Mailbox Client Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.1**

IP Version: **20.0.2**

Subscribe

Send Feedback

UG-20087 | 2021.03.29

Latest document on the web: **PDF** | **HTML**

Contents

1. Mailbox Client Intel FPGA IP User Guide

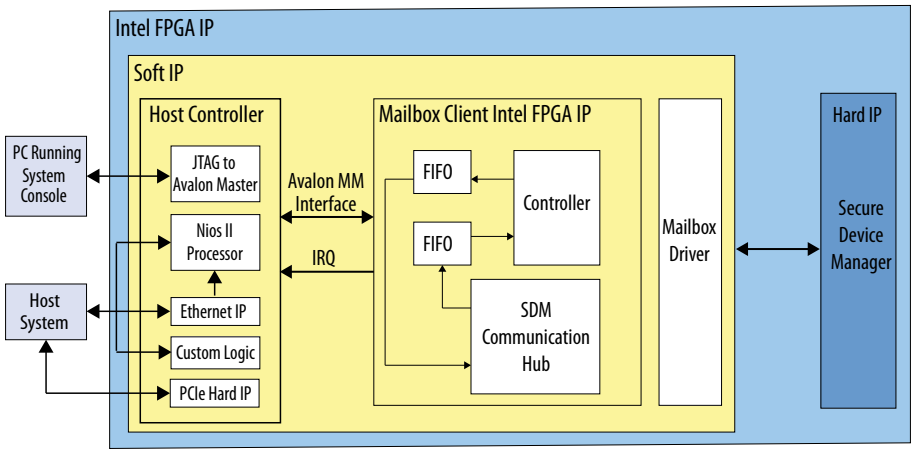
The Mailbox Client Intel FPGA IP is a bridge between a host and the Intel® Stratix® 10 secure device manager (SDM). You use the Mailbox Client Intel FPGA IP to send commands and receive status from SDM peripheral clients. The Mailbox Client defines functions that the SDM runs.

The following pre-defined functions are available:

- Reading the Chip ID
- Reading temperature sensors
- Reading voltage sensors
- Reading and writing external quad serial peripheral interface (SPI) flash memory
- Performing remote system updates (RSU)

The following block diagram shows how to use the Mailbox Client Intel FPGA IP in an interactive session. The diagram also emphasizes different ways of communicating with IP through the Host Controller.

Figure 1. Mailbox Client Intel FPGA IP System Block Diagram



This block diagram includes the following components:

- Host Controller: provides possible ways of accessing the Mailbox Client Intel FPGA IP. Use any of the specified ways to communicate with the host controller:
  - System Console with the JTAG to Avalon® Master Bridge Intel FPGA IP. The System Console provides a Tcl Console pane that you can use to run the IP functions. The JTAG to Avalon Master Bridge Intel FPGA IP translates the commands it receives from the System Console to Avalon Memory-Mapped (Avalon MM) format that the Mailbox Client Intel FPGA IP requires.
  - Nios® II processor: sends commands to the Mailbox Client Intel FPGA IP.
  - Custom logic: It sends commands to the Mailbox Client Intel FPGA IP.
  - PCIe* Hard IP
  - Ethernet IP
- Mailbox Client Intel FPGA IP: drives commands and receives responses from the SDM. This component includes FIFOs with a maximum depth of 1024 entries to store commands and responses. The Mailbox Client Intel FPGA IP interrupt indicates when the input FIFO in full and when the output FIFO contains valid data. You can size these FIFOs to accommodate the commands the you intend to send.

Intel provides a reference design that uses the System Console and the JTAG master to drive the Mailbox Client Intel FPGA IP. In the Intel Design Store, search for *Intel Agilex™ Mailbox Client Intel FPGA IP Core Design Example (QSPI flash Access and Remote System Update)* to view the design.

Related Information

- Avalon Interface Specifications
- Secure Device Manager in Intel Stratix 10 Devices
- Operation Commands on page 6
- Analyzing and Debugging Designs with System Console
- Agilex Mailbox Client Intel FPGA IP Core Design Example (QSPI flash Access and Remote System Update)

1.1. Device Family Support

The following lists the device support level definitions for Intel FPGA IPs:

- **Advance support** — The IP is available for simulation and compilation for this device family. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- **Preliminary support** — The IP is verified with preliminary timing models for this device family. The IP meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.
- **Final support** — The IP is verified with final timing models for this device family. The IP meets all functional and timing requirements for the device family and can be used in production designs.

Table 1. Device Family Support

Device Family	Support
Intel Stratix 10	Final
Intel Agilex	Advance

Related Information

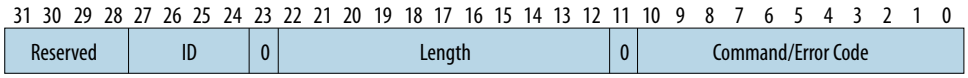
- Mailbox Client Intel FPGA IP Release Notes
- Mailbox Client with Avalon Streaming Interface Intel FPGA IP Release Notes

1.2. Commands and Responses

The host controller communicates with the SDM using command and response packets via the Mailbox Client Intel FPGA IP.

The first word of the command and response packets is a header that provides basic information about the command or response.

Figure 2. Command and Response Header Format



Note: The LENGTH field in the command header must match the command length of corresponding command.

The following table describes the fields of the header command.

Table 2. Command and Response Header Description

Header	Bit	Description
Reserved	[31:28]	Reserved.
ID	[27:24]	The command ID. The response header returns the ID specified in the command header. Refer to *Operation Commands* for command descriptions.
0	[23]	Reserved.
LENGTH	[22:12]	Number of words of arguments following the header. The IP responds with an error if a wrong number of words of arguments is entered for a given command.
Reserved	[11]	Reserved. Must be set to 0.
Command Code/Error Code	[10:0]	Command Code specifies the command. The Error Code indicates whether the command succeeded or failed. In the command header, these bits represent command code. In the response header, these bits represent error code. If the command succeeds, the Error Code is 0. If the command fails, refer to the error codes defined in the *Error Code Responses*.

1.2.1. Operation Commands

Resetting Quad SPI Flash

Important: For Intel Stratix 10 devices, do not reset the quad SPI flash when used as the configuration device and data storage device with FPGA. Resetting the quad SPI flash during the FPGA configuration and reconfiguration, or in the QSPI's READ/WRITE/ERASE operations, causes undefined behavior for quad SPI flash and the FPGA. To recover from the unresponsive behavior, you must power cycle your device. To reset the quad SPI flash via the external host, you must first complete the FPGA configuration and reconfiguration, or a quad SPI operation, and only then toggle the reset. The quad SPI operation is complete when the exclusive access to the quad SPI flash is closed by issuing the QSPI_CLOSE command via the Mailbox Client Intel FPGA IP or CLOSE command via the Serial Flash Mailbox Client Intel FPGA IP.

Important: For Intel Agilex devices, you must connect the serial flash or quad SPI flash reset pin to the AS_nRST pin. The SDM must fully control the QSPI reset. Do not connect the quad SPI reset pin to any external host.

Table 3. Command List and Description

Command	Code (Hex)	Command Length (1)	Response Length (1)	Description
NOOP	0	0	0	Sends an OK status response.
GET_IDCODE	10	0	1	The response contains one argument which is the JTAG IDCODE for the device
GET_CHIPID	12	0	2	The response contains 64-bit CHIPID value with the least significant word first.
GET_USERCODE	13	0	1	The response contains one argument which is the 32-bit JTAG USERCODE that the configuration bitstream writes to the device.
GET_VOLTAGE	18	1	1	The GET_VOLTAGE command has a single argument which is a bitmask specifying the channels to read. Bit 0 specifies channel 0, bit 1 specifies channel 1, and so on. The response includes a one-word argument for each bit set in the bitmask. The voltage returned is an unsigned fixed-point number with 16 bits below the binary point. For example, a voltage of 0.75V returns 0x0000C000.(2) (3)
GET_TEMPERATURE	19	1	n(4)	The GET_TEMPERATURE command returns the temperature or temperatures of the core fabric or transceiver channel locations you specify. For Intel Stratix 10 devices, use the sensor_req argument to specify the locations. The sensor_req includes the following fields:

continued...

(1) This number does not include the command or response header.

(2) Refer to *Intel Stratix 10 Analog to Digital Converter User Guide* for more information about reading voltage sensors on Intel Stratix 10 devices.

(3) Refer to *Intel Agilex Power Management User Guide for more information* about temperature sensor channels and locations.

(4) Index n depends on the number of sensor masks.

<table>
<tr><th>Command</th><th>Code (Hex)</th><th>Command Length (1)</th><th>Response Length (1)</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>• Bits[31:9]: Reserved.
• Bits[8:0]: <code>Sensor mask</code>. Specifies the TSD location.
The channels return the temperatures for the following locations:
• Channel 0: Samples the temperature from the core fabric.
• Channels 1- 6: Samples the temperature from the specified transceiver tile.
• Channels 7-8: Samples the temperature from the high-bandwidth DRAM memory (HBM2) stacks.
The temperature returned is a signed fixed value with 8 bits below the binary point. For example, a temperature of 10°C returns 0x00000A00. A of temperature -1.5°C returns 0xFFFFFE80.
If the bitmask specifies an invalid <code>Location</code>, the command returns an error code which is any value in the range 0x80000000 -0x800000FF.
For Intel Stratix 10 devices, refer to the Temperature Sensor Channels and Locations in the Intel Stratix 10 Analog to Digital Converter User Guide for more information about sensor locations.</td></tr>
<tr><td>RSU_IMAGE_ UPDATE</td><td>5C</td><td>2</td><td>0</td><td>Triggers reconfiguration from the data source that can be either the factory or an application image.
This command takes an optional 64-bit argument that specifies the reconfiguration data address in the flash. When sending the argument to the IP, you first send bits [31:0] followed by bits [63:32]. If you do not provide this argument its value is assumed to be 0.
• Bit [31:0]: The start address of an application image.
• Bit [63:32]: Reserved (write as 0).
Once the device processes this command, it returns the response header to response FIFO before it proceeds to reconfigure the device. Ensure the host PC or host controller stops servicing other interrupts and focuses on reading the response header data to indicate the command completed successfully. Otherwise, the host PC or host controller may not be able to receive the response once the reconfiguration process started.
Once the device proceeds with reconfiguration, the link between the external host and FPGA is lost. If you use PCIe in your design, you need to re-enumerate the PCIe link.
Important: When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.</td></tr>
<tr><td>RSU_GET_SPT</td><td>5A</td><td>0</td><td>4</td><td>RSU_GET_SPT retrieves the quad SPI flash location for the two sub-partition tables that the RSU uses: SPT0 and SPT1.
The 4-word response contains the following information:
<table>
<tr><th>Word</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>SPT0[63:32]</td><td rowspan="2">SPT0 address in quad SPI flash.</td></tr>
<tr><td>1</td><td>SPT0[31:0]</td></tr>
<tr><td>2</td><td>SPT1[63:32]</td><td rowspan="2">SPT1 address in quad SPI flash.</td></tr>
<tr><td>3</td><td>SPT1[31:0]</td></tr>
</table></td></tr>
<tr><td>CONFIG_ STATUS</td><td>4</td><td>0</td><td>6</td><td>Reports the status of the last reconfiguration. You can use this command to check the configuration status during and after configuration. The response contains the following information:
<table>
<tr><th>Word</th><th>Summary</th><th>Description</th></tr>
</table></td></tr>
</table>

continued...

(1) This number does not include the command or response header.

Command	Code (Hex)	Command Length [1]	Response Length [1]	Description		
				0	State	Describes the most recent configuration related error. Returns 0 when there are no configuration errors. The error field has 2 fields: • Upper 16 bits: Major error code. • Lower 16 bits: Minor error code. Refer to *Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions* in the *Mailbox Client Intel FPGA IP User Guide* for more information.
				1	Version	The version of the RSU data structure.
				2	Pin status	• Bit [31]: Current nSTATUS output value (active low) • Bit [30]: Detected nCONFIG input value (active low) • Bit [29:3]: Reserved • Bit [2:0]: The MSEL value at power up
				3	Soft function status	Contains the value of each of the soft functions, even if you have not assigned the function to an SDM pin. • Bit [31:6]: Reserved • Bit [5]: HPS_WARMRESET • Bit [4]: HPS_COLDRESET • Bit [3]: SEU_ERROR • Bit [2]: CVP_DONE • Bit [1]: INIT_DONE • Bit [0]: CONF_DONE
				4	Error location	Contains the error location. Returns 0 if there are no errors.
				5	Error details	Contains the error details. Returns 0 if there are no errors.
RSU_STATUS	5B	0	9	Reports the current remote system upgrade status. You can use this command to check the configuration status during configuration and after it has completed. This command returns the following responses:		
				Word	**Summary**	**Description**
				0-1	Current image	Flash offset of the currently running application image.
				2-3	Failing image	Flash offset of the highest priority failing application image. If multiple images are available in flash memory, stores the value of the first image that failed. A value of all 0s indicates no failing images. If there are no failing images, the remainder of the remaining words of the status information do not store valid information.

continued...

(1) This number does not include the command or response header.

Command	Code (Hex)	Command Length [1]	Response Length [1]	Description		
						Note: A rising edge on nCONFIG to reconfigure from ASx4, does not clear this field. Information about failing image only updates when the Mailbox Client receives a new RSU_IMAGE_UPDATE command and successfully configures from the update image.
				4	State	Failure code of the failing image. The error field has two parts: • Bit [31:16]: Major error code • Bit [15:0]: Minor error code Returns 0 for no failures. Refer to *Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions* in the *Mailbox Client Intel FPGA IP User Guide* for more information.
				5	Version	The version of the RSU software.
				6	Error location	Stores the error location of the failing image. Returns 0 for no errors.
				7	Error details	Stores the error details for the failing image. Returns 0 if there are no errors.
				8	Current image retry counter	Count of the number of retries that have been attempted for the current image. The counter is 0 initially. The counter is set to 1 after the first retry, then 2 after a second retry. Specify the maximum number of retries in your Intel Quartus® Prime Settings File (.qsf). The command is: `set_global_assignment -name RSU_MAX_RETRY_COUNT 3`. Valid values for the MAX_RETRY counter are 1-3. The actual number of available retries is `MAX_RETRY -1` This field was added in version 19.3 of the Intel Quartus Prime Pro Edition software.
RSU_NOTIFY	5D	1	0	Clears all error information in the RSU_STATUS response and resets the retry counter. The one-word argument has the following fields: • 0x00050000: Clear current reset retry counter. Resetting the current retry counter sets the counter back to zero, as if the current image was successfully loaded for the first time. • 0x00060000: Clear error status information. • All other values are reserved. This command is not available before version 19.3 of the Intel Quartus Prime Pro Edition software.		
QSPI_OPEN	32	0	0	Requests exclusive access to the quad SPI. You issue this request before any other QSPI requests. The SDM accepts the request if the quad SPI is not in use and the SDM is not configuring the device. Returns OK if the SDM grants access.		

continued...

(1) This number does not include the command or response header.

Command	Code (Hex)	Command Length [1]	Response Length [1]	Description
				Note: The SDM grants exclusive access to the client using this mailbox. Other clients cannot access the quad SPI until the active client relinquishes access using the QSPI_CLOSE command. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_CLOSE	33	0	0	Closes the exclusive access to the quad SPI interface. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_SET_CS	34	1	0	Specifies one of the attached quad SPI devices via the chip select lines. Takes a one-word argument as described below: • Bits[31:28]: Flash device to select. The value 4'b0000 selects the flash that corresponds to nCSO[0]. nCSO[0] is the only signal that the FPGA can use to access the quad SPI flash device. The HPS can use nCSO[3:0] to access HPS data. • Bits[27:0]: Reserved (write as 0). The HPS can use nCSO[3:1] to access 3 additional quad SPI devices. This command is optional for the AS x4 configuration scheme. Is required for all other configuration schemes. Access to the QSPI flash memory devices using SDM_IO pins is only available for the AS x4 configuration scheme, JTAG configuration, and a design compiled for ASx4 configuration. For the Avalon streaming interface (Avalon ST) configuration scheme, you must connect QSPI flash memories to GPIO pins. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_READ	3A	2	N	Reads the attached quad SPI device. The maximum transfer size is 4 kilobytes (KB) or 1024 words. Takes two arguments: • The quad SPI flash address (one word). The address must be word aligned. The device returns the 0x1 error code for non-aligned addresses. • Number of words to read (one word). When successful, returns OK followed by the read data from the quad SPI device. A failure response returns an error code. For a partially successful read, QSPI_READ may erroneously return the OK status. *Note:* You cannot run the QSPI_READ command while device configuration is in progress. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_WRITE	39	2+N	0	Writes data to the quad SPI device. The maximum transfer size is 4 kilobytes (KB) or 1024 words. Takes three arguments: • The flash address offset (one word). The write address must be word aligned. • The number of words to write (one word). • The data to be written (one or more words). A successful write returns the OK response code. To prepare memory for writes, use the QSPI_ERASE command before issuing this command. *Note:* You cannot run the QSPI_WRITE command while device configuration is in progress.

continued...

(1) This number does not include the command or response header.

Command	Code (Hex)	Command Length [1]	Response Length [1]	Description
				Important: When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_ERASE	38	2	0	Erases a sector of the quad SPI device. Takes two arguments: • The flash address offset to start the erase (one word). The address must be the start address of a sector within the flash memory; consequently, the address must be 64 KB aligned. Returns an error for non-64 KB aligned addresses. • The number of words to erase specified in multiples of 0x4000 words. A successful erase returns the OK response code. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_READ_DEVICE_REG	35	2	N	Reads registers from the quad SPI device. The maximum read is 8 bytes. Takes two arguments: • The opcode for the read command. • The number of bytes to read. A successful read returns the OK response code followed by the data read from the device. The read data return is in multiple of 4 bytes. If the bytes to read is not an exact multiple of 4 bytes, it is padded with multiple of 4 bytes until the next word boundary and the padded bit value is zero. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_WRITE_DEVICE_REG	36	2+N	0	Writes to registers of the quad SPI. The maximum write is 8 bytes. Takes three arguments: • The opcode for the write command. • The number of bytes to write. • The data to write. To perform a sector erase or sub-sector erase, you must specify the serial flash address in most significant byte (MSB) to least significant byte (LSB) order as the following example illustrates. To erase a sector of a Micron 2 gigabit (Gb) flash at address 0x04FF0000 using the QSPI_WRITE_DEVICE_REG command, write the flash address in MSB to LSB order as shown here: Header: 0x00003036 Opcode: 0x000000DC Number of bytes to write: 0x00000004 Flash address: 0x0000FF04 A successful write returns the OK response code. This command pads data that is not a multiple of 4 bytes to the next word boundary. The command pads the data with zero. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
QSPI_SEND_DEVICE_OP	37	1	0	Sends a command opcode to the quad SPI. Takes one argument: • The opcode to send the quad SPI device. A successful command returns the OK response code. *Important:* When resetting quad SPI, you must follow instructions specified in Resetting Quad SPI Flash on page 6.
REBOOT_HPS	47	0	0	Triggers the HPS cold reset. Firmware loads the new bitstream from the boot source based on MSEL settings. The loaded bitstream and your device must use the same firmware version. When loading the new bitstream, SDM wipes HPS and loads the HPS bootloader to HPS OCRAM and releases MPU.

(1) This number does not include the command or response header.

Command	Code (Hex)	Command Length (1)	Response Length (1)	Description
				Refer to *Intel Stratix 10 Hard Processor System Technical Reference Manual* and *Intel Stratix 10 SoC FPGA Boot User Guide* for more details on HPS reset.

For CONFIG_STATUS and RSU_STATUS major and minor error code descriptions, refer to *Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions*.

Related Information

- Appendix A: CONFIG_STATUS and RSU_STATUS Error Code Descriptions
- Intel Stratix 10 Analog to Digital Converter User Guide
  For more information about the temperature sensor channel numbers and temperature sensing diodes (TSDs).
- Intel Stratix 10 Hard Processor System Technical Reference Manual
- Intel Stratix 10 FPGA Boot User Guide
- Intel Agilex Power Management User Guide
  For more information about the temperature sensor channel numbers and temperature sensing diodes (TSDs).

1.2.2. Error Code Responses

Table 4. Error Codes

Value (Hex)	Error Code Response	Description
0	OK	Indicates that the command completed successfully. A command may erroneously return the OK status if a command, such as QSPI_READ is partially successful.
1	INVALID_COMMAND	Indicates that the currently loaded boot ROM cannot decode or recognize the command code.
3	UNKNOWN_COMMAND	Indicates that the currently loaded firmware cannot decode the command code.
4	INVALID_COMMAND_PARAMETERS	Indicates that the command is incorrectly formatted. For example, the length field setting in header is not valid.
6	COMMAND_INVALID_ON_SOURCE	Indicates that the command is from a source for which it is not enabled.
8	CLIENT_ID_NO_MATCH	Indicates that the Client ID cannot complete the request to close the exclusive access to quad SPI. The Client ID does not match the existing client with the current exclusive access to quad SPI.
9	INVALID_ADDRESS	The address is invalid. This error indicates one of the following conditions: • An unaligned address • An address range problem • A read permission problem • An invalid chip select value, displaying value of more than 3 • An invalid address in RSU case • An invalid bitmask value for GET_VOLTAGE command • An invalid page selection for GET_TEMPERATURE command

continued...

(1) This number does not include the command or response header.

<table>
<tr><th>Value (Hex)</th><th>Error Code Response</th><th>Description</th></tr>
<tr><td>A</td><td>AUTHENTICATION_FAIL</td><td>Indicates the configuration bitstream signature authentication failure.</td></tr>
<tr><td>B</td><td>TIMEOUT</td><td>This error indicates time out due to the following conditions:
Command
Waiting for QSPI_READ operation to complete
Waiting for the requested temperature reading from one of the temperature sensors. May indicate a potential hardware error in the temperature sensor.</td></tr>
<tr><td>C</td><td>HW_NOT_READY</td><td>Indicates one of the following conditions:
The hardware is not ready. Can indicate either an initialization or configuration problem. The hardware may refer to quad SPI.
RSU image is not used to configure the FPGA.</td></tr>
<tr><td>D</td><td>HW_ERROR</td><td>Indicates that the command completed unsuccessfully due to unrecoverable hardware error.</td></tr>
<tr><td>80 - 8F</td><td>COMMAND_SPECIFIC_ ERROR</td><td>Indicates a command specific error due to an SDM command you used.
<table>
<tr><th>SDM Command</th><th>Error Name</th><th>Error code</th><th>Description</th></tr>
<tr><td>GET_CHIPID</td><td>EFUSE_SYSTEM_ FAILURE</td><td>0x82</td><td>Indicates that the eFuse cache pointer is invalid.</td></tr>
<tr><td>QSPI_OPEN/ QSPI_CLOSE/ QSPI_SET_CS/ QSPI_READ_D EVICE_REG/ QSPI_WRITE_ DEVICE_REG/ QSPI_SEND_D EVICE_OP/ QSPI_READ</td><td>QSPI_HW_ERROR</td><td>0x80</td><td>Indicates QSPI flash memory error. This error indicates one of the following conditions:
A QSPI flash chip select setting problem
A QSPI flash initialization problem
A QSPI flash resetting problem
A QSPI flash settings update problem</td></tr>
<tr><td></td><td>QSPI_ALREADY_ OPEN</td><td>0x81</td><td>Indicates that the client's exclusive access to QSPI flash via QSPI_OPEN command is already open.</td></tr>
</table></td></tr>
<tr><td>100</td><td>NOT_CONFIGURED</td><td>Indicates that the device is not configured.</td></tr>
<tr><td>1FF</td><td>ALT_SDM_MBOX_RESP_ DEVICE_ BUSY</td><td>Indicates that the device is busy due to following use cases:
RSU: Firmware is unable to transition to different version due to an internal error.
HPS: HPS is busy when in HPS reconfiguration process or HPS cold reset.</td></tr>
<tr><td>2FF</td><td>ALT_SDM_MBOX_RESP_NO _ VALID_RESP_AVAILABLE</td><td>Indicates that there is no valid response available.</td></tr>
<tr><td>3FF</td><td>ALT_SDM_MBOX_RESP_ ERROR</td><td>General Error.</td></tr>
</table>

1.2.3. Error Code Recovery

The table below describes possible steps to recover from an error code. Error recovery depends on specific use case.

Table 5. Error Code Recovery for known Error Codes

Value	Error Code Response	Error Code Recovery
4	INVALID_COMMAND_ PARAMETERS	Resend the command header or header with arguments with corrected parameters. For example, ensures that the length field setting in header is sent with the correct value.
6	COMMAND_INVALID_ ON_SOURCE	Resend the command from valid source such as JTAG, HPS, or core fabric.
8	CLIENT_ID_NO_MATCH	Wait for the client who opened the access to quad SPI to complete its access and then closes the exclusive access to quad SPI.
9	INVALID_ADDRESS	Possible error recovery steps: For GET_VOLTAGE command: Send command with a valid bitmask. For GET_TEMPERATURE command: Send command with valid sensor location and sensor mask. For QSPI operation: • Send command with a valid chip select. • Send command with a valid QSPI flash address. For RSU: Send command with a valid start address of the factory image or application.
B	TIMEOUT	Possible recovery steps: For GET_TEMPERATURE command: Retry to send the command again. If problem persists, reconfigure or power cycle the device. For QSPI operation: Check signal integrity of QSPI interfaces and attempt command again. For HPS restart operation: Retry to send the command again.
C	HW_NOT_READY	Possible recovery steps: For QSPI operation: Reconfigure the device via source. Ensure that IP used to build your design allows access to the QSPI flash. For RSU: Configure the device with RSU image.
80	QSPI_HW_ERROR	Check the QSPI interface signal integrity and to ensure the QSPI device is not damaged.
81	QSPI_ALREADY_OPEN	Client already opened QSPI. Continue with the next operation.
82	EFUSE_SYSTEM_FAILURE	Attempt reconfiguration or power cycle. If error persists after reconfiguration or power cycle, the device may be damaged and unrecoverable.
100	NOT_CONFIGURED	Send a bitstream that configures the HPS.
1FF	ALT_SDM_MBOX_RESP_ DEVICE_ BUSY	Possible error recovery steps: For QSPI operation: Wait for ongoing configuration or other client to complete operation. For RSU: Reconfigure device to recover from internal error. For HPS restart operation: Wait for reconfiguration via HPS or HPS Cold Reset to complete.

1.3. Mailbox Client Intel FPGA Core Signals

The host communicates with the Mailbox Client Intel FPGA over its Avalon Memory-Mapped (Avalon MM) interface. The Avalon MM interface is standard memory-mapped interface. For detailed definitions of these signals, refer to the *Avalon Memory-Mapped Interfaces* chapter in the *Avalon* Interface Specifications.

Table 6. Mailbox Client Intel FPGA Signal Descriptions

Signal Role	Width	Direction	Description
Avalon-MM Interface Signals			
avmm_address	4	Input	Avalon MM address.
avmm_write	1	Input	Avalon MM write request.
avmm_read	1	Input	Avalon MM read request.
avmm_writedata	32	Input	Avalon MM writedata bus.
avmm_readdata	32	Output	Avalon MM readdata bus.
avmm_readdatavalid	1	Output	Avalon MM readdata valid.
Clock and Reset			
clk	1	Input	Input clock to clock the IP. The maximum frequency is 250 MHz.
reset	1	Input	Reset that resets the IP. To reset the IP, assert the reset signal high for at least 2 clk cycles. To ensure the Mailbox Client Intel FPGA IP functions correctly when the device enters user mode, your design must include the Reset Release Intel FPGA IP to hold the reset until the FPGA fabric entered user mode. Intel recommends using a reset synchronizer when connecting the user reset or output of the Reset Release IP to the reset port of the Mailbox Client IP. To implement the reset synchronizer, use the Reset Bridge Intel FPGA IP available in the Platform Designer. *Note:* For IP instantiation and connection guidelines in the Platform Designer, refer to the *Required Communication and Host Components for the Remote System Update Design Example* figure in the *Intel Stratix 10 Configuration User Guide*. *Note:* For IP instantiation guidelines, refer to the *Configuration User Guide*.
irq	1	Output	Interrupt signal. Drives the value of the AND of the interrupt status and interrupt enable registers.

Related Information

- Avalon Memory-Mapped Interface Signal Roles
- Intel Stratix 10 Configuration User Guide
  For information about including the Reset Release IP in your design.
- Intel Agilex Configuration User Guide
  For information about including the Reset Release IP in your design.

1.4. Mailbox Client Intel FPGA IP Avalon MM Memory Map

Table 7. Avalon MM Memory Map

Offset (word)	R/W	31	30:2	1	0
Base address + 0	W	Command			
Base address + 1	W	Command last word (eop)			
Base address + 2	R	Command FIFO empty space			
Base address + 3	N/A	Reserved			
Base address + 4	N/A	Reserved			
Base address + 5	R	Response data			
Base address + 6	R	Response FIFO fill level		EOP	SOP
Base address + 7	R/W	Interrupt enable register (IER)			
Base address + 8	R	Interrupt status register (ISR)			
Base address + 9	R/W	Timer 1 enable	Timer 1 period		
Base address + 10	R/W	Timer 2 enable	Timer 2 period		

1.4.1. Interrupt Enable Register

Use the `Interrupt Enable` register to enable or disable interrupts.

Table 8. Interrupt Enable Register

Bit	Fields	Access	Default Value	Description
31:6	Reserved			
5	EN_BACKPRESSURE_TIMEOUT	R/W	0x0	The enable interrupt bit for SDM backpressure timeout. • 1: Enable the SDM backpressure timeout interrupt • 0: Disable the SDM backpressure timeout interrupt
4	EN_EOP_TIMEOUT	R/W	0x0	The enable interrupt bit for EN_EOP_TIMEOUT. • 1: Enable the EOP timeout interrupt • 0: disable the EOP timeout interrupt
3	EN_COMMAND_INVALID	R/W	0x0	The enable interrupt bit for COMMAND_INVALID. • 1: Enable the command invalid interrupt • 0: Disable the command invalid interrupt
2	Reserved	—	—	Reserved.
1	EN_CMD_FIFO_NOT_FULL	R/W	0x0	The enable for the command FIFO full interrupt. • 1: Enable the FIFO full interrupt • 0: Disable the FIFO full interrupt
0	EN_DATA_VALID	R/W	0x0	The enable for the data valid interrupt. • 1: Enable the data valid interrupt • 0: Disable the data valid interrupt

1.4.2. Interrupt Status Register

Use the `interrupt_status` register to monitor the status of the FIFO and identify invalid commands.

Your logic can poll the error bits of the `interrupt_status` register. Or, you can configure the `EN_COMMAND_INVALID` bit of the interrupt enable register to interrupt when an error occurs.

When an error occurs, the Mailbox Client IP clears all pending responses. Your logic should not expect any response from Mailbox Client IP after an error occurs. Your logic must assert reset for a minimum of 10 clock cycles to reset the Mailbox Client IP.

Table 9. Interrupt Status Register

Bit	Fields	Access	Default Value	Description
31:6	Reserved			
5	BACKPRESSURE_TIMEOUT	R	0x0	SDM backpressure timer interrupt. • 1: The SDM backpressure timer has timeout. Indicates that a fatal error occurred in SDM. You must reset the device. To reset, reconfigure or power cycle the device. • 0: The SDM backpressure timer has not timeout.
4	EOP_TIMEOUT	R	0x0	End of Packet (EOP) timer interrupt. • 1: Indicates that the EOP timer has timeout. You must reset the Mailbox Client IP. • 0: The EOP timer has not timeout. Indicates that the Mailbox Client IP did not receive the full command with EOP due to: • Mailbox did not receive the last argument with EOP. • Mailbox already received all arguments without the EOP in it.
3	COMMAND_INVALID	R	0x0	Invalid command interrupt. Indicates a mismatch between the command length specified in the command header and the number of words sent. Hardware clears this bit. • 1: Indicates that the command is invalid. You must reset the Mailbox client. • 0: The command is valid.
2	Reserved	—	—	Reserved.
1	CMD_FIFO_NOT_FULL	R	0x0	Command FIFO is not full interrupt. • 1: Indicates command FIFO is not full. The client can drive data. • 0: Indicates the FIFO is full. The FIFO automatically clears this bit. You do not need to clear this bit manually.
0	DATA_VALID	R	0x0	Data valid interrupt. • 1: Indicates that valid data is available. The master can read. • 0: Indicates the FIFO is empty. The FIFO automatically clears this bit. You do not need to clear this bit manually.

1.4.3. Timer Registers

Use `timer` registers to monitor and address incomplete transactions between host and the Mailbox Client IP.

Incomplete Command Transaction Error

When a host fails to send the last command word to the Mailbox Client IP or the system stops sending data before the last word, the incomplete command transaction error occurs. Timer 1 allows you to set a specific transaction time period to complete each command. When the timer's timeout occurs, `ISR[4]` is set to indicate the error. To recover the system, you must reset the Mailbox Client IP.

Table 10. Timer 1 Register

Bit	Fields	Access	Default Value[5]	Description
31	`Timer 1 enable`	R/W	0x0	`Timer 1 period` enable bit. The bit is enabled once. • 1: Enable timer 1 • 0: Disable timer 1 If a time out occurs, the timer 1 register becomes disabled. You must apply the Mailbox Client Intel FPGA IP reset. To start the timer 1, you must re-enable it again.
30:0	`Timer 1 period`	R/W	0x7FF_FFFF	When enabled, the timer counts down the specified period as the maximum number of clock cycles the system has not received a valid command. The timer 1 starts the count down as soon as the transaction writes the first data word into the `Command` FIFO (base address +0). The timer resets when the Mailbox Client Intel FPGA IP receives complete command transaction, indicated by successfully writing the last word into the `command last word` register (base address +1). When the timer 1 resets itself, it returns to its default or other defined value.

SDM Backpressure Error

SDM typically backpressures while it processes commands and sends responses. The SDM backpressure error occurs when SDM backpressures for some time period not allowing you to write any data into the Mailbox fabric and SDM. The timer 2, by setting a specific wait time, allows you detect the long wait and take steps to recover your system. When a timer's timeout occurs, `ISR[5]` is set to indicate an error. Note that this is a fatal error received from SDM, possibly indicating a system error. Resetting the Mailbox Client won't recover the system.

(5) Resetting the Mailbox Client IP resets the timer 1 register to the default value.

Table 11. Timer 2 Register

Bit	Fields	Access	Default Value[6]	Description
31	Timer 2 enable	R/W	0x0	Timer 2 period enable bit. The bits is enabled once. • 1: Enable timer 2 • 0: Disable timer 2 If a time out occurs, the timer 2 register becomes disabled. You must apply the Mailbox Client Intel FPGA IP reset. To start the timer 2, you must re-enable it again.
30:0	Timer 2 period	R/W	0x7FF_FFFF	When enabled, the timer counts down the specified period as the maximum number of clock cycles the system has not asserted ready high signal. The SDM backpressures commands sent by host to the Mailbox Client Intel FPGA IP.

1.5. Specifying the Command and Response FIFO Depths

The optimal depth of the command and response FIFOs depends on the specific application. You should size these FIFOs to accommodate the maximum command and responses that your application requires.

The following example illustrates this point. Consider an application sends a maximum of eight back-to-back GET_TEMPERATURE commands to the FPGA core and one transceiver bank. Each command consists of a header and one command argument, specifying the mask of the temperature sensors to read. The optimal setting for the command FIFO is 16 words.

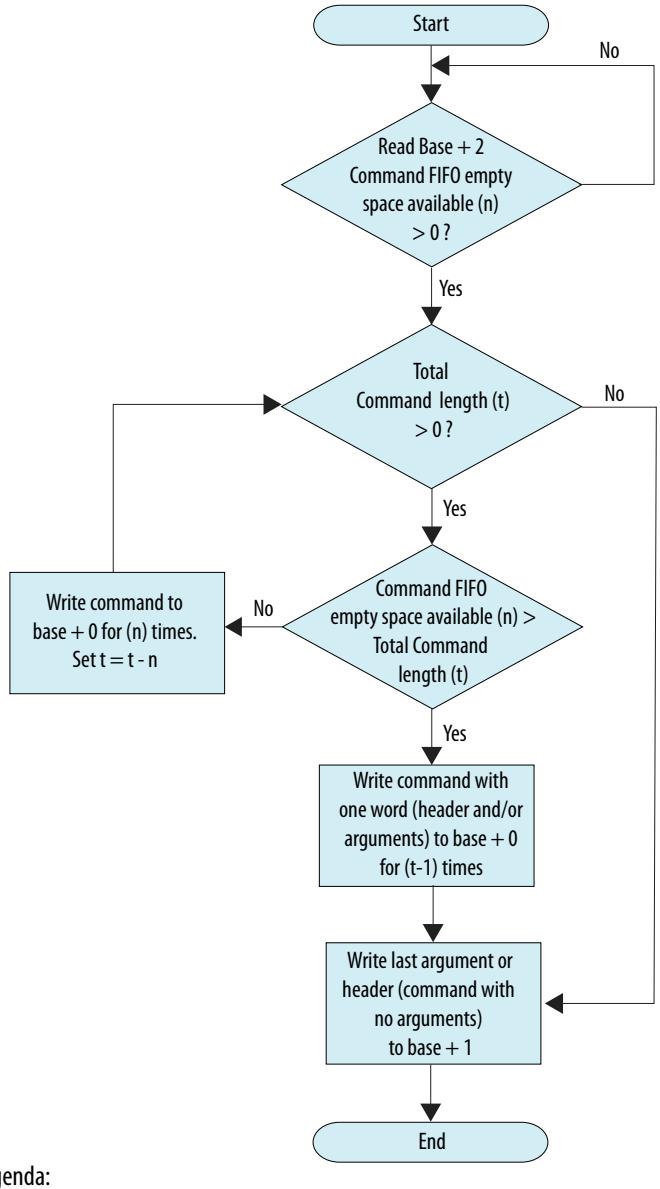
For the response FIFO, each response has one header word, and one word each for the core temperature and transceiver bank temperature. Each response is three words. The optimal setting for the response FIFO Depth is 24 words.

(6) Resetting the Mailbox Client IP resets the timer 2 register to the default value.

1.6. Using the Mailbox Client Intel FPGA IP

Writing Command Packet

Figure 3. Flow Chart for Writing Command Packet



Agenda:
n: Command FIFO empty space
t: Total Command Length

Write Command Description

When you send a command to the SDM, write the command word into command register, which is the base address. To stay in sync with the hardware, while the command length (t) is greater than zero, write the header and arguments in the Command register which is (base address + 0). Continue writing the header and/or

arguments, one word at the time, in the Command register (base address + 0) while there is available free space in the FIFO for commands (n > t). Write the last word to the Command last word register which is (base address +1). For commands with no arguments, write the header to the Command last word register, (base address +1).

Reading from (base address + 2) shows the remaining available free space in the FIFO for commands. The command FIFO can become full when the SDM is busy. The IP requires 3 clock cycles to update the Command FIFO empty space value. You can begin reading the Command FIFO empty space value 3 clock cycles after writing the command to the IP.

You must check the Command FIFO empty space register, (base address + 2) before proceeding to write into the Command/Command last word registers. The behavior of the IP is undefined if you write to (base address + 0) and (base address + 1) while the FIFO is full. The write data is discarded.

Unexpected or undefined behavior may occur if you send more commands than required. For example, send the following commands to read the Chip ID value:

- Write the command header to (base address + 0).
- Write again the command header to (base address + 1).

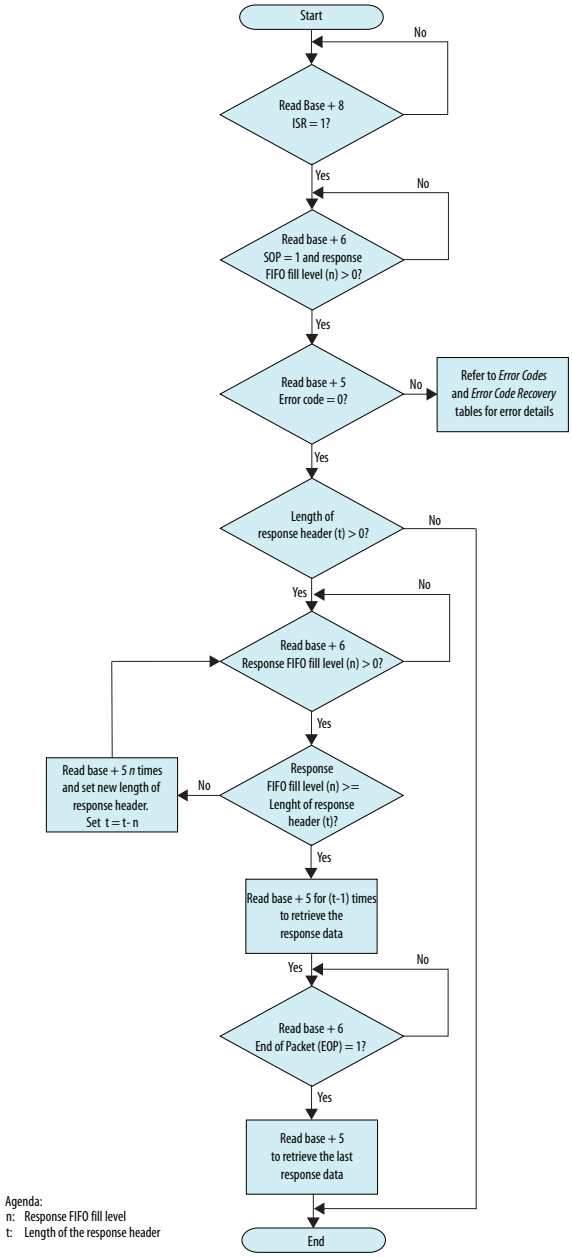
In the above scenario, the IP core expects a 3-word response (command header and 2 data words). However, the SDM only returns a one-word response, which is the error response code.

You must send commands in the correct order to the Command or Command last word register, as described in the Writing Command Packet on page 20. Failure to send commands in the correct order can result in loss of services for all mailbox clients, including the following standalone IP cores:

- Temperature Sensor Intel FPGA IP
- Voltage Sensor Intel FPGA IP
- Chip ID Intel FPGA IP
- Advanced SEU Detection Intel IP
- Partial Reconfiguration Controller Intel IP
- Partial Reconfiguration External Configuration Controller Intel FPGA IP

Reading Response Packet

Figure 4. Flow Chart for Reading Response Packet



Read Command Description

1. Read (base address + 8) to check if bit 0 of `Interrupt status` register is 1, to indicate the valid data is available for the master to read. You can poll the `Interrupt status` register continuously until bit 0 is 1.
2. Read (base address + 6) to check the SOP (start of packet), EOP (end of packet), and the `Response FIFO fill level` (n).

To read multiple words, complete the following steps:

a. If SOP = 1 and EOP = 0, the response has multiple words.

b. If the Response FIFO fill level (n) is non-zero, the FIFO has valid data.

c. For example, if you perform a QSPI_READ operation to read 10 words from quad SPI flash, a return value of 0x0000002d indicates that the SDM wrote 11 words to the response FIFO. The 11 words comprise a response header word and 10 data words.

To read a single word, complete the following steps:

a. If SOP = 1 and EOP = 1, the response has a single word.

b. If the Response FIFO fill level is non-zero, the FIFO has valid data.

c. A return value of 0x00000007 indicates that the SDM wrote a single word to the response FIFO. This single is both the start and end of the single-cycle packet.

3. Read the response header at (base address + 5). The *LENGTH* value specifies the number of words in the response. Proceed to step 4 if the response error code is zero. The response error code is non-zero for unsuccessful commands. Refer to Table 4 on page 12 for more information.

4. When the length of the response header (t) is greater than zero (*LENGTH* > 1) , read (base address + 5) to retrieve the response data. While continuously reading the response data, you must also continuously poll (base address + 6) to check the Response FIFO fill level (n). For the final word of the packet, the Response FIFO fill level (n) and EOP value are expected to be 1 at the same time. You must check for EOP = 1 before proceeding to read the final word from the response data.

   Note: If the response FIFO is empty, the return data is undefined. You must check the Interrupt status register to ensure that valid data is available. You must verify that the Response FIFO fill level (n) is non-zero before reading the response data.

   Ensure that you read or flush out the content in the response FIFO before issuing a new command to the mailbox. Continuously sending commands without reading back the valid data from the response FIFO gradually fills the response FIFO. When the response FIFO overflows the SDM freezes.

   If the SDM freezes you must reconfigure the device. The Intel Quartus Prime software supports device reconfiguration starting in version 19.1. For earlier versions of the Intel Quartus Prime software, power cycle the device to recover.

Restrictions

1. You can only issue one request and read back the response before issuing a new request to the Mailbox Client IP. Wait 10 ms between back-to-back commands to the SDM mailbox.

2. Do not instantiate more than six mailbox clients in your design. For designs requiring more than six mailbox clients, use the Mailbox Client IP to replace the following standalone IP cores:

- Voltage Sensor Intel FPGA IP
- Chip ID Intel FPGA IP
- Serial Flash Mailbox Client Intel FPGA IP
- Temperature Sensor Intel FPGA IP

1.7. Mailbox Client Intel FPGA IP Core Use Case Examples

The Mailbox Client Intel FPGA IP is an Avalon MM slave component that must connect to an Avalon MM master. The simplest Avalon MM master is the JTAG-to-Avalon Master.

The `rsu1.tcl` script provides examples to perform all the available command functions. You can run the functions available in the `rsu1.tcl` script via System Console of the Intel Quartus Prime software.

The following example shows how to access the quad SPI flash memory. Follow this sequence to prevent errors. Refer to Table 3 on page 6 for more information about these commands.

1. `QSPI OPEN`
2. `QSPI_SET_CS`
3. Any of the following quad SPI operations:
   - `QSPI_READ`
   - `QSPI_WRITE`
   - `QSPI_ERASE`
   - `QSPI_READ_DEVICE_REG`
   - `QSPI_WRITE_DEVICE_REG`
   - `QSPI_SEND_DEVICE_OP`
4. `QSPI_CLOSE`

Related Information

Example of Tcl Script
A Tcl script that implements all the Mailbox Client operations.

1.8. Nios II HAL Driver

This section describes the HAL driver for the Mailbox Client Intel FPGA IP core. Intel provides HAL system library drivers that enable you to perform Mailbox Client operation using the HAL API functions. The operations along with their descriptions are listed in Table 3 on page 6. In addition, the HAL driver also provides exclusive functions on QSPI flash device related operations to simplify your design flow.

The HAL API is available for this controller in the following software files:

- `altera_s10_mailbox_client.h`
- `altera_s10_mailbox_client.c`
- `altera_s10_mailbox_client_flash.h`
- `altera_s10_mailbox_client_flash.c`

These files implement the Mailbox Client core device driver for the HAL system library. To use the HAL API, enable `altera_safeclib` in the *BSP Software Package* from BSP Editor. Enter `mailbox_client_open()` function to start the HAL API. Note that the interrupt connection to the processor is necessary to use the HAL API.

Related Information

- Nios II Software Developer Handbook
  For more information about using flash devices.
- Nios II Configuration and Booting Solutions
  For more information about booting Nios II from flash.

1.8.1. Driver API

Table 12. mailbox_client_open

Prototype:	mailbox_client_open(const char *name)
Include:	<altera_s10_mailbox_client.h>
Parameter:	• name – character pointer to name of Mailbox Client Intel FPGA peripheral as registered with the HAL.
Return:	Return non-NULL if successful and otherwise return: • -NULL for Invalid argument. Found no device or wrong input name.
Description:	Retrieve a pointer to the Mailbox Client block instance. Search for the list of registered mailbox client instances for the one with the supplied name.

Table 13. mailbox_client_send_cmd

Prototype:	mailbox_client_send_cmd (intel_mailbox_client* fd, alt_u8 id, alt_u32 cmd, alt_u32* arg, int arg_length, int cmd_length, alt_u32* input_data, alt_u32* resp_buf, alt_u32* resp_buf_len)
Include:	<altera_s10_mailbox_client.h>

continued...

Parameter:	• fd – pointer to the flash device structure • id - command ID • cmd - mailbox client command • arg - mailbox Client argument(7) • arg_length - number of arguments • cmd_length - command length (arg_length + input_data length) • input_data - input data • resp_buf - response buffer • res_buf_len - response buffer length
Return:	Return 0 if successful and otherwise return: • error codes for mailbox client in Table 4 on page 12. • -EINVAL for Invalid argument • -ETIME for polling timeout and skipping the loop after 5 seconds • -EBACK_TOUT for back pressure timer interrupt(8) • -EEOP_TOUT for end of packet timer interrupt(9) • -ECMD_INVLD for invalid command interrupt • -ENOBUFS for the response buffer length is insufficient • -ENOSYS for unmatched returned command ID from SDM
Description:	Send commands and data to the Mailbox Client. Reads back the response when data valid interrupt occurs. For more information, refer to Example 1 in Driver API Application on page 28.

Table 14. mailbox_client_flash_open

Prototype:	mailbox_client_flash_open(intel_mailbox_client* fd)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure.
Return:	Return 0 if successful and otherwise return: • -EINVAL for Invalid argument. • -ENODEV for unrecognized flash device.
Description:	Send command to the mailbox client for QSPI open and QSPI chip select. Provides exclusive access to the quad SPI interface. Determines the QSPI flash size.

Table 15. mailbox_client_flash_close

Prototype:	mailbox_client_flash_close(intel_mailbox_client* fd)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure.
Return:	Return 0 if successful and otherwise return: • -EINVAL for Invalid argument.
Description:	Send command to the mailbox client for QSPI close. Stops the exclusive access to the quad SPI interface.

(7) Argument represents the parameters needed for mailbox client operation, not including input data.

(8) The recommended error code recovery is the full system reconfiguration.

(9) The recommended error code recovery is to reset the Mailbox Client Intel FPGA IP.

Table 16. mailbox_client_flash_read

Prototype:	mailbox_client_flash_read (intel_mailbox_client* fd, int offset, void* dest_addr, int length)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure. • offset - read flash address (unaligned access) • dest_addr - destination buffer (in byte size) • length - size of read data (in byte size)
Return:	Return 0 if successful and otherwise return: • -EINVAL for Invalid argument.
Description:	Read data from selected address specified by the length in byte size. Length is a non-zero value.

Table 17. mailbox_client_flash_erase_block

Prototype:	mailbox_client_flash_erase_block (intel_mailbox_client* fd, int block_offset)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure. • block_offset - address offset from the start of flash sector specified to be erased.
Return:	Return 0 if successful and otherwise return: • -EINVAL for Invalid argument. • -EIO for erase failed and sector might be protected
Description:	Erase a single flash sector.

Table 18. mailbox_client_flash_write_block

Prototype:	mailbox_client_flash_write_block (intel_mailbox_client* fd, int block_offset, int data_offset, const void* data, int length)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure. • block_offset - sector address. Starts at the flash sector specified to be written to. • data_offset - byte offset (unaligned access) of write into memory • data - data buffer to be written (in byte size) • length - size of write data (in byte size)
Return:	Return 0 if successful and otherwise return: • -EINVAL for Invalid argument
Description:	Write one block or sector of data into the flash. The write data length cannot split between adjacent sectors. The function assumes the address is empty. You must erase it prior its programming.

Table 19. mailbox_client_flash_write

Prototype:	mailbox_client_flash_write (intel_mailbox_client* fd, int offset, const void* src_addr, int length)
Include:	<altera_s10_mailbox_client_flash.h>
Parameter:	• fd – pointer to the flash device structure. • offset - flash address (unaligned access) of write to the flash memory. • src_addr - source buffer (in byte size) • length - size of write data (in byte size)
Return:	Return 0 if successful and otherwise return:

continued...

	• -EINVAL for Invalid argument
Description:	Program the data into the flash at the selected address. Automatically erase the block as needed before programming.

Related Information

Error Code Responses on page 12

1.8.2. Driver API Application

The driver API application starts by calling `mailbox_client_open()`. This procedure opens the Mailbox Client Intel FPGA IP, which returns a file handle. In addition, the API application calls the flash exclusive access when the quad SPI operations are carried out.

```
//OPEN MAILBOX CLIENT IP
fd = mailbox_client_open("/dev/mailbox_client");
printf("Non-QSPI operation can be carried out here.");

//FLASH EXCLUSIVE ACCESS OPERATIONS
ret_code = mailbox_client_flash_open(fd);
if(ret_code == 0)
{
   printf("QSPI operation can be carried out here.");
}
ret_code = mailbox_client_flash_close(fd);
```

The absolute addressing to the quad SPI memory specifies all the offset-related variables. You must complete the quad SPI operation by calling `mailbox_client_flash_close()` to end the process.

Example 1. Send Operation Command using `mailbox_client_send_cmd()`

This operation sends a command and retrieve a response through mailbox client. The example below uses QSPI_WRITE and QSPI_READ commands using one word (N = 1).

For information about the command list and command descriptions, refer to Table 3 on page 6.

```
//QSPI_WRITE
id = <4-bit_number>;
cmd = 0x39;
arg[0] = <flash_address_offset>;
arg[1] = N;
arg_length = 2;
cmd_length = 2 + N;
input_data[0] = 0xabcdef34;
resp_length = 0;

ret_code = mailbox_client_send_cmd(fd, id, cmd, arg, arg_length, cmd_length,
input_data, resp_buf, resp_length);
if(ret_code == 0)
{
   printf("Write data at address 0x%08X is successful.\n", arg[0]);
}

//QSPI_READ
id = <4-bit_number>;
cmd = 0x3a;
arg[0] = <same_flash_address_offset>;
arg[1] = N;
arg_length = 2;
```

```
cmd_length = 2;
resp_length = N;

ret_code = mailbox_client_send_cmd(fd, id, cmd, arg, arg_length, cmd_length,
input_data, resp_buff, resp_length);
if(ret_code == 0)
{
   printf("Read data at address 0x%08X is 0x%08X.\n", arg[0], resp_buff[0]);
}
```

Example 2. Read Data from Flash using `mailbox_client_flash_read()`

This operation reads data from the flash device from selected address for the specified length in bytes.

```
ret_code = mailbox_client_flash_read(fd, offset, dest, dest_length);
if(ret_code == 0)
{
   for(int i = 0; i < dest_length; i++)
   {
      printf("Data read from address 0x%08X is 0x%02X", (offset+i), dest[i]);
   }
}
```

Example 3. Erase Flash Sector using `mailbox_client_flash_erase_block()`

This operation erases a single block in the flash memory.

```
ret_code = mailbox_client_flash_erase_block(fd, block_offset);
if(ret_code == 0)
{
   printf("Flash sector from address 0x%08X to 0x%08X is erased.",
block_offset, (block_offset+sector_size-1));
}
```

Example 4. Write Single Section in Flash using `mailbox_client_flash_write_block()`

This operation writes specified length of data in bytes to a single empty block in the flash memory.

```
ret_code = mailbox_client_flash_write_block(fd, block_offset, data_offset,
source, source_size);
if(ret_code == 0)
{
   printf("Flash memory from address 0x%08X to 0x%08X is written.",
data_offset, (data_offset+source_size-1));
}
```

Example 5. Write Data in Flash using `mailbox_client_flash_write()`

This operation writes specified length of data in bytes to the flash device and automatically erase the block if needed.

```
ret_code = mailbox_client_flash_write(fd, offset, source, source_size);
if(ret_code == 0)
{
   printf("The affected blocks are erased.\n");
   printf("Flash memory from address 0x%08X to 0x%08X is written.", offset,
(offset+source_size-1));
}
```

Related Information

Operation Commands on page 6

1.9. Mailbox Client Intel FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

If an IP core version is not listed, the user guide for the previous IP core version applies.

Intel Quartus Prime Version	IP Core Version	User Guide
20.4	20.0.0	Mailbox Client Intel FPGA IP User Guide
20.3	20.0.0	Mailbox Client Intel FPGA IP User Guide
20.2	20.0.0	Mailbox Client Intel FPGA IP User Guide
20.1	20.0.0	Mailbox Client Intel FPGA IP User Guide
19.3	19.1	Mailbox Client Intel FPGA IP User Guide
18.1	18.1	Mailbox Client Intel FPGA IP User Guide
17.1	17.1	Mailbox Client Intel FPGA IP User Guide

1.10. Document Revision History for the Mailbox Client Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	Changes
2021.03.29	21.1	Made the following changes: • Revised the *Flow Chart for Response Packet* figure and the *Read Command Description* section. • Revised RSU_IMAGE_UPDATE description in the *Command List and Description* table. • Added new topics: — *Nios II HAL Driver* — *Driver API* — *Driver API Application* • Restructured *Operation Commands*. Moved major and minor error code descriptions for the CONFIG_STATUS and RSU_STATUS commands to the *Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions*.
2020.12.14	20.4	Made the following changes: • Revised block diagram description in the *Mailbox Client Intel FPGA IP User Guide* topic. • Updated the *Mailbox Client Intel FPGA IP System Block Diagram* figure. The figure depicts various ways to communicate with Mailbox Client IP. • Added important note about resetting QSPI flash in the *Operation Commands* topic. • Updated the *Command List and Description* table: — Revised GET_TEMPERATURE command description. Clarified difference between Intel Stratix 10 and devices. — Revised RSU_IMAGE_UPDATE command description. • Added text about resetting QSPI flash. • Added text describing behavior between the external host and FPGA. • Removed text: *Returns a non-zero response if the device is already processing a configuration command.* — Updated QSPI_WRITE and QSPI_READ descriptions to specify that the maximum transfer size is 4 kilobytes or 1024 words. — Corrected response length from 1 to 0 for the QSPI_OPEN, QSPI_CLOSE and QSPI_SET_CS command. — Revised QSPI_OPEN, QSPI_WRITE, QSPI_READ_DEVICE_REG, and QSPI_WRITE_DEVICE_REG descriptions. — Added a new command: REBOOT_HPS. • Added new topic: Error Code Recovery. • Revised *Timer Registers* topic. Added footnotes and updated registers description. • Updated *Flow Chart for Reading Response Packet* figure.
2020.10.05	20.3	Made the following changes: • Revised GET TEMPERATURE command description for Intel Agilex devices in the *Command List and Description* table. • Added recommendation about the reset synchronizer in the *Mailbox Client FPGA Core Signals* section. • Updated the *Error Codes* table. Added new error code responses: — HW_ERROR — COMMAND_SPECIFIC_ERROR
2020.06.30	20.2	Made the following changes:

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Revised LENGTH and Command Code/Error Code descriptions in the *Command and Response Header Description* table. • Revised GET_TEMPERATURE command description in the *Command List and Description* table. • Removed UNKNOWN_BR command from the *Error Codes* table. • Added new timer feature to handle the error detection for the incomplete transaction timeout error and the SDM backpressure timeout fatal error. • Added support for an EOP_TIMEOUT interrupt which indicates that the full command did not include the EOP. • Added support for a BACKPRESSURE_TIMEOUT interrupt which indicates that an error within the SDM occurred. • Removed SD/MMC text from the CLIENT_ID_NO_MATCH description in the *Error Codes* table. • Updated write and read command descriptions in the *Using the Mailbox Client Intel FPGA IP* section.
		continued...

Document Version	Intel Quartus Prime Version	Changes
2020.04.13	20.1	Made the following changes: • Added the following restriction to the definition of QSPI_SET_CS: *Access to the QSPI flash memory devices using SDM_IO pins is only available for the AS x4 configuration scheme, JTAG configuration, and a design compiled for ASx4 configuration. For the Avalon ST configuration scheme, you must connect QSPI flash memories to GPIO pins.* • Added the following text to the definition of the `Failing image` field of the RSU_STATUS command: *Note:* A rising edge on nCONFIG to reconfigure from ASx4, does not clear this field. Information about failing image only updates when the Mailbox Client receives a new RSU_IMAGE_UPDATE command and successfully configures from the update image. • Added RSU_NOTIFY command in the *Command List and Description* table. • Revised the *Flow Chart for Writing Command Packet* and *Flow Chart for Reading Response Packet* to include the correct sequence for writing commands into a command FIFO and reading response packets from a response FIFO. Updated corresponding *Write Command Description* and *Read Command Description* sections.
2020.03.17	19.3	Made the following changes: • Updated the *Error Codes* table: — Renamed INVALID_COMMAND_PARAMETERS to INVALID_LENGTH. — Changed COMMAND_INVALID_ON_SOURCE hex value from 5 to 6. — Changed CLIENT_ID_NO_MATCH hex value from 6 to 8. — Changed INVALID_ADDRESS hex value from 7 to 9. — Added AUTHENTICATION_FAIL command. — Changed TIMEOUT hex value from 8 to B. — Changed HW_NOT_READY hex value from 9 to C.
2019.09.30	19.3	Made the following changes: • Added device support for the Intel Agilex device. • Added support for a COMMAND_INVALID interrupt which indicates the command length specified in the header does not match the actual command sent. • Changed name of the IP from Mailbox Client Intel Stratix 10 FPGA IP to Mailbox Client Intel FPGA IP. • Revised introduction including the *Figure 1: Mailbox Client Intel FPGA IP System Block Diagram*. • Revised the *Flow Chart for Writing Command Packet* and *Flow Chart for Reading Response Packet* to include logic to handle multiple word commands and responses. • Changed references to names of all mailbox client IPs. The mailbox clients IP no longer include the Intel Stratix 10 FPGA in their names. • Added reference to *AN 891: Using the Reset Release Intel FPGA IP*. • Added reference to the *Intel Agilex Power Management User Guide*. • Updated the description of the GET_TEMPERATURE command to say the mask argument is optional. When omitted, the command returns the temperature for sensor 0. • Updated the RSU_STATUS command to say the highest priority failing image, not the last failing image. The error information is for the first failing image which is the highest priority failing image. • Added descriptions for CONFIG_STATUS and RSU_STATUS major and minor error codes.

Document Version	Intel Quartus Prime Version	Changes
		• Added HPS_COLDRESET and HPS_WARMRESET to the list of soft functions for the CONFIG_STATUS command. • Added *Mailbox Client Intel FPGA IP User Guide Archives* topic. • Added the following Intel FPGA IPs to the list of IPs that require proper use of the Command and Command last registers: — Advanced SEU Detection Intel IP — Partial Reconfiguration Controller Intel IP — Partial Reconfiguration External Configuration Controller Intel FPGA IP — Edited entire user guide for clarity and style.

Document Version	Changes
2019.04.19	• Updated the *Feature Description* topic. • Added a note to Figure: *Command and Response Header Format*. • Updated Table: *Mailbox Client Intel Stratix 10 FPGA IP Command and Response Header Description* to update the description for bit[11] of the command and response header. • Updated Table: *Command List and Description* to update the descriptions for CONFIG_STATUS and RSU_STATUS. • Renamed topic title *Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-MM Interface* to *Mailbox Client Intel Stratix 10 FPGA IP Core Signals*. • Renamed table title *Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-MM Interface* to *Mailbox Client Intel Stratix 10 FPGA IP Core Signal Description*. • Updated Table: *Mailbox Client Intel Stratix 10 FPGA IP Core Signal Description* to include information on clock and reset signals. • Updated Table: *Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-Memory Map* to remove urgent command and urgent FIFO empty space. • Updated the *Using the Mailbox Client Intel Stratix 10 FPGA IP Core* topic: — Added new Figures: *Flow Chart for Writing Command Packet* and *Flow Chart for Reading Response Packet*. — Added a new section—*Restrictions*. — Updated the description in the *Writing Command Packet* section. • Updated the *Mailbox Client Intel Stratix 10 FPGA IP Core Use Case Examples* topic. • Made editorial updates throughout the document.
2019.03.14	• Updated the *Mailbox Client Intel Stratix 10 FPGA IP Core User Guide* topic. • Updated Figure: *Mailbox Client Intel Stratix 10 FPGA IP Core and System Block Diagram*. • Updated Table: *Command List and Description*: — Updated the column name *Number of Commands* to *Command Length*. — Updated the column name *Number of Responses* to *Respond Length*. — Corrected the description for QSPI_READ, QSPI_WRITE, and QSPI_ERASE.
2019.02.25	• Updated the description in the *Mailbox Client Intel Stratix 10 FPGA IP Core User Guide* topic. • Updated Figure: *Mailbox Client Intel Stratix 10 FPGA IP Core User Guide*. • Updated Table: *Interrupt Status Register* to update the description for DATA_VALID. • Renamed the following topic titles: — *Commands and Error Codes* to *Commands and Responses* — *Commands* to *Operation Commands*. • Updated Table: *Mailbox Client Intel Stratix 10 FPGA IP Command and Response Header Description* to update the descriptions for Length and Command Code/Error Code. • Updated Table: *Command List and Description*: — Updated the number of responses and description for CONFIG_STATUS. — Updated the number of responses for RSU_STATUS. — Updated the descriptions for QSPI_READ, QSPI_WRITE, and QSPI_ERASE.

continued...

Document Version	Changes
	• Updated Table: *Mailbox Client Intel Stratix 10 FPGA IP Error Code Responses and Description* to update the description for UNKNOWN_BR. • Updated the *Writing Command Packet* and *Reading Command Packet* sections in the *Using the Mailbox Client Intel Stratix 10 FPGA IP Core* topic. • Updated the *Mailbox Client Intel Stratix 10 FPGA IP Core Use Case Examples* topic. • Removed the following topics: — *Example 1: Reading Intel Stratix 10 IDCODE and Voltage* — *Example 2: Read and Write EPCQ-L or QSPI Devices*
2018.10.15	• Updated Table: *Command List and Description* to include the following commands: — Updated the descriptions for GET_TEMPERATURE. — Added new commands: • RSU_IMAGE_UPDATE • CONFIG_STATUS • RSU_STATUS — Removed the command GET_DESIGNHASH. • Updated Table: *Error Code Responses and Description* to update the value of the following error code responses: — NOT_CONFIGURED — ALT_SDM_MBOX_RESP_DEVICE_BUSY — ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE — ALT_SDM_MBOX_RESP_ERROR • Added a note to Figure: *Mailbox Client Intel Stratix 10 FPGA IP Core Block Diagram*. • Made minor editorial updates.
2018.02.14	Initial release.

2. Appendix: CONFIG_STATUS and RSU_STATUS Error Code Descriptions

The CONFIG_STATUS and RSU_STATUS commands allows you to check the current configuration status or the current remote system upgrade status. The commands return 0 when there is no error. If the operation was unsuccessful, the command returns at least one error code described in the table below.

The `Error Code` field in the command header provides details of major and minor error codes. For more information about specific bits representing the major and minor error codes in the CONFIG_STATUS and RSU_STATUS command, refer to the *Command List and Description* table.

Figure 5. Command Header Format

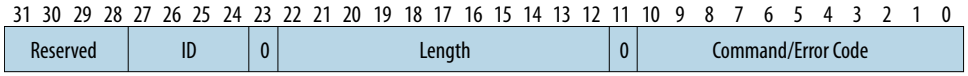


Table 20. CONFIG_STATUS and RSU_STATUS Major Error Code Descriptions

Table displays the major error codes and their descriptions received through the `Error Code` field.

Major Error Code	Error Type	Description
0xF001	ERR_BITSTREAM_ERROR	Indicates a bitstream error.
0xF002	ERR_EXT_HW_ACCESS_FAIL	Indicates an external hardware access error.
0xF003	ERR_BITSTREAM_ CORRUPTION	Indicates a bitstream corruption error.
0xF004	ERR_INTERNAL_ERROR	Indicates an internal error due to misunderstood bitstream element.
0xF005	ERR_DEVICE_ERROR	Indicates a device operation error.
0xF006	ERR_HPS_WDT	Indicates the HPS watchdog timeout failure. Ensure that your design resets the watchdog timer correctly.
0xF007	ERR_INTERNAL_UNKNOWN_ ERROR	Indicates an internal device error due to an unknown task.
0xF008	ERR_SYSTEM_INIT_ERROR	Indicates an error due to the system initialization failure.
0xF009	ERR_DECRYPTION_ERROR	Indicates an error due to a bitstream decryption.

Table 21. CONFIG_STATUS and RSU_STATUS Minor Error Code Descriptions

Table displays the minor error codes and their descriptions received through the `Error Code` field.

Minor Error Code	Description
0x0001	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0002	Indicates a QSPI-related error due to the following conditions: • An incorrect connection between the QSPI device and the FPGA device. • QSPI device is in reset mode.
0x0003	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0004	Indicates a configuration error due to an incompatible bitstream with the device. Ensure the usage of a correct bitstream.
0x0005 - 0x0007	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0008	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0009 - 0x0014	
0x0015	Indicates a bitstream authentication error during configuration. Ensure you signed the bitstream with the correct signing key.
0x0016	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0017 - 0x0024	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0025	Indicates a firmware transitional error during configuration. Ensure the device firmware and the current Intel Quartus Prime software version are compatible. To recover, remove the current running firmware from the device.
0x0026 - 0x0031	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0032	Indicates a PMBUS error during configuration due to an incorrect VID setting in the Intel Quartus Prime project. The target device failed to communicate with a smart regulator or PMBUS master on a board.
0x0033	Indicates a PMBUS error during configuration due to an incorrect VID setting in the Intel Quartus Prime project. The target device failed to communicate with a smart regulator or PMBUS master on a board.
0x0034 - 0x0035	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0036	Reserved
0x0037 - 0x0041	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0042	Indicates an incompatible partial reconfiguration (PR) bitstream. Ensure you use the PR bitstream compatible with the current base design.
0x0043 - 0x0049	Reserved
0x004A- 0x004F	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.

continued...

Minor Error Code	Description
0x0050	Indicates an error during configuration due to a device mismatch between the Intel Quartus Prime project and the target device.
0x0051 - 0x0052	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0053 - 0x0054	Indicates a bitstream decryption error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0055	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0056 - 0x0058	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0059 - 0x0061	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0062	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0063	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0064 - 0x0066	Indicates a configuration error due to a corrupted bitstream. Ensure a valid connection between the device and the configuration source.
0x0067	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0x0068	Indicates that the detected bitstream is incompatible due to the security enabled settings. You cannot use the bitstream from an advanced security-enabled devices on a non-advanced security-enabled device. Ensure the Intel Quartus Prime project device matches the target device.
0x0069	Indicates that the detected bitstream is invalid due to reached maximum number of supported partial reconfiguration (PR) authentication. The bitstream supports up to 32 PR partitions.
0x006A - 0x0075	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xC001	Indicates a firmware error during reconfiguration. Check the latest Intel Quartus Prime software release for possible fixes.
0xC002 - 0xC006	Indicates a bitstream error during reconfiguration. Check the bitstream validity. If corrupted, regenerate and configure bitstream again.
0xC007	Indicates an error due to transitioning to other firmware version or application image. Ensure the bitstream is valid. If corrupted, regenerate and reprogram QSPI flash with RSU image via the JTAG interface.
0xC008	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xC009	Indicates a bitstream authentication error during reconfiguration. Ensure you use the correct signing key when signing the bitstream.
0xC00A	Indicates an error during configuration. To recover, power cycle the device.
0xC00B	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.

continued...

Minor Error Code	Description
0xC00D	Indicates a hardware error during reconfiguration. To recover, power cycle the device.
0xC00E	Indicates a bitstream error during reconfiguration. Check the bitstream validity. If corrupted, regenerate and configure bitstream again.
0xC00F	Indicates an error when accessing the QSPI flash memory. Reconfigure device by toggling nCONFIG pin signal or power cycle the device.
0xD001	Indicates an authentication failure for the firmware. Ensure you use the correct firmware signing key when enabling firmware co-signing feature.
0xD002	Indicates an authentication failure for the design. Ensure you sign the bitstream with a correct signing key.
0xD003	Indicates an error when loading the application image from QSPI flash. Ensure the application image located at the correct address in QSPI flash.
0xD004	Indicates an error when parsing the RSU CPB block. RSU CPB block is corrupted. To recover, toggle the nCONFIG pin to restart the RSU. If the issue persists, reprogram the RSU CPB block data.
0xD005	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xD006	Failed to load factory image. Ensure the factory image is valid. If corrupted, regenerate and reprogram the factory image in the flash. When authentication enabled, ensure you use the correct signing key.
0xD007	Indicates and error when loading the application image. Check the application image validity. If corrupted, regenerate and reprogram again the application image in the flash.
0xD008	Indicates and error during factory image update in flash memory. Check the factory update image validity. If corrupted, regenerate and reprogram again the factory updated image in the flash.
0xD009	Indicates an error during DCMF update in flash memory. Check the factory update image validity. If corrupted, regenerate and reprogram again the factory updated image. Ensure you use the correct signing key when authentication enabled.
0xD00A	Indicates an error during DCMF update in flash memory. Flash memory may have reset during the update process. To recover, toggle the nCONFIG signal to restart the update process.
0xD00B	Indicates an error during DCMF update in flash memory. Flash memory may have reset during the update process. To recover, toggle the nCONFIG signal to restart the update process.
0xD00C	Indicates an error during RSU CPB table update in flash. RSU CPB block data may be corrupted. To recover, regenerate the RSU image containing the updated factory image and program it to flash.
0xE001	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xE002	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xE003 - 0xE008	Reserved
0xE009 - 0xE00B	Indicates an error during configuration. Refer to the *Configuration User Guide* for details on the debug guidelines. Check the latest Intel Quartus Prime software release for possible fixes.
0xE00C	Reserved

Related Information

- Intel Stratix 10 Configuration User Guide: Debugging Guide

- Intel Agilex Configuration User Guide: Debugging Guide