected, charging over the Type-C port is not required. The EC chooses only a fixed 5-V sink PDO in the SELECT_SINK_PDO register. If a Type-C source such as a power adapter connects to the notebook over the Type-C port, the CCG3/4 device will request a 5-V power contract only. The EC should make sure that the notebook charging circuitry is connected to the barrel. This will ensure that the notebook sinks no power over the Type-C port.

The EC updates the SELECT_SOURCE_PDO register and sets the MSB of this register to indicate that a barrel is connected. The EC updates the source PDO mask and typically chooses higher capability source PDOs as a barrel is connected.

After the Type-C connect event, CCG3/4 advertises an "Externally powered" status in the source and sink PDOs, whichever is applicable based on the port partner's power role.

Type-C Port Is Connected and CCG3/4 Is Operating as Provider

The EC updates the source and sink PDO masks indicating the barrel connected status and typically adds extra PDOs at a higher power level as a barrel is connected.

Since the CCG3/4 device is operating as a provider, it sends updated source capabilities to the port partner indicating that the notebook is now externally powered. Refer to section **3.2.2.3**, for details.

Type-C Port Is Connected and CCG3/4 Is Operating as Consumer

The EC disconnects charging from VBUS and connects notebook charging circuitry to the barrel.

The EC updates both the source and sink PDO masks indicating the barrel connected status. It unmasks only the fixed 5-V sink PDO.

The CCG3/4 device requests source capabilities of the provider and requests only a 5-V power contract. Refer to section **3.2.2.5** for details.

Once the contract is reestablished, the EC can trigger a power role swap, if required, through the PD_CONTROL register as the notebook can now source power to the port partner.

4.2.14.2 Barrel Disconnect

Figure 17 illustrates the sequence of activities when a barrel is disconnected. If the CCG3/4 device was already in contract with a power source such as a power adapter, the CCG3/4 device starts PD negotiations with a Get Source Capability command.

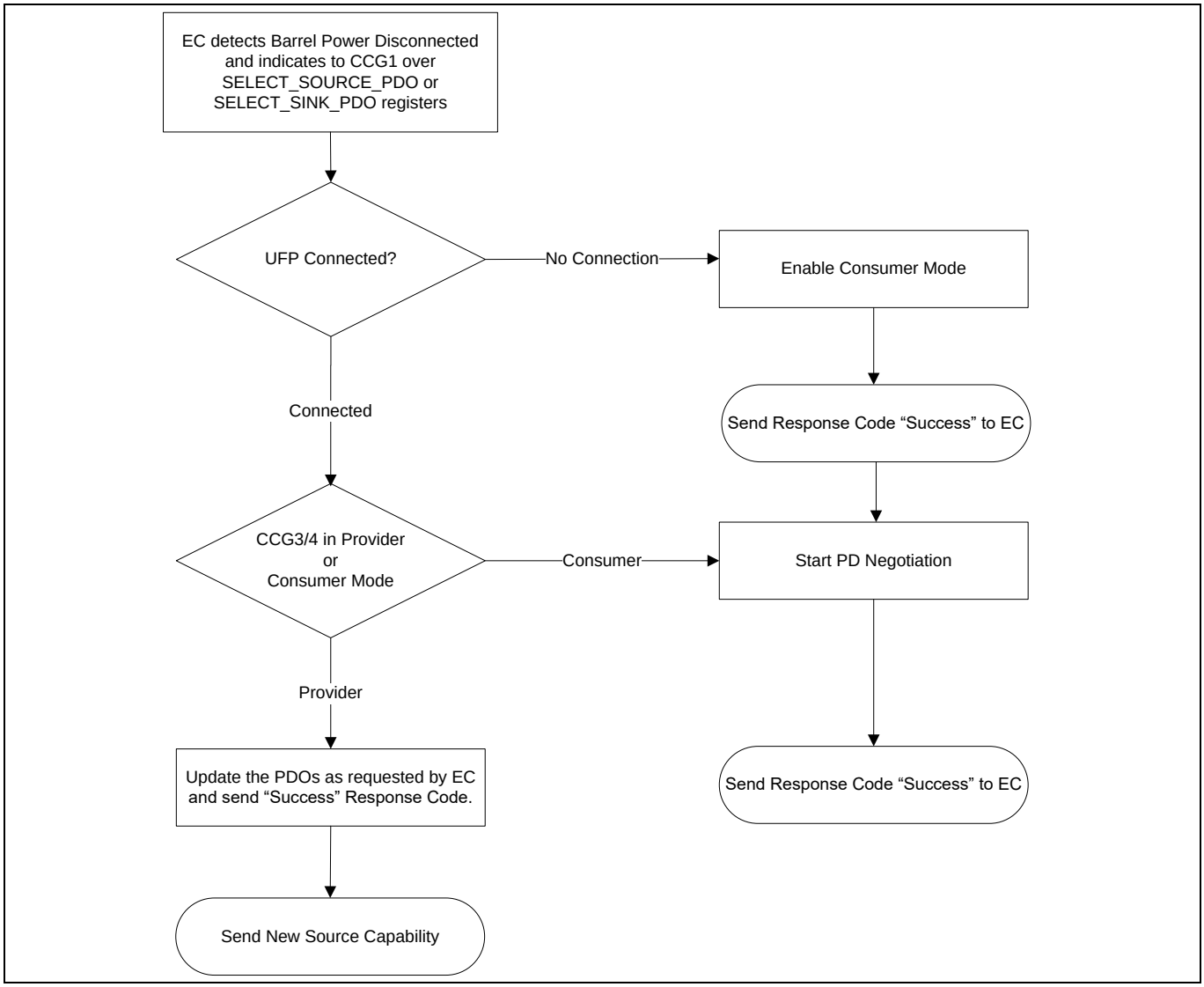

Figure 17 Barrel Disconnect Sequence

4.2.15 Updating Type-C Profile

The CCG3/4 device reads the PD profile at initialization to determine the default Type-C current to be advertised. The EC can choose a different Type-C current profile at run time by writing to the PD_CONTROL

register. For example, writing 0x01 to the PD_CONTROL register sets a Type-C 1.5A profile, whereas 0x02 sets a Type-C 3A profile.

4.3 VDM Handling and DisplayPort

The USB PD Specification allows ports to exchange nonstandard information in the form of VDMs. By default, the CCG3/4 device handles structured VDMs related to the DisplayPort alternate mode. Unstructured VDMs are sent to the EC as is, to which the EC is expected to respond. Refer to section 6.4.4 of the USB Power Delivery Specification Rev. 2.0, Version 1.1 for details on how to use these messages.

4.3.1 Sending VDMs to the Port Partner

The VDM_CONTROL register is used by the EC to request CCG3/4 to send one VDM. Refer to section **3.3.1** for more details on the VDM_CONTROL register. **Figure 18** shows the sequence of tunneling VDMs to the port partner.

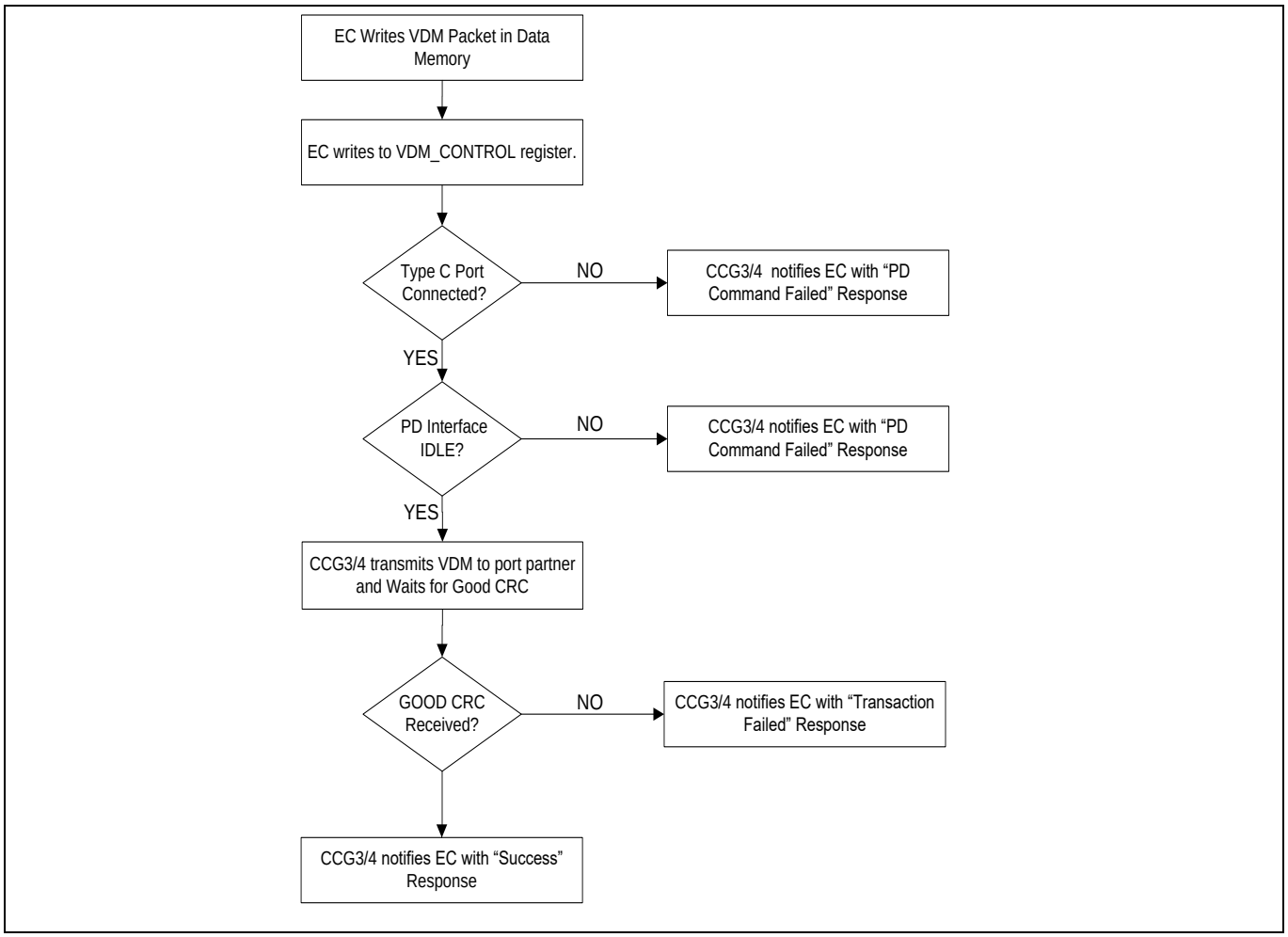

Figure 18 Sending VDMs to the Port Partner

4.3.2 Response from Port Partner

When the port partner receives a VDM from the CCG3/4 device, it processes the command, performs any additional tasks, if needed, and may send a response back to the CCG3/4 device. Once a GOOD_CRC is sent back and the VDM is received from the port partner or the EMCA, CCG3/4 raises the VDM Received event. The

CCG3/4 device updates the length of the VDM in the RESPONSE register, and the VDM data is updated in read data memory.

The pseudo-code for sending and receiving VDMs is as follows:

```
BOOL send_vdm (
          UINT8 sop_type,        //VDM mode as defined in VDM_CONTROL register
          UINT8 *vdm_data,       // VDM command
          UINT8 vdm_size)        // Length of VDM command
{
     /* First write to data memory. */
if (STATUS_SUCCESS != hpi_comm->reg_write (CY_PD_REG_FWDATA_MEMEORY_ADDR,
vdm_data, vdm_size))
     {
            return STATUS_ERROR;
     }
     else
     {
          /* Write to VDM control register. */
          UINT8 u_vdm_cmd[2];
          u_vdm_cmd[0] = sop_type;
          u_vdm_cmd[1] = vdm_size;
return hpi_comm->reg_write (CY_PD_REG_U_VDM_CTRL_ADDR, u_vdm_cmd, 2);
     }
}

BOOL read_vdm_pkt (void)
{
  /*wait_for_event is to wait till the INTR line is asserted. */
     if (STATUS_SUCCESS != wait_for_event(5000))
     {
          printf ("Response Register Read Failed.\n");
          return STATUS_ERROR;
     }
  /*received_event_code is based on the specific alternate mode events.
The event type value of alternate mode (0x0C) is used to send DisplayPort
specific events to EC */
```

```
    if(CY_PD_RESP_VDM_RECEIVED != received_event_code)
    {
        printf ("VDM Received Event Not Received.\n");
        return STATUS_ERROR;
    }
    return STATUS_SUCCESS;
}
```

4.3.3 Unstructured VDMs

The CCG3/4 device does not handle (initiate or respond to) any unstructured VDMs on its own and forwards all unstructured VDMs as-is to the EC. If the EC wants to send an unstructured VDM, it should follow the process described in section **4.3.1**. If the EC receives an unstructured VDM, it should follow the sequence described in section **4.3.2** to read and respond back.

4.3.4 Alternate Mode Handling

The CCG3/4 device implements a generic alternate mode infrastructure that allows multiple alternate modes to be implemented by the CCG3/4 device firmware. The command and events used for alternate mode negotiation have been made generic to allow different alternate modes to be supported.

The CCG3/4 device notebook with AUTO DP (Auto DisplayPort) mode support implements an automatic discovery process to identify if the attached device supports alternate modes. The CCG3/4 device supports only DisplayPort specification in the alternate mode discovery.

If the peer device presents other alternate modes, the discover identity/mode response with other alternate modes is presented to the EC for handling. Refer to sections **3.4.1** and **3.4.2** for details on the registers used by the EC to send alternate mode related commands to the CCG3/4 device. Refer to sections **3.4.4** and **3.4.5** for details on the alternate mode related events sent by the CCG3/4 device to the EC.

The configuration table can be used to specify whether the CCG3/4 device should automatically discover and enter alternate modes or wait for EC intervention. The EC intervention is done through the ALT_MODE_CMD register. The EC can discover the alternate modes supported by the port partner using the VDM command tunneling supported by the CCG3/4 device. Once the SVID corresponding to the desired alternate mode has been discovered, the EC can use the ALT_MODE_CMD infrastructure to trigger entry into the corresponding alternate mode. The following alternate mode commands are supported by the CCG3/4 device, as described in section **3.4.1**:

- Enable EC Trigger of Alternate Mode: This command is used to inform the CCG3/4 device to wait for the EC to trigger alternate mode entry instead of doing so automatically.
- Disable EC Trigger of Alternate Mode: This command is used to ask the CCG3/4 device to resume automatic discovery of and entry into alternate modes without waiting for EC control.
- Alternate Mode Entry: This command is used to initiate entry into the desired alternate mode. This command is supported only when EC triggering of alternate modes is enabled.
- Alternate Mode Exit: This command is used to initiate exit from the desired alternate mode. This command is supported only when EC triggering of alternate modes is enabled.
- Alternate Mode Specific command: This is an extension that allows the EC to send commands that are specific to particular alternate modes such as DisplayPort or Thunderbolt.

4.3.4.1 DisplayPort Alternate Mode

DisplayPort is the primary alternate mode that is completely implemented in the CCG3/4 device's firmware. A set of DisplayPort specific events and commands is defined to allow the EC to have full control over the DisplayPort configuration.

4.3.4.2 DisplayPort Specific Events

Section **3.4.1** describes the structure of the alternate mode events sent by the CCG3/4 device firmware. The event type value of alternate mode (0x0C) is used to send DisplayPort specific events to the EC. In this case, the actual event information is provided through bytes 7:4 of the event data.

As shown in **Table 47**, two DisplayPort specific events are reported by the CCG3/4 device:

- The DisplayPort Pin Configuration event provides information on the pin configurations that are supported by the port partner so that the EC can select a proper value from the supported ones. Read of this event is useful in the scenario when notebooks disconnect power to the data mux in sleep states, but CCG4 is powered and VBUS is also available if a Type-C device is connected to the Type-C port. In this scenario, if CCG3/4 tries to configure the data mux, the mux does not get configured as it does not have any power. The EC needs to reconfigure the mux when the notebook is back to an active powered state. It needs to read this event and configure the APP_HW_CMD register based on the desired mux configuration.
- The DisplayPort Status Update event is used to notify the EC whenever there is a status update or attention message from the DP sink device.

4.3.4.3 DisplayPort Specific Commands

Section **3.4.1** describes the alternate mode (such as DisplayPort, Thunderbolt) related commands that can be initiated by the EC. Each alternate mode has specific commands, which are sent by the EC to control the particular alternate mode. The alternate mode specific command with Command ID option 5 is used to send commands specific to the DisplayPort alternate mode. The following DisplayPort specific commands are supported:

1. EC DP Control: This command is used to enable/disable EC control of the DisplayPort configuration. The EC needs to send this command (all 8 bytes of data as given below) to the write the data memory of the CCG3/4 device as shown in **Figure 8** over HPI to control the DisplayPort alternate mode.
    a. To enable EC control of the DP configuration, the following command format should be used:

        Byte 0: 0x0B (Alternate Mode Specific Command to CCG3/4 DFP)

        Byte 1: 0x00 (DisplayPort related)

        Bytes 3:2: 0xFF01 (DisplayPort SVID)

        Byte 4: 0x01 (Enable EC control)

        Bytes 6:5: 0x0000 (Reserved)

        Byte 7: 0x01 (EC DP Control command)

    b. To disable EC control of DisplayPort configuration, the following command format should be used:

Byte 0: 0x0B (Alternate Mode Specific Command to CCG3/4 DFP)

Byte 1: 0x00 (DP related)

Bytes 3:2: 0xFF01 (DisplayPort SVID)

Byte 4: 0x00 (Disable EC control)

Bytes 6:5: 0x0000 (Reserved)

Byte 7: 0x01 (EC DP Control command)

2. Select DP Configuration Command: This command is used to select the desired DisplayPort configuration and to send the DP Configure VDM. The command format is as follows:

   Byte 0: 0x0B (Alternate Mode Specific Command to CCG3/4 DFP)

   Byte 1: 0x00 (DP related)

   Bytes 3:2: 0xFF01 (DisplayPort SVID)

   Byte 4: Desired MUX Configuration (0 – Data MUX in isolate mode, 2 –Data MUX in USB3.1 connection, 3 – Data MUX in 2 lane DP+USB Connection, 4 – Data MUX in 4 lane DisplayPort connection)

   Bytes 6:5: 0x0000 (Reserved)

   Byte 7: 0x02 (DP Configure command)

Revision history

Document version	Date of release	Description of changes
**	2016-06-21	New application note
*A	2019-08-05	Updated template
*B	2021-03-08	Updated to Infineon template

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-03-08
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to www.cypress.com/support

Document reference
002-11898 Rev. *B

IMPORTANT NOTICE
The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.