Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

EZ-PD™ BCR Host Processor Interface Specification

Doc. No. 002-26784 Rev. **

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134
Phone (USA): 800.858.1810
Phone (International): +1.408.943.2600
www.cypress.com

Copyrights

© Cypress Semiconductor Corporation, 2019. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, EZ-PD, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

Contents

1. Introduction 5

1.1 Modes of operation 5
1.2 Introduction to HPI 5
1.3 HPI Capabilities 6

2. I2C Transport 7

2.1 I2C Write to BCR 7
2.2 I2C Read from BCR 7
2.3 Interrupt (Alert) pin from BCR 8

3. HPI Protocol 9

3.1 Types of Registers 9
3.2 Reading Status from BCR (Status Registers) 9
3.3 Command – Response Model (Command/Response Registers) 9
3.4 Asynchronous PD Messages and Events (Response Registers) 10
3.5 Flow Diagrams 10
3.6 Data Memory Accesses 10
3.7 Summary of Registers 10

4. Status Registers 12

4.1 DEVICE_MODE 12
4.2 SILICON_ID 12
4.3 INTERRUPT 12
4.4 PD_STATUS 13
4.5 TYPE_C_STATUS 14
4.6 BUS_VOLTAGE 14
4.7 CURRENT_PDO 15
4.8 CURRENT_RDO 15
4.9 SWAP_RESPONSE 15
4.10 EVENT_STATUS 16
4.11 READ_GPIO_LEVEL 16
4.12 SAMPLE_GPIO 17

5. Command Registers 18

5.1 RESET 18
5.2 EVENT_MASK 19
5.3 DM_CONTROL (Send PD Data Message) 20

5.4 SELECT_SINK_PDO 21
5.5 PD_CONTROL (Send PD Control Message) 22
5.6 REQUEST 23
5.7 SET_GPIO_MODE 24
5.8 SET_GPIO_LEVEL 24

6. Response Registers 25

6.1 DEV_RESPONSE 25
6.2 PD_RESPONSE 25
6.3 Response Codes 26

7. Data Memory 31

7.1 Data Memory Regions 31

Revision History 32

1. Introduction

EZ-PD™ Barrel Connector Replacement (BCR) is an easy to use Type-C and Power Delivery controller for electronic devices that consume power over the Type-C port. With a few resistor settings, you can customize EZ-PD™ BCR to match your design requirement, without having to write firmware or load configuration options over a protocol interface.

In addition to the resistor settings, the EZ-PD™ BCR device can also be connected to a microcontroller or processor that wants more information about the current state of the Type-C port or wants to dynamically change the power settings of the device.

The Host Processor Interface protocol over I2C is used for just this purpose.

1.1 Modes of operation

The EZ-PD™ BCR device can operate in one of two modes

1. Standalone mode
2. Host Processor Interface (HPI) mode

In the standalone mode, the EZ-PD™ BCR device uses the voltage and current settings specified using resistor dividers on its 4 configuration pins.

In HPI mode, the EZ-PD™ BCR device is controlled by an external I2C master which specifies the voltage and current options it requires at any point.

The Host Processor Interface is active in both modes, but certain write-able registers are only available in the HPI mode.

1.2 Introduction to HPI

The Host Processor Interface is a register-based protocol over an I2C slave interface. Frequencies upto 400kHz are supported and an interrupt line is provided as an alert indicator.

The connection between a system controller and EZ-PD™ BCR is as shown in Figure 1 below.

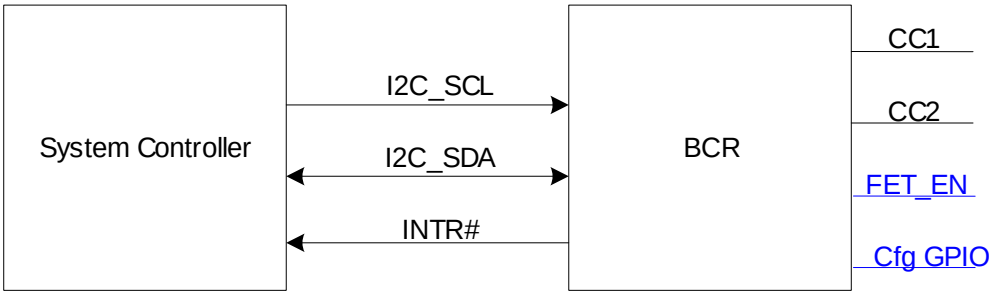


Figure 1: Connections between System Controller and EZ-PD™ BCR Device

1.3 HPI Capabilities

The main capabilities provided by CCG to the EC using HPI include:

- Reporting of Type-C and USB-PD connection status
- Interrupt based event reporting when connection status changes
- Control of USB-PD power profiles
- Ability to send and receive USB-PD Vendor Defined Messages (VDMs)
- Ability to initiate USB-PD control messages including resets, role swaps etc.

EZ-PD™ BCR provides these functionalities using commands, responses, events and asynchronous messages modeled as registers over the I2C slave interface.

A set of read-only **Status Registers** report the current state of the Type-C port and Power Delivery contract.

Command Registers provide commands to perform Type-C and PD interface related tasks. The system controller can send commands to EZ-PD™ BCR by writing to these registers. EZ-PD™ BCR processes these commands and responds back through the DEV_RESPONSE or PD_RESPONSE registers.

EZ-PD™ BCR also notifies the system controller about Type-C and PD specific events and messages through the PD_RESPONSE register.

The INTR line is used to notify the system controller about response, event and asynchronous messages.

In addition to the status, command and response registers; the HPI address map includes a 256-byte Data Memory space. This space is used as buffer area to exchange data associated with the commands, responses and events.

2. I2C Transport

The Host Processor Interface uses I2C as a transport mechanism to read and write to registers inside the device. The following sections describes how reads and writes are done.

2.1 I2C Write to BCR

I2C writes begin with a Start bit and the read preamble of 7-bit device address and the Write bit (0). BCR responds back with a 1-bit ACK/NAK response.

EZ-PD™ BCR uses 0x08 as the 7-bit device address.

The first two bytes following a write preamble is the address of the 16-bit register. The least significant byte (LSB) of register address is transferred first followed by the most significant byte. All bytes following the address bytes are the register data.

Figure 2 shows the bus sequence for a two-byte I2C write operation.

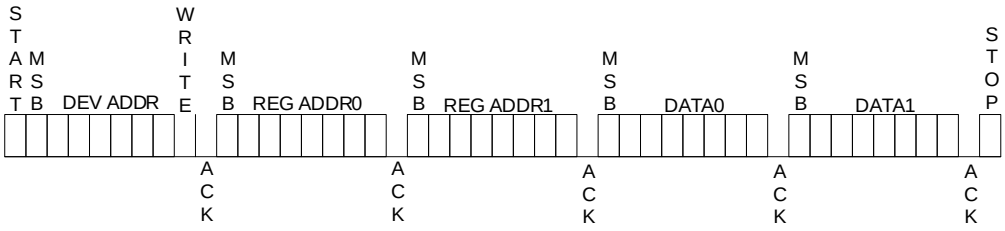


Figure 2: I2C Write Sequence

Writes to invalid register addresses are ignored.

2.2 I2C Read from BCR

I2C reads begin with a Start bit and write preamble where the system controller writes the 16-bit register address that it wants to read from.

The address is followed by another Start bit (i.e Restart) on the bus, instead of a Stop bit. The master then sends a read preamble with 7-bit device address and Read bit (0) and reads data from the device.

The maximum read size is 512 bytes. Reads beyond the valid address region will be NAK-ed by the BCR device. Figure 3 shows the I2C transfer sequence for a two-byte read operation.

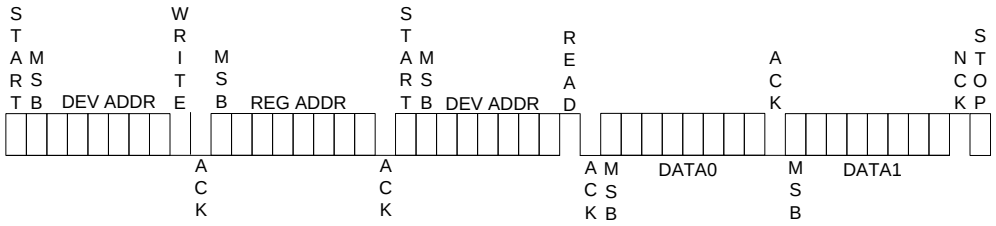


Figure 3: HPIv1 I2C Read Sequence

EZ-PD™ BCR will NAK the read preamble if the I2C master does not write a register address followed by Restart before reading register contents.

The I2C master on the system controller must support clock stretching. EZ-PD™ BCR will stretch the I2C clock in the following scenarios:

1. At the acknowledge (ACK/NAK) phase of the read or write preamble. EZ-PD™ BCR stretches the clock for 3~5 cycles before responding to the preamble sent by the I2C master.

2. EZ-PD™ BCR has an 8-byte FIFO to receive data sent by I2C master. The clock is stretched at the acknowledgement phase of every 4~8 data bytes as the FIFO gets filled up and the internal logic drains bytes to process them.

2.3 Interrupt (Alert) pin from BCR

The INTR pin is an active low output I/O that EZ-PD™ BCR drives to notify the system controller of responses, events and asynchronous messages. INTR is an open drain signal that can only be driven low. An external pull up (~5kohm) is required.

The cause of the interrupt can be detected by reading the interrupt status register (see INTERRUPT).

Two separate response registers are provided for general device events as well as events corresponding to the USB Type-C port. Separate interrupt status bits are provided for each of these event types.

The BCR device maintains an event queue. The INTR pin will be re-asserted until the system controller clears all events from the queue. BCR waits for 50us (after system controller clears the interrupt) before re-asserting INTR.

3. HPI Protocol

3.1 Types of Registers

EZ-PD™ BCR has four types of registers

1. Read-only **Status Registers**
2. Write-only **Command Registers**
3. Read-only **Response Registers**
4. Read/Write **Data Memory**

3.2 Reading Status from BCR (Status Registers)

The Status Registers are read-only and contain the state of the device, the Type-C port and the Power Delivery contract. A read of these registers will not trigger any assertion of INTR.

3.3 Command – Response Model (Command/Response Registers)

If the system controller wants to control any aspect of BCR, then it must write to one or more Command Registers.

The interaction model with these registers is as follows:

1. The system controller sends a command to BCR by writing to a Command Register
2. BCR executes the command and sends a response code back to the system controller through one of two Response Registers
3. If the response has any data associated with it, the data is returned through the data memory region. BCR uses the INTR# pin to notify EC that a response is available in the register
4. The system controller then clears the interrupt by writing to the INTERRUPT register. EZ-PD™ BCR then de-asserts the INTR pin

The response to a command is sent in either the DEV_RESPONSE register (for device-specific commands) or the PD_RESPONSE register (for port-specific commands). The INTERRUPT register contains separate status bits for each of these response types. And the INTR pin will be asserted if at least one of these response registers contain a message.

The system controller can identify the response register to be serviced and can clear only those specific bits in INTERRUPT.

EZ-PD™ BCR uses two queues to hold responses and other events to be sent to the system controller. These queues can hold a limited number of entries and can overflow if the system controller does not read out the data in time

EZ-PD™ BCR clears the contents of a specific Response Register only after EC writes to the corresponding bit in the INTERRUPT register. If the system controller sends a new command before reading the outstanding response to a previous command, the response to new command is queued in the internal message queue.

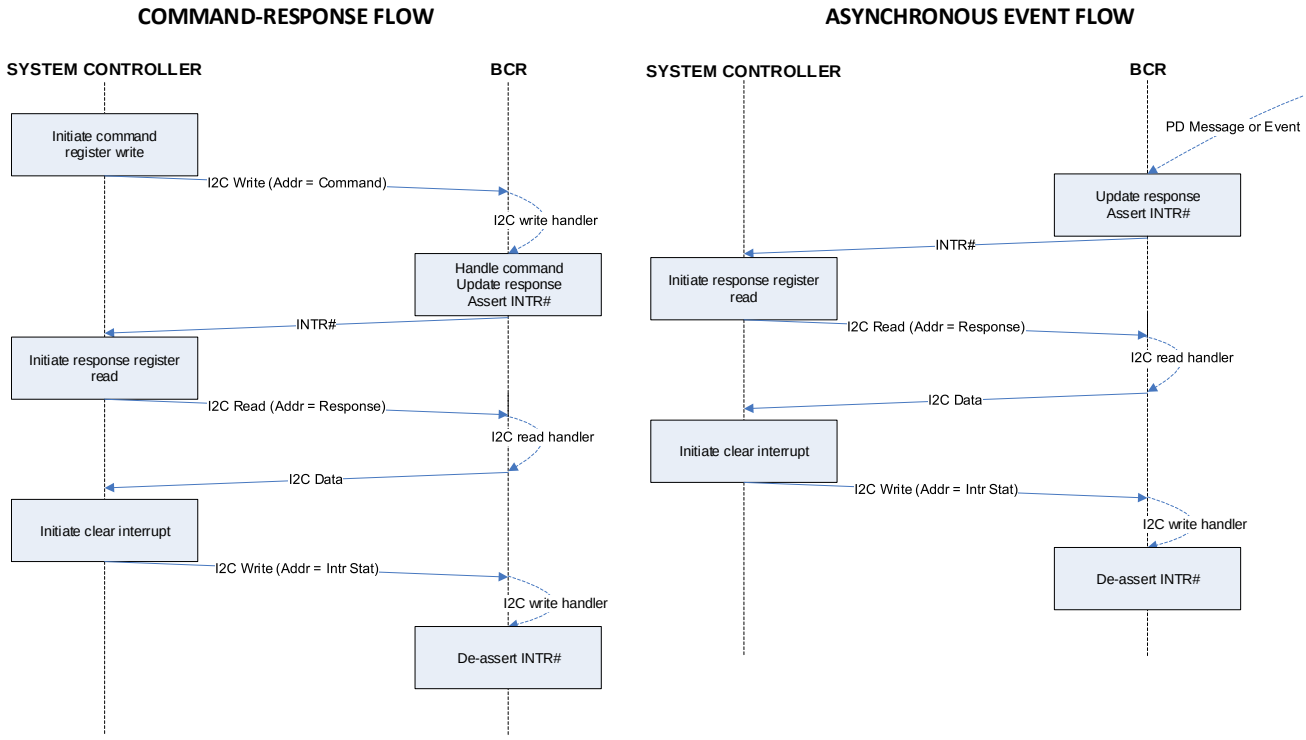
There can be a delay of up to 50 µsec from the time system controller clears the interrupt to the time the INTR signal is de-asserted. The system controller should take care to not sample the INTR signal in this period.

3.4 Asynchronous PD Messages and Events (Response Registers)

In addition to responses to commands received, the PD_RESPONSE register is also used by EZ-PD™ BCR to notify the system controller about asynchronous USB Power Delivery messages. BCR will assert the INTR pin when there is a new response or event stored in the response register. The system controller is expected to clear the interrupt after reading the response/event from the registers.

These asynchronous events can be enabled by writing to specific bits of the EVENT_MASK register.

3.5 Flow Diagrams



3.6 Data Memory Accesses

EZ-PD™ BCR supports separate read and write data memory regions (see Data Memory)

Any data memory access operation (valid or invalid) will not result in a response or event generation. If read operation crosses the data memory boundary, CCG will NAK the transfer. If write operation crosses the data memory boundary, CCG drops further data bytes written by the system controller. No action is taken based on the write transaction and data memory is not updated in that situation.

3.7 Summary of Registers

Here is a table of typical uses of HPI and the specific registers used to perform the action. Note that the list is not exhaustive, and more than one register may be used to perform the action.

#	PURPOSE	REGISTER
1	Reset the BCR device or it's I2C port	RESET
2	Check if a device is attached on the Type-C port	TYPE_C_STATUS
3	Read the status of PD contract Check if port is DFP or UFP	PD_STATUS
4	Get the voltage and current requested by BCR	CURRENT_PDO and CURRENT_RDO
5	Change BCR's response to DR_Swap and VCONN_Swap	SWAP_RESPONSE
6	Get changes to Type-C or PD port state	**Poll:** Read the EVENT_STATUS register for a list of changes since last read; write 1 to any bit to clear event **Interrupt:** Set events in EVENT_MASK and wait for interrupt. Then read PD_RESPONSE and clear interrupt.
7	Send a Vendor Defined Message and read the device's response to the VDM	**1:** Set the 'VDM Received' bit in EVENT_MASK Use DM_CONTROL to send the VDM **2:** Ensure PD_RESPONSE indicates SUCCESS **3:** Wait for 'VDM Received' event in PD_RESPONSE and read Data Memory
8	Read the VID/PID of the power adapter	**1:** Clear EVENT_STATUS **2:** If port data role is UFP (see #3), use PD_CONTROL to send a DR_Swap message. Ensure SUCCESS in PD_RESPONSE. Wait until "Data Role Swap Complete" is set in EVENT_STATUS **3:** Read "Current Data Role" from PD_STATUS **4a:** If role is still UFP, we cannot continue since power adapter doesn't support this feature **4b:** If role is DFP, perform procedure #7. Fill the "VDM" with "Discover Identity" request. Refer to the PD specification for message format.
9	Change the drive mode of GPIO_1 pin	SET_GPIO_MODE
10	Read logic level or voltage of GPIO_1	READ_GPIO_LEVEL and SAMPLE_GPIO
11	Read the voltage on VBUS line	BUS_VOLTAGE

4. Status Registers

4.1 DEVICE_MODE

This register indicates the active device mode. BCR always returns a constant byte from this register.

NAME	DEVICE_MODE
ADDRESS	0x0000
SIZE	1-byte

FIELD NAME	R/W	FIELD OFFSET	DESCRIPTION
Current Mode	RO	Byte 0	0x92 (BCR will always return this value when powered)

4.2 SILICON_ID

The system controller can use this register to read the Silicon ID of the device. The Silicon ID is fixed for BCR and will not change with the part.

This register can be used to identify if the HPI device is EZ-PD™ BCR or any other device in the EZ-PD™ CCG family from Cypress.

NAME	SILICON_ID
ADDRESS	0x0002
SIZE	2-bytes

FIELD NAME	R/W	FIELD OFFSET	DESCRIPTION
Silicon ID	RO	Bytes [0..1]	0x11B0 (BCR will always return this value when powered)

4.3 INTERRUPT

This register indicates the type of interrupt for which INTR pin was asserted.

NAME	INTERRUPT
ADDRESS	0x0006
SIZE	1-byte

FIELD NAME	R/W	FIELD OFFSET	DESCRIPTION
Interrupt Status	R/W1C	Byte 0	**Bit 0: Device Interrupt** 0 = No device interrupt pending 1 = New response available in DEV_RESPONSE register
			Bit 1: PD Port Interrupt 0 = No port interrupt pending 1 = New response available in PD_RESPONSE register
			Bits 2..7: Reserved (0x00)

4.4 PD_STATUS

This read-only status register reports the Default PD configuration of BCR and the current state of the PD contract.

NAME	PD_STATUS
ADDRESS	0x1008
SIZE	4-bytes

FIELD NAME	R/W	FIELD OFFSET	DESCRIPTION
Default Config	RO	Byte 0 [0..5]	**Bits 0..5: 0x00 (UFP Data Role and Sink Power Role)** This field indicates the default configuration supported by BCR
Current Config	RO	Byte 0 [6..7] Byte 1..3	**Bit 6: Current Port Data Role** 0 = UFP 1 = DFP This field is updated with the current data role when the Type-C port is attached to a partner device. The role can be changed by sending or receiving a DR_Swap message.
			Bit 7: Reserved (0)
			Bit 8: Current Port Power Role (0) 0 = Sink BCR always functions as a sink and so this bit is never expected to be set to 1.
			Bit 9: Reserved (0)
			Bit 10: Contract State 0 = No contract exists with port partner 1 = Explicit PD contract exists with port partner "Current Config" bits (Bits 6..31) in this register are valid only if this bit is set to 1. Use the CURRENT_PDO and CURRENT_RDO registers to fetch details of the contract.
			Bits 11..13: Reserved (0)
			Bit 14: Sink Tx Ready Status (Rp state) 0 = Rp is in SinkTxOk (Sink can send messages to Source) 1 = Rp is in SinkTxNG (Sink shouldn't send messages)
			Bit 15: Policy Engine State (see Chapter 8 of PD spec) 0 = Port is not in PE_SNK_Ready state 1 = Port is in PE_SNK_Ready state
			Bits 17..16: PD Spec revision supported by BCR 00 = PD 2.0 01 = PD 3.0 Others = Reserved
			Bit 18: Partner (Attached device) PD Spec revision 0 = Partner is PD 2.0 device 1 = Partner is PD 3.0 device
			Bit 19: Partner un-chunked extended message support status

			0 = Partner doesn't support un-chunked messages 1 = Partner supports un-chunked messages
			Bits 20..31: Reserved (0)

4.5 TYPE_C_STATUS

This read-only status register reports the status of the Type-C port on the BCR device.

NAME **ADDRESS** **SIZE**	TYPE_C_STATUS 0x100C 4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Current Type-C Status	RO	Byte 0	**Bit 0: Port Partner Connection Status** 0 = Port is not connected to partner 1 = Port is connected to partner Other bits in this register are valid only if this bit is set.
			Bit 1: CC Polarity 0 = CC1 1 = CC2
			Bits 4..2: Attached device type 000 = Nothing attached 010 = Source attached 011 = Debug Accessory attached *Other values = Reserved*
			Bit 5: Reserved (0)
			Bits 7..6: Type-C Current Level (Rp of partner device) 00 = Default (900mA) 01 = 1.5A 10 = 3A 11 = Reserved This is the Rp value advertised by the attached port partner. If the CC line voltage is invalid, then this field will indicate 00.

4.6 BUS_VOLTAGE

This read-only status register reports the live voltage on the VBUS supply for the specified port (in 100 mV units). The voltage will be measured using the internal ADC on the BCR device at the time of reading this register.

NAME **ADDRESS** **SIZE**	BUS_VOLTAGE 0x100D 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Voltage on the VBUS line	RO	Byte 0	**Voltage** Live VBUS voltage, in 100mV units

4.7 CURRENT_PDO

This read-only status register holds the active PDO requested by BCR from the attached power adapter. This field has the Voltage(s) and Maximum Current (or Power) advertised by the attached power adapter.

Refer to the Power Delivery specification for details on how to interpret the bits in this field.

NAME **ADDRESS** **SIZE**	CURRENT_PDO 0x1010 4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Current PDO	RO	Bytes 0..3	**Active PDO** This is the 32-bit Power Data Object (PDO) that was most recently received from the attached Source. Refer to Chapter 6.4 of the PD 2.0 or 3.0 specifications for defails on how to interpret the bits in a PDO.

4.8 CURRENT_RDO

This read-only status register reports the Request Data Object (RDO) which was used to establish the PD contract. This register contains the Maximum Operating Current and Operating Current values requested by BCR.

NAME **ADDRESS** **SIZE**	CURRENT_RDO 0x1014 4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Current RDO	RO	Bytes 0..3	**Active RDO** This is the 32-bit Request Data Object (PDO) that was most recently sent from BCR to the attached Source. Refer to Chapter 6.4 of the PD 2.0 or 3.0 specifications for defails on how to interpret the bits in an RDO.

4.9 SWAP_RESPONSE

By default, BCR will accept Data Role swaps and VCONN swaps and will reject Power Role swaps.

The runtime response of BCR to Data and VCONN swaps can be modified by modifying this register.

NAME **ADDRESS** **SIZE**	SWAP_RESPONSE 0x1028 2-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Swap Responses	R/W	Bytes 0..1	**Bits 0..1: DR_Swap Response** **Bits 2..3: Reserved (0)** **Bits 4..5: VCONN_Swap Response** **Bits 6..7: Reserved (0)**

			The response types are: 00 = Accept Swap 01 = Reject Swap 10 = Send Wait to the swap message 11 = Swap is not supported (in PD 2.0 contracts, BCR will send "Reject"; in PD 3.0 contracts, BCR will send "Not_Supported")

4.10 EVENT_STATUS

Instead of listening for events, the system controller can read this register at any time to know what has happened on the Type-C/PD port until then.

This is a "sticky" register i.e any bit set to 1 will remain so until the system controller clears it. The controller must write '1' to any bit to clear any event.

NAME **ADDRESS** **SIZE**	EVENT_STATUS 0x1044 4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Event Status	R/W1C	Bytes 0..3	Bit 0: Type-C Device Attached
			Bit 1: Type-C Device Disconnected
			Bit 2: PD Contract Negotiation Completed
			Bit 3: Power Role Swap Completed
			Bit 4: Data Role Swap Completed
			Bit 5: VCONN Swap Completed
			Bit 6: Hard Reset received
			Bit 7: Hard Reset sent
			Bit 8: Soft Reset sent
			Bit 9: Cable Reset sent
			Bit 10: Type-C Error Recovery initiated
			Bit 11: BCR Entered Source Disabled state (i.e Type-C-only charger detected)
			Bits 12..28: Reserved (0)
			Bit 29: Unexpected voltage on VBUS
			Bit 30: VBUS voltage is outside expected range
			Bit 31: Reserved (0)

4.11 READ_GPIO_LEVEL

This register is used to read the logic level of the GPIO_1 pin in BCR. The mode of the GPIO should be set to one of the digital drive modes (Values 1~7 in SET_GPIO_MODE) before reading this register. Else, this register will always contain 0x00.

NAME ADDRESS SIZE	READ_GPIO_LEVEL 0x0082 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Logic Level of GPIO	RO	Byte 0	**Logic Level** 0 = GPIO is being driven Low 1 = GPIO is being driven High

4.12 SAMPLE_GPIO

This register is used to sample the voltage on the GPIO_1 pin in BCR. The mode of the GPIO should be set to Analog mode (Value 0 in SET_GPIO_MODE) before reading this register. Else, this register will always contain 0x00.

NAME ADDRESS SIZE	SAMPLE_GPIO 0x0083 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Voltage Level on GPIO	RO	Byte 0	**Voltage Level** 8-bit voltage level referenced to VDDD, typically at 3.3V. So, 0V reads as 0x00 and VDDD (or higher) reads as 0xFF.

5. Command Registers

Command Registers used to trigger changes in the BCR device or in the Type-C (or PD) port state. Any write to a Command Register will result in INTR pin being asserted and a response populated in one of the Response Registers.

After writing to a command register, the system controller must wait for INTR pin to assert. The time taken to assert is dependent on the command being executed.

Once the response is read (from either DEV_RESPONSE or PD_RESPONSE), the system controller must clear the interrupt by writing to the corresponding bit in INTERRUPT.

5.1 RESET

This register is used to either reset the device or the I2C block.

When the I2C block is reset, the I2C module used for the HPI is reconfigured. All outstanding commands and responses are flushed.

When the device is reset, BCR undergoes a full chip reset causing the Power Delivery contract to be renegotiated and the sink FET switch to be opened. This will result in the system losing power if no alternative power source (such as a battery) is available.

NAME	RESET		
ADDRESS	0x0008		
SIZE	1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Signature	WO	Byte 0	**"R" (0x52)** A reset is executed only if the signature byte is ASCII 'R' (0x52)
Reset Type	WO	Byte 1	**Bit 0: Reset Type** 0 = I2C Reset 1 = Device Reset
			Bits 1..7: Reserved (0x00)

Responses in DEV_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** When the I2C block has been reset successfully
2. **RESET_COMPLETE (0x80):** When the device has been reset successfully
3. **INVALID_COMMAND (0x09):** When the signature is not 'R', or if any of the reserved bits are not '0'.

5.2 EVENT_MASK

This register will enable the set of events that BCR will notify the system controller about.

Whenever BCR detects an event (or PD message) for which a mask bit is set, it asserts the INTR pin and sets the “Port Interrupt” bit in INTERRUPT register.

The actual event code will be available in the PD_RESPONSE register and any associated data will be available in the Data Memory.

Upon device reset, all mask bits are set to 0.

Responses in DEV_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** Any write to EVENT_MASK will result in this response

NAME	EVENT_MASK
ADDRESS	0x1024
SIZE	4-bytes

FIELD NAME	R/W	FIELD OFFSET	DESCRIPTION
Event Filter	R/W	Bytes 0..3	**Bits 0..1: Reserved (0)**
			Bit 2: Over Voltage Detected Event
			Bit 3: Type-C Device Connected Event
			Bit 4: Type-C Device Disconnected Event
			Bit 5: PD Contract Negotiation Completed Event
			Bit 6: PD Control Message Received Event This event mask bit will enable the following events: - Data Role Swap completed - PS_RDY Message received - Accept Message received - Reject Message received - Wait Message received - Hard Reset received
			Bit 7: Vendor Defined Message (VDM) Received Event **Bit 8: Source Capabilities Message Received**
			Bit 9: Sink Capabilities Message Received
			Bit 10: Reserved
			Bit 11: Error and Timeout Events This event mask bit enables the following events - Hard Reset sent - Soft Reset sent - Source Disabled state entered (PD not detected) - SenderResponseTimer timeout - No response received for VDM - Unexpected voltage on VBUS - Type-C Error Recovery
			Bit 12: Reserved (0)
			Bit 13: Rp value change detected event

			Bits 16..14: Reserved (0)
			Bit 17: PD 3.0 Extended Data Message Received Event
			Bits 18..31: Reserved (0)

5.3 DM_CONTROL (Send PD Data Message)

The system controller can use this register to send various Power Delivery Data Messages to the Port Partner (SOP) or Type-C cable marker (SOP' or SOP'') devices.

This register is most typically used to send Vendor Defined Messages (VDM) to the port partner to perform design-specific operations.

To send any message, the system controller must follow these steps

1. Update the Data Memory with the complete PD Data message (excluding the Message Header, which will be prepended by BCR). The message should be written in 32-bit chunks in little endian order. When sending VDMs, the Data Memory must also contain the VDM header along with the rest of the actual message.
2. Write to the DM_CONTROL register with the Recipient Type (SOP=Port Partner, SOP*=Cable) and the size of the Data Message in bytes
3. BCR will attempt to send the message. If the message was sent properly or if any errors are seen, it will notify the system controller and the response code will be placed in PD_RESPONSE register.
4. If the port partner sends any response to the message sent by BCR, then BCR will notify the system controller separately. To listen for such partner messages, write to the corresponding bit in the EVENT_MASK register or poll the EVENT_STATUS register.

NAME **ADDRESS** **SIZE**	DM_CONTROL 0x1000 2-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
PD Data Message Send Settings	WO	Bytes 0..1	**Bits 1..0: Packet Recipient Type** 00 = SOP 01 = SOP' (Near End of Cable) 10 = SOP'' (Far End of Cable) Others = Reserved
			Bit 2: PD 3.0 Data Message Set this bit when sending Alert or Battery_Status messages
			Bit 3: Extended Data Message Set this bit when sending PD 3.0 Extended Data Messages
			Bit 4: Disable *SenderResponseTimer* Set this bit to not use the SenderResponseTimer. Refer to Section 6.2.2 of the PD spec for details.
			Bit 5: Reserved (0)
			Bits 6..7: Message Length Bits 9..8 **Bits 8..15: Message Length Bits 7..0** Length of the PD Data Message in bytes.

Responses in PD_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** If the Data Message was transmitted successfully (i.e GoodCRC was received from the recipient device)
2. **TRANSACTION_FAILED (0x0C):** If the transmission failed i.e GoodCRC was not received
3. **INVALID_COMMAND (0x09):** If any reserved bits are set
4. **PD_COMMAND_FAILED (0x0D):** If the Type-C port is not connected to a partner, or if a PD contract does not exist, or if the BCR device is not ready to send the message. Check the "PE_Ready" and "Sink Tx OK" bits of PD_STATUS register to know if BCR is ready to send a message.

5.4 SELECT_SINK_PDO

This register is used to selectively enable or replace Sink Capabilities of the BCR device. On power-up, BCR has two Sink Capabilities (or Sink PDOs):

PDO #1: Fixed Supply, 5V, 900mA, Higher_Capability = Yes if VBUS_MAX > 5V else No
PDO #2: Variable Supply, Minimum Voltage = VBUS_MIN, Maximum_Voltage = VBUS_MAX, Operational Current = ISNK_COARSE + ISNK_FINE

These PDOs are configured based on the resistor dividers on the 4 configuration pins on BCR.

The system controller can change the Sink PDOs in two ways

1. The system controller can enable/disable the two default PDOs. It does this by updating the PDO mask
2. The default Sink PDOs can be replaced with a custom list. To do this, the system controller must update the Data Memory with the PDO list and write to the SELECT_SINK_PDO register for the new PDOs to take effect.

Note: Once the Sink PDOs are updated, BCR will re-negotiate the PD contract with the power adapter. This will cause momentary power loss to the system.

To replace the default sink PDOs, the system controller must write the following packet into Data Memory and then write to the SELECT_SINK_PDO register.

1. Byte 0..3: ASCII string "SNKP" (i.e 0x50 0x4B 0x4E 0x53)
2. Byte 4..7: Sink PDO #1
3. Byte 8..11: Sink PDO #2
4. Byte 12..15: Sink PDO #3
5. Byte 16..19: Sink PDO #4
6. Byte 20..23: Sink PDO #5
7. Byte 24..27: Sink PDO #6
8. Byte 28..31: Sink PDO #7

If the system controller wants to use fewer than 7 PDOs, then it must set the remaining bytes to 0.

Responses in PD_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** If the Sink PDOs are updated successfully
2. **INVALID_ARGUMENT (0x09):** If mask is 0 or a non-existent PDO is enabled

NAME ADDRESS SIZE	SELECT_SINK_PDO 0x1005 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Sink PDO Mask	WO	Byte 0	**Bit 0: Enable PDO 1** **Bit 1: Enable PDO 2** **Bit 2: Enable PDO 3** **Bit 3: Enable PDO 4** **Bit 4: Enable PDO 5** **Bit 5: Enable PDO 6** **Bit 6: Enable PDO 7** **Bit 7: Set the "Unconstrained Power" bit in PDO 1** Once this register is written to, BCR will check if the first 4 bytes of Data Memory has the "SNKP" signature. If signature is present, it updates the Sink PDO list and uses the mask as specified in Bits 0..6. If signature is not present, it enables PDOs selected by the mask in Bits 0..6. If all bits are 0x00 then BCR will fall back to the default Sink PDOs as determined by the 4 configuration pins.

5.5 PD_CONTROL (Send PD Control Message)

This register is used to send Power Delivery Control messages to the port partner.

The port partner may respond back to a PD Control message sent using this register. To look for these responses, poll the EVENT_STATUS register or enable the "PD Control Message Received" event in EVENT_MASK and wait for an interrupt from BCR.

NAME ADDRESS SIZE	PD_CONTROL 0x1006 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Command	WO	Byte 0	**0x00~0x04: Reserved**
			0x05: Send DR_Swap message BCR will attempt to send DR_Swap message and exchange data roles
			0x06~0x08: Reserved
			0x09: Send VCONN_Swap message BCR will attempt to send VCONN_Swap message and exchange VCONN source direction
			0x0A: Send Get_Source_Cap message BCR will attempt to retrieve the Port Partner's Source Capabilities. **Note**: This will cause PD contract to be renegotiated resulting in a power loss to the system

			0x0B~0x0C: Reserved
			0x0D: Send Hard_Reset **Note**: This will cause PD contract to be renegotiated resulting in a power loss to the system
			0x0E: Send Soft_Reset message **Note**: This may cause PD contract to be renegotiated resulting in a power loss to the system
			0x0F~0x15: Reserved
			0x16: Send Get_Source_Cap_Extd message
			0x17: Send Get_Status message
			0x18: Send Not_Supported message
			0x1C~0xFF: Reserved

Responses in PD_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** If the PD Control message was sent successfully i.e GoodCRC was received from the port partner
2. **TRANSACTION_FAILED (0x0C):** If the transmission failed i.e GoodCRC was not received
3. **INVALID_COMMAND (0x09):** If the command is any reserved value
4. **PD_COMMAND_FAILED (0x0D):** If the Type-C port is not connected to a partner, or if a PD contract does not exist, or if the BCR device is not ready to send the message. Check the "PE_Ready" and "Sink Tx OK" bits of PD_STATUS register to know if BCR is ready to send a message.

5.6 REQUEST

This register is used to send a custom Request Data Object (RDO) to the port partner.

A write to this register will take effect only if the Sink PDO mask (in SELECT_SINK_PDO) is non-zero. This requirement is made so that the system controller can fully determine the state of the PD contract.

When the system controller writes to this register, BCR sends the request as-is to the port partner. BCR will not validate the contents of the packet. If the packet is malformed or if some bits are not set correctly, then the port partner may issue Soft Resets or Hard Resets to BCR causing the system to lose power.

The port partner may respond back to a PD Control message sent using this register. To look for these responses, poll the EVENT_STATUS register or enable the "PD Control Message Received" event in EVENT_MASK and wait for an interrupt from BCR.

NAME	REQUEST		
ADDRESS	0x1050		
SIZE	4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Request Data Object	WO	Byte 0..3	**32-bit Request Data Object** Contents of the RDO to be sent out, in little endian order

Responses in PD_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** Request message was queued to be sent to the port partner. A SUCCESS does not guarantee that GoodCRC was received, nor does it guarantee successful PD Contract renegotiation. Look for events in the EVENT_STATUS register or listen to events using EVENT_MASK for more details.
2. **COMMAND_FAILED (0x06):** If the Request message could not be sent. Try again later.

5.7 SET_GPIO_MODE

This register is used to change the mode of the GPIO_1 pin in BCR.

NAME **ADDRESS** **SIZE**	SET_GPIO_MODE 0x0080 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
GPIO Mode	WO	Byte 0	**Drive Mode** 0 = Analog Input (for ADC usage) 1 = High-Impedance Digital Input (external driver on the pin) 2 = Resistive Pull-up (Strong Low, Weak High) 3 = Resistive Pull-down (Weak Low, String High) 4 = Open-drain Drive Low 5 = Open-drain Drive High 6 = Strong drive (String Low, String High) 7 = Resistive drive (Weak Low, Weak High) With “Analog Input”, use SAMPLE_GPIO to read the voltage on the pin. With other digital drive modes, use READ_GPIO_LEVEL and SET_GPIO_LEVEL to read or set the logic level on the pin

Responses in DEV_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** The GPIO drive mode was changed successfully

5.8 SET_GPIO_LEVEL

This register is used to change the drive level of the GPIO_1 pin in BCR.

NAME **ADDRESS** **SIZE**	SET_GPIO_LEVEL 0x0081 1-byte		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
GPIO Mode	WO	Byte 0	**Drive Level** 0 = Drive Low 1 = Drive High

Responses in DEV_RESPONSE after writing to this register:

1. **SUCCESS (0x02):** The GPIO drive level was changed successfully
2. **INVALID_ARGUMENT (0x09):** The GPIO drive mode (set by SET_GPIO_MODE) was “Analog”. Change the mode to one of the digital drives before changing its level

6. Response Registers

6.1 DEV_RESPONSE

The Device Response (DEV_RESPONSE) register is used to store responses to commands sent on certain Command Registers. In addition, this register will also hold device events such as reset completion.

NAME **ADDRESS** **SIZE**	DEV_RESPONSE 0x007E 2-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Response Code	RO	Byte 0	**Bit 7: Type of Response** 0 = Response to a Command Register write 1 = Asynchronous Event generated by BCR
			Bits 6..0: Response Code See Response Codes for a list of codes
Length	RO	Byte 1	**Length of the response (in bytes)** If this field is non-zero, the Data Memory will be filled with data up to this size.

6.2 PD_RESPONSE

The Port Response (PD_RESPONSE) register is used to store responses to commands sent on Command Registers that trigger port state changes or send PD messages.

This register will also hold any PD message responses or asynchronous PD/Type-C events.

NAME **ADDRESS** **SIZE**	PD_RESPONSE 0x1400 4-bytes		
FIELD NAME	**R/W**	**FIELD OFFSET**	**DESCRIPTION**
Response Code	RO	Byte 0	**Bit 7: Type of Response** 0 = Response to a Command Register write 1 = Asynchronous Event generated by BCR
			Bits 6..0: Response Code See Response Codes for a list of codes
Length1	RO	Byte 1	**Length of the response (in bytes) if length < 256**

			If this field is non-zero, the Data Memory will be filled with data up to this size.
Length2	RO	Byte 2..3	**Length of the response (in bytes) if length ≥ 256** If this field is non-zero, the Data Memory will be filled with data up to this size. If length of the response is less than 256, then Length1 is the same as Length2. If length of the response is more than 256 bytes, then Length2 > Length1.

6.3 Response Codes

BCR can send 2 types of responses

1. **Responses to Command Registers**
   This can be of two types: responses to device-specific commands and responses to port-specific commands
2. **Asynchronous events**
   This can also be of two types: events generated by device state change or events generated when PD port state changes

Hence, we list 4 tables for each type of response.

Note that any response code not listed here is reserved for future use. In addition, only asynchronous port events contain additional data which the system controller can read. These are documented as an additional column in that table.

TYPE	Responses to Commands	
RESPONSE NAME	**CODE**	**DESCRIPTION**
No Response	0x00	**No Response** No outstanding command or event in BCR. Or BCR is processing a command that will take a long time to complete.
Success	0x02	**Success** Command was handled successfully. Refer to the specific Command Register definition to understand what a successful handling of command means.
Invalid Command or Argument	0x05, 0x09	**Invalid Command or Argument** Partial register writes, reserved bits set, unexpected command code or unexpected command sizes.
Not Supported	0x0A	**Command Not Supported in mode** Command is not supported in the current mode
Transaction Failed	0x0C	**Transaction Failed** The PD message was not sent successfully 1. GoodCRC was not received in response to BCR sending the command.

		2. BCR could not send the message due to a packet collision 3. Message was invalid or formatted incorrectly If the system controller receives this response, it is expected to poll the "PE_Ready" and "SinkTxOk" bits in PD_STATUS and retry when these two bits indicate transmission readiness.
PD Command Failed	0x0D	**PD Command Failed** BCR was unable to send the command because the port wasn't connected or if BCR was handling another PD command. If the system controller receives this response, it is expected to poll the "PE_Ready" and "SinkTxOk" bits in PD_STATUS and retry when these two bits indicate transmission readiness.
Port Busy	0x12	**PD Port Busy** The command wasn't sent because the PD PHY was busy.

TYPE	Asynchronous Device Events	
RESPONSE NAME	**CODE**	**DESCRIPTION**
Reset Complete	0x80	**Reset Completed Event** BCR completed a power-on reset or I2C initiated reset and is back in operational mode.
Message Queue Overflow	0x81	**Message Queue Overflow** The number of messages in the message queue exceeded capacity and further responses/events have been dropped. When the system controller detects this event, it must flush the queue by repeatedly clearing interrupts in the INTERRUPT register until there are no more interrupts.
Over-Voltage detected on VBUS	0x83	**VBUS Voltage is outside expected range** When VBUS is out of the (VBUS_MIN, VBUS_MAX) range, the OVP event is triggered. The Sink FET is turned off to protect the system from excess voltage.

TYPE	Asynchronous Port Events		
RESPONSE NAME	**CODE**	**DESCRIPTION**	**DATA SENT WITH RESPONSE**
Type-C Port Connected	0x84	**Type-C Attach Detected** A new device was attached. BCR detected the device and entered the Attached.SNK state in the Type-C Sink State machine.	None
Type-C Port Disconnected	0x85	**Type-C Detach Detected** Device was detached and BCR entered the Unattached.SNK state	None

PD Contract Negotiation Complete	0x86	**PD Contract Negotiation Complete** BCR finished the process of negotiating a power contract with the attached adapter. The contract may have succeeded or failed. The result is available in the 8-bytes of response data sent along with this response.	**8-bytes in Data Memory** **Bit 0:** Set if PD contract negotiation is successful **Bit 1:** Set if BCR sets "Capability Mismatch" in the Request message. **Bit 2..4: Reason for PD Contract Negotiation Failure:** 011 = Power adapter rejected BCR's request and there was a previous explicit PD contract 100 = Power adapter rejected BCR's request and there was no explicit PD contract 101 = Power adapter accepted BCR's request but didn't send PS_RDY Others = Reserved **Bits 7..5:** Reserved (0) **Bytes 1..3**: Reserved (0) **Bytes 4..7: Request Data Object sent by BCR to Power Adapter**
Swap Completed	0x87	**PD Swap Completed** A data role swap or VCONN direction swap was completed. The process may have succeeded or failed. Read the response data in Data Memory for details.	**8-bytes in Data Memory** **Bit 0:** Set if PD contract negotiation is successful **Bit 1:** Set if BCR sets "Capability Mismatch" in the Request message. **Bit 2..4: Reason for PD Contract Negotiation Failure:** 011 = Power adapter rejected BCR's request and there was a previous explicit PD contract 100 = Power adapter rejected BCR's request and there was no explicit PD contract 101 = Power adapter accepted BCR's request but didn't send PS_RDY Others = Reserved **Bits 7..5:** Reserved (0) **Bytes 1..3**: Reserved (0) **Bytes 4..7: Request Data Object sent by BCR to Power Adapter**
PS_RDY	0x8A	**PS_RDY Message Received**	None
GotoMin	0x8B	**GotoMin Message Received** Power adapter requested the system to enter the lower current mode	None
Accept	0x8C	**Accept Message Received**	None
Reject	0x8D	**Reject Message Received**	None

Wait	0x8E	**Wait Message Received**	None
Hard_Reset	0x8F	**Hard Reset Received**	None
VDM Received	0x90	**PD VDM (Vendor Defined Message) Received** BCR responds automatically to certain Structured VDMs defined in the PD spec. The system controller can respond to Unstructured VDMs and commands associated with alternate modes if necessary. Note that if a response will be sent, it must be done within tVDMSenderResponse time as defined in the PD spec. The Data Memory has the actual VDM message received from the port partner.	**Length = 4 + 4 * Number_of_VDOs** **Byte 1..0:** Message Header **Byte 2:** SOP type (0=SOP, 1=SOP', 2=SOP'') **Byte 3:** Reserved (0) **Byte 4..7:** VDM header **Byte 8..N:** VDM data
Source Capabilities	0x91	**Source Capabilities Received** The port partner sent a new set of Source Capabilities message. This will cause the PD contract to be re-negotiated.	**Length = 4 + 4 * Number_of_PDOs** **Byte 1..0:** Message Header **Byte 2:** SOP type (0) **Byte 3:** Reserved (0) **Byte 4..N:** Source PDOs received from port partner
Sink Capabilities	0x92	**Sink Capabilities Received** Port partner sent sink capabilities message.	**Length = 4 + 4 * Number_of_PDOs** **Byte 1..0:** Message Header **Byte 2:** SOP type (0) **Byte 3:** Reserved (0) **Byte 4..N:** Sink PDOs received from port partner
Hard Reset Sent	0x9A	**Hard Reset was sent to port partner** BCR sends Hard Resets in response to certain events. Refer to the PD spec for details.	None
Soft Reset Sent	0x9B	**Soft Reset was sent to port partner** BCR sends Soft Resets when certain Atomic Message Sequences are interrupted or under certain other conditions. Refer to the PD spec for details.	None
Source Disabled	0x9D	***SourceDisabled* state entered** BCR enters this state if it doesn't detect Power Delivery messages from the power adapter. The attached adapter may be Type-C only or a legacy charger.	None

No VDM Response Received	0x9F	**No VDM Response Received** BCR reports this error when a VDM was sent by system controller but port partner did not send a response back to BCR.	None
Type-C Error Recovery	0xA1	**Type-C Error Recovery** BCR entered the Type-C Error Recovery State. Refer to the Type-C Connector Specification for details on this state.	None
Battery Status Received	0xA2	**Battery_Status message received**	None
Alert Received	0xA3	**Alert message received**	None
Not Supported received	0xA4	**Not_Supported message received**	None
Rp change detected	0xAA	**Rp change detected**	None
Extended Data Message Received	0xAC	**PD 3.0 Extended Data Message Received**	**Length = 4 + Extended message length** **Byte 1..0:** Message Header **Byte 3..2:** Extended Message Header **Byte 4..N:** Data from the extended message
CC OVP	0xBA	**CC Over-voltage detected** Voltage on CC exceeded expected range. This may be due to CC shorting to VBUS.	None

7. Data Memory

BCR has a 512-byte Data Memory region split into two portions, for writes and reads. Any command that requires data to be sent or received will use the data memory region.

7.1 Data Memory Regions

The Data Memory is split into two regions: Read Data Memory and Write Data Memory.

DATA MEMORY REGION	ACCESS	ADDRESS RANGE	DESCRIPTION
Read Data Memory	RO	**0x1404 to 0x150B**	The 264-byte Read Data Memory is placed adjacent to the PD_RESPONSE register so that it can be read in one single I2C read. This region will contain the PD Message data, contract information and other information as listed in the Response Codes section.
Write Data Memory	WO	**0x1800 to 0x19FF**	The 512-byte Write Data Memory is used when writing to Command Registers to send additional data for the command.

Any write or read from a Data Memory region will not trigger an INTR assertion. Reads or writes beyond the expected address range will be NAKed.

Revision History

Document Revision History

Document Title: EZ-PD™ BCR Host Processor Interface Specification Document Number: 002-26784			
Revision	**Issue Date**	**Orifgin of Change**	**Description of Change**
**	03/13/2019	KUMR	New BCR HPI Guide