

# Mailbox Avalon<sup>®</sup> ST Client Intel<sup>®</sup> FPGA IP User Guide

Updated for Intel<sup>®</sup> Quartus<sup>®</sup> Prime Design Suite: **20.1**

IP Version: **1.0.0**



[Subscribe](#)

[Send Feedback](#)

**UG-20254 | 2020.04.13**

Latest document on the web: [PDF](#) | [HTML](#)



## Contents

---

<b>1. Mailbox Avalon® ST Client Intel FPGA IP Overview.....</b>	<b>3</b>
1.1. Device Family Support.....	4
1.1.1. Parameters.....	5
1.1.2. Interfaces.....	5
1.2. Command and Response Header.....	7
1.3. Supported Commands.....	8
1.4. Temperature Sensor Locations.....	12
1.5. Error Codes.....	14
1.6. Mailbox Avalon ST Client Intel FPGA IP User Guide Document Archives .....	15
1.7. Document Revision History for the Mailbox Avalon ST Client Intel FPGA IP User Guide ..	15

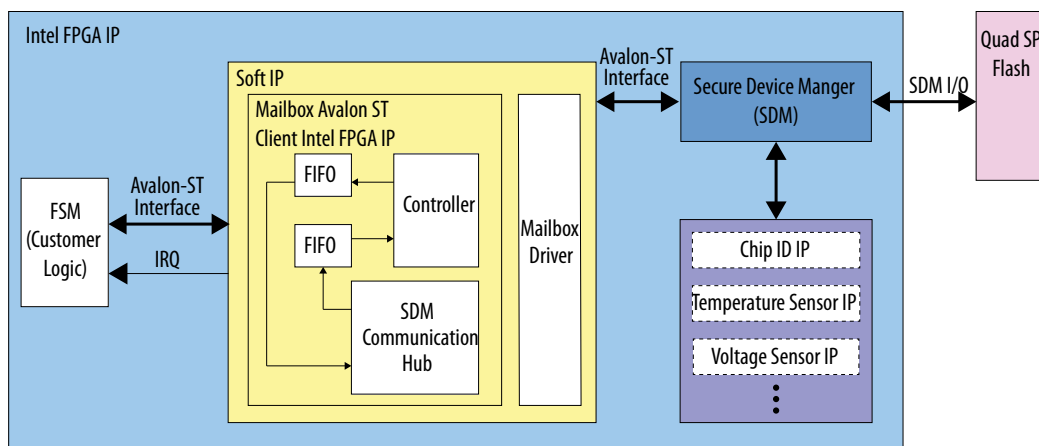
## 1. Mailbox Avalon® ST Client Intel FPGA IP Overview

The Mailbox Avalon® ST Client Intel® FPGA IP provides a communication channel between your custom logic and the secure device manager (SDM). You can use the Mailbox Avalon ST Client IP to send command packets and receive response packets from SDM peripheral modules. The Mailbox Avalon ST Client IP defines functions that the SDM runs.

Your custom logic can use this communication channel to receive information and access flash memory from the following peripheral modules:

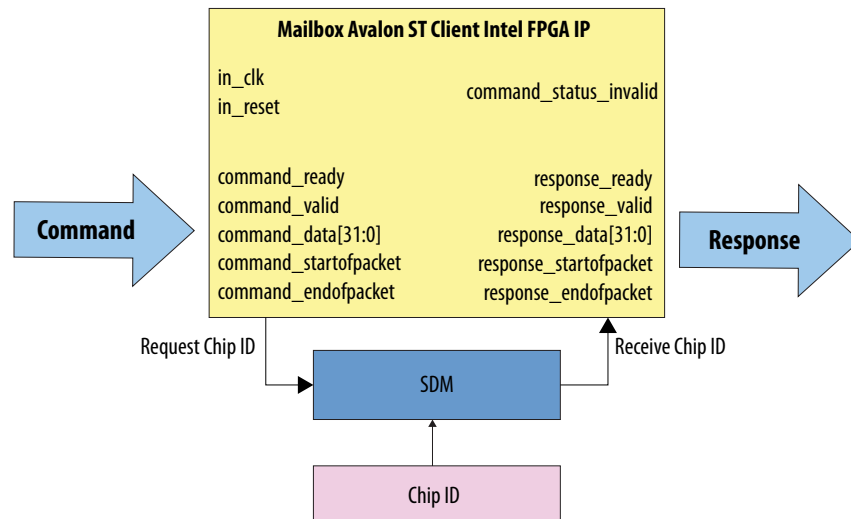
- The Chip ID
- The Temperature Sensor
- The Voltage Sensor
- Quad serial peripheral interface (SPI) flash memory

**Figure 1. Mailbox Avalon ST Client Intel FPGA IP System Design**



The following figure shows an application in which the Avalon ST Client Intel FPGA IP reads the Chip ID.

Figure 2. Mailbox Avalon ST Client Intel FPGA IP Reads Chip ID



### 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 Agilex™	Advance

*Note:* Intel does not provide simulation modes for the Mailbox Avalon ST Client Intel FPGA IP.

#### Related Information

[Mailbox Avalon ST Client Release Notes](#)



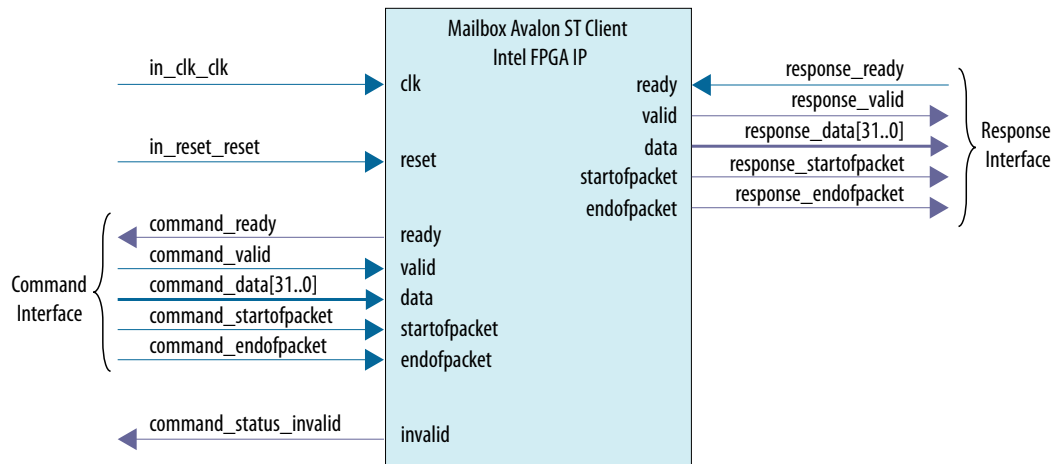
### 1.1.1. Parameters

Parameter Name	Value	Description
<b>Enable status interface</b>	<b>On</b> <b>Off</b>	When you enable this interface, the Mailbox Avalon ST Intel FPGA Client IP includes the <code>command_status_invalid</code> signal. When <code>command_status_invalid</code> asserts, you must reset the IP.

### 1.1.2. Interfaces

The following figure illustrates the Mailbox Avalon ST Client IP interfaces:

**Figure 3. Mailbox AvalonST Client FPGA IP Interfaces**



For more information about Avalon ST interfaces, refer to the *Avalon Interface Specifications*.

#### Related Information

[Avalon Interface Specifications](#)

#### 1.1.2.1. Clock and Reset Interfaces

**Table 2. Clock and Reset Interfaces**

Signal Name	Direction	Description
<code>in_clk</code>	Input	This is the clock for the Avalon ST interfaces. The maximum frequency is 250 MHz.
<code>in_reset</code>	Input	This is an active high reset. Assert <code>in_reset</code> to reset the Mailbox Avalon ST Client Intel FPGA IP. When the <code>in_reset</code> signal asserts, the SDM must flush any pending activity from the Mailbox Avalon ST Client Intel FPGA IP. The SDM continues to process commands from other clients.

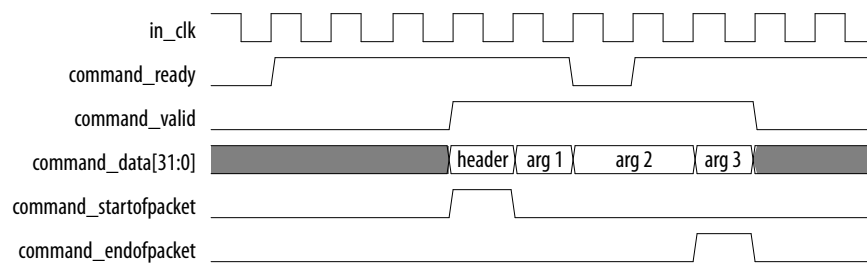
### 1.1.2.2. Command Interface

Use the Avalon Streaming (Avalon ST) interface to send commands to the SDM.

**Table 3. Command Interface**

Signal Name	Direction	Description
command_ready	Output	The Mailbox Avalon ST Client Intel FPGA IP asserts <code>command_ready</code> when it is ready to receive commands from the application. The <code>ready_latency</code> is 0 cycles. The Mailbox Avalon ST Client can accept <code>command_data[31:0]</code> in the same cycle that <code>command_ready</code> asserts.
command_valid	Input	The <code>command_valid</code> signal asserts to indicate that <code>command_data</code> is valid.
command_data[31:0]	Input	The <code>command_data</code> bus drives commands to the SDM. Refer to <a href="#">Table 7</a> on page 8 for definitions of the commands.
command_startofpacket	Input	The <code>command_startofpacket</code> asserts in the first cycle of a command packet.
command_endofpacket	Input	The <code>command_endofpacket</code> asserts in the last cycle of command a packet.

**Figure 4. Timing for Avalon ST Command Packet**



### 1.1.2.3. Response Interface

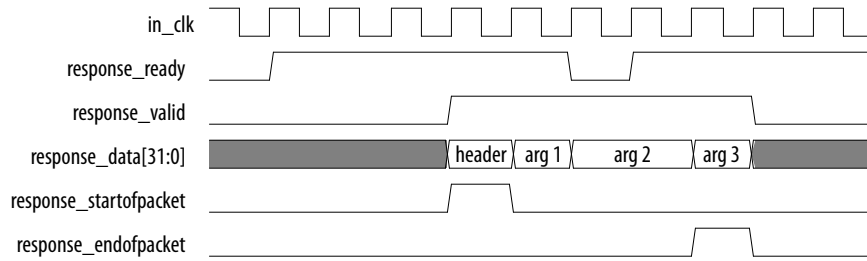
The SDM Avalon ST Client IP sends responses to your application using the response interface.

**Table 4. Response Interface**

Signal 5	Direction	Description
response_ready	Input	Application logic can assert the <code>response_ready</code> signal whenever it is able to receive a response.
response_valid	Output	The SDM asserts <code>response_valid</code> to indicate that <code>response_data</code> is valid.
response_data[31:0]	Output	The SDM drives <code>response_data</code> to provide the requested information. The first word of the response is a header that identifies the command that the SDM is providing. Refer to <a href="#">Table 7</a> on page 8 for definitions of the commands.
response_startofpacket	Output	The <code>response_startofpacket</code> asserts in the first cycle of a response packet.
response_endofpacket	Output	The <code>response_endofpacket</code> asserts in the last cycle of a response packet.



**Figure 5. Timing for Avalon ST Response Packet**

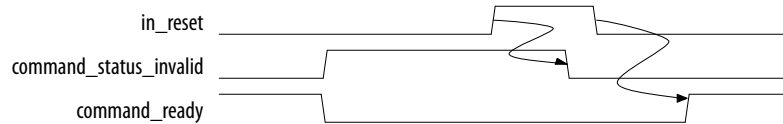


### 1.1.2.4. Command Status Interface

**Table 5. Command Status Interface**

Signal Name	Direction	Description
command_status_invalid	Output	The <code>command_status_invalid</code> asserts to indicate an error. This signal typically asserts to indicate that the length of the command specified in the command header does not match the length of the command sent. When <code>command_status_invalid</code> asserts, your application logic must assert <code>in_reset</code> to restart the Mailbox Avalon ST Client Intel FPGA IP.

**Figure 6. Reset After command\_status\_invalid Asserts**



## 1.2. Command and Response Header

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

**Figure 7. Command and Response Header Format**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ID				0	Length											0	Command/Error Code										

*Note:*

The following table describes the fields of the header.

**Table 6. Command and Response Header Format**

Header	Bit	Description
Reserved	[31:28]	Reserved.
ID	[27:24]	The command ID. Application logic creates this ID which is present in both the command and response header. Use this ID to match the command to the command response.
0	[23]	Reserved.

*continued...*



Header	Bit	Description
LENGTH	[22:12]	Number of words of data following the header. When reading the command header, the LENGTH field in the command header must match the command LENGTH of command data. When reading the response header, the LENGTH of the response header. Refer to Table 7 on page 8 for the expected command data length and response length. When you turn on <b>Enable status interface</b> , the IP asserts the <code>command_status_invalid</code> signal if the LENGTH received does not match this value.
Reserved	[11]	Reserved. Must be set to 0.
COMMAND CODE or ERROR CODE	[10:0]	For the <code>command</code> header specifies the COMMAND CODE. Refer to Table 7 on page 8, <i>Supported Commands</i> for the hex value of each COMMAND CODE. For the <code>response</code> header specifies the ERROR CODE. If the command succeeds, the ERROR CODE is 0. If the command fails, the ERROR CODE specifies one of the error codes defined in the Table 9 on page 14, <i>Error Code Responses</i> .

### 1.3. Supported Commands

Table 7. Command Code List and Description

Command	Code (Hex)	Command Data Length	Response Length	Description and Response
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. 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. Intel Agilex devices have a single voltage sensor. Consequently, the response is always one word.
GET_TEMPERATURE	19	1	Varies	The GET_TEMPERATURE command returns the temperature or temperatures of the core fabric or transceiver channel locations you specify. Use the <code>sensor_req</code> argument to specify the locations. The <code>sensor_req</code> includes the following fields: <ul style="list-style-type: none"> <li>Bits[31:28]: Reserved.</li> <li>Bits[27:16]: <code>Sensor Location</code>. Specifies the TSD location. Refer to Table 8 on page 14 for the mapping between <code>Sensor Location</code> and the bitmask position.</li> <li>Bits[15:0]: <code>Sensor mask</code>. Specifies the sensors to read for the <code>sensor location</code> specified. The response contains one word for each temperature requested. If omitted, the command reads channel 0. The least significant bit (lsb) corresponds to sensor 0. The most significant bit (msb) corresponds to channel 15.</li> </ul> The temperature returned is a signed fixed value with 8 bits below the binary point. For example, a temperature of 10°C returns 0x0000A00. A of a temperature -1.5°C returns 0xFFFFFE80.

continued...





Command	Code (Hex)	Command Data Length	Response Length	Description and Response		
				If the bitmask specifies an invalid Location, the command returns an error code which is any value in the range 0x8000000 -0x80000FF.		
RSU_IMAGE_UPDATE	5C	2	0	Triggers reconfiguration from the data source which 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. <ul style="list-style-type: none"> <li>Bit [63:32]: Reserved (write as 0).</li> <li>Bit [31:0]: The start address of an application image.</li> </ul> Returns a non-zero response if the device is already processing a configuration command.		
CONFIG_STATUS	4	0	6	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:		
				<b>Word</b>	<b>Summary</b>	<b>Description</b>
				0	State	Describes the most recent configuration related error. Returns 0 when there are no configuration errors. The error field has 2 fields: <ul style="list-style-type: none"> <li>Upper 16 bits: Major error code.</li> <li>Lower 16 bits: Minor error code.</li> </ul> Refer to , <i>Error Codes for the CONFIG_STATUS and RSU_STATUS</i> for more information.
				1	Version	The version of the RSU data structure.
				2	Pin status	<ul style="list-style-type: none"> <li>Bit [31]: Current nSTATUS output value (active low)</li> <li>Bit [30]: Detected nCONFIG input value (active low)</li> <li>Bit [29:3]: Reserved</li> <li>Bit [2:0]: The MSEL value at power up</li> </ul>
				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. <ul style="list-style-type: none"> <li>Bit [31:6]: Reserved</li> <li>Bit [5]: HPS_WARMRESET</li> <li>Bit [4]: HPS_COLDRESET</li> <li>Bit [3]: SEU_ERROR</li> <li>Bit [2]: CVP_DONE</li> <li>Bit [1]: INIT_DONE</li> <li>Bit [0]: CONF_DONE</li> </ul>
				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:		

**continued...**



Command	Code (Hex)	Command Data Length	Response Length	Description and Response		
				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 1s 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.
				4	State	Failure code of the failing image. The error field has two parts: <ul style="list-style-type: none"> <li>• Upper 16 bits: Major error code.</li> <li>• Lower 16 bits: Minor error code.</li> </ul> Returns 0 for no failures. Refer to , <i>Error Codes for the CONFIG_STATUS and RSU_STATUS</i> 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: <code>set_global_assignment -name RSU_MAX_RETRY_COUNT 3</code> . Valid values for the MAX_RETRY counter are 1-3. The actual number of available retries is MAX_RETRY -1
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: <ul style="list-style-type: none"> <li>• 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.</li> <li>• 0x00060000: Clear error status information.</li> <li>• All other values are reserved.</li> </ul> This command is not available before version 19.3 of the Intel Quartus Prime Pro Edition Software.		
QSPI_OPEN	32	0	1	Requests exclusive access to the quad SPI. 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. Returns the ALT_SDM_MBOX_RESP_DEVICE_BUSY when the quad SPI flash is busy.		

*continued...*



Command	Code (Hex)	Command Data Length	Response Length	Description and Response
				<i>Note:</i> 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.
QSPI_CLOSE	33	0	1	Closes the exclusive access to the quad SPI interface.
QSPI_SET_CS	34	1	1	Specifies one of the attached quad SPI devices via the chip select lines. Takes a one-word argument as described below: <ul style="list-style-type: none"> <li>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 nCSO[3:1] to access HPS data.</li> <li>Bits[27:0]: Reserved (write as 0). The HPS can use nCSO[3:1] to access 3 additional quad SPI devices.</li> </ul> This command is optional for the AS x4 configuration scheme. Is required for all other configuration schemes.
QSPI_READ	3A	2	N	Reads the attached quad SPI device. The maximum read size is 4 kilobytes (KB). Takes two arguments: <ul style="list-style-type: none"> <li>The quad SPI flash address (one word). The address must be word aligned. The device returns the 0x1 error code for non-aligned addresses.</li> <li>Number of words to read (one word).</li> </ul> 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. <i>Note:</i> You cannot run the QSPI_READ command while device configuration is in progress.
QSPI_WRITE	39	2+N	0	Writes data to the quad SPI device. Takes three arguments: <ul style="list-style-type: none"> <li>The flash address offset (one word). The write address must be word aligned. The device returns error code 0x3FF for non-aligned addresses.</li> <li>The number of words to write (one word).</li> <li>The data to be written (one or more words).</li> </ul> A successful write returns the OK response code. To prepare memory for writes, Intel recommends using the QSPI_ERASE command before issuing this command. <i>Note:</i> You cannot run the QSPI_WRITE command while device configuration is in progress.
QSPI_ERASE	38	2	0	Erases a sector of the quad SPI device. Takes two arguments: <ul style="list-style-type: none"> <li>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.</li> <li>The number of words to erase specified in multiples of 0x4000 words.</li> </ul> A successful erase returns the OK response code.
QSPI_READ_DEV ICE_REG	35	2	N	Reads registers from the quad SPI device. The maximum read is 8 bytes. Takes two arguments. <ul style="list-style-type: none"> <li>The opcode for the read command.</li> <li>The number of bytes to read.</li> </ul>

*continued...*



Command	Code (Hex)	Command Data Length	Response Length	Description and Response
				A successful read returns the OK response code followed by the data read from the device. Pads data that is not a multiple of 4 bytes to the next word boundary.
QSPI_WRITE_DEVICE_REG	36	2+N	0	<p>Writes to registers of the quad SPI. The maximum write is 8 bytes. Takes three arguments:</p> <ul style="list-style-type: none"><li>• The opcode for the write command.</li><li>• The number of bytes to write.</li><li>• The data to write.</li></ul> <p>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:</p> <p>Header: 0x00003036 Opcode: 0x000000DC Number of bytes to write: 0x00000004 Flash address: 0x0000FF04</p> <p>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.</p>
QSPI_SEND_DEVICE_OP	37	1	0	<p>Sends a command opcode to the quad SPI. Takes one argument:</p> <ul style="list-style-type: none"><li>• The opcode to send the quad SPI device.</li></ul> <p>A successful command returns the OK response code.</p>

### Intel Agilex E-tile Transceiver Local Temperature Sensor Design Consideration

The E-tile transceivers include four local temperature sensors distributed across the transceiver die. The temperature readings may vary due to the activity level of each channel. Intel recommends that you use the highest temperature as the trip point for the thermal control.

#### Related Information

[Intel Agilex Power Management User Guide](#)

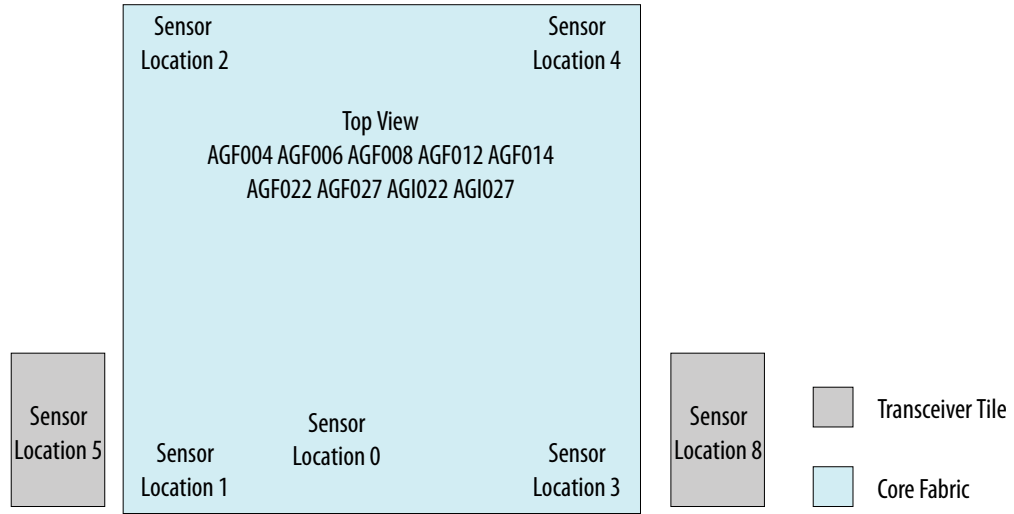
## 1.4. Temperature Sensor Locations

The number and locations of temperature sensors varies with the size of the device and the transceiver tiles available.

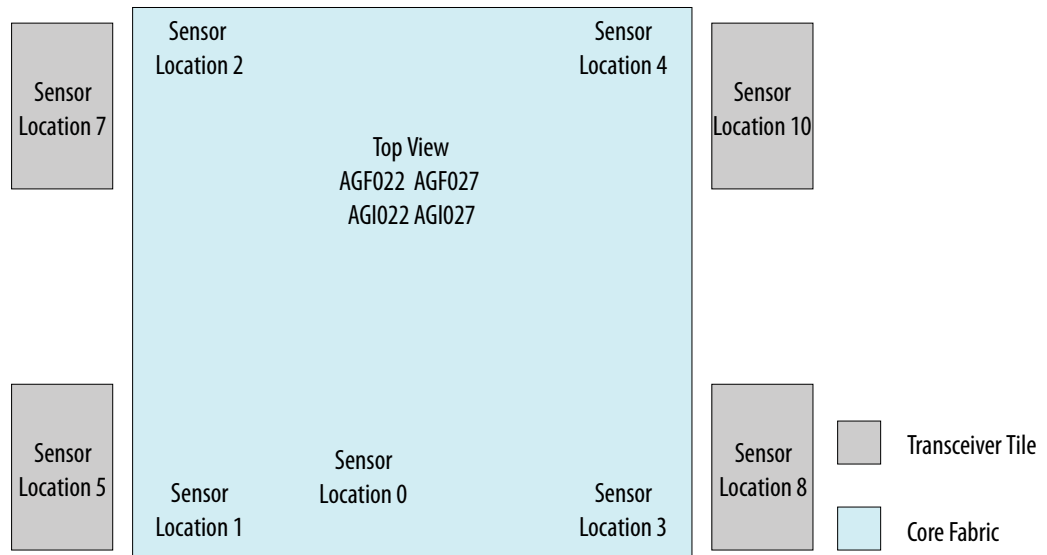
The following figures show the locations of sensors for Intel Agilex devices with two and four transceiver tiles.



**Figure 8. Temperature Sensing Diode Locations- Two Transceiver Tiles**



**Figure 9. Temperature Sensing Diode Locations - Four Transceiver Tiles**

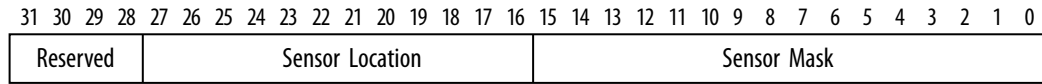


Use the `sensor_req` argument to the `GET_TEMPERATURE` command to specify the location or locations to read. E-tile transceivers include four local temperature sensor diodes distributed across the transceiver die. The temperature readings may vary due to the activity level of each sensor location. Consequently, Intel recommends that you set the `sensor_mask` field to `0x000F` for E-tile devices and use the highest temperature as the trigger point for the thermal control.

P-tile devices and the core fabric have a single temperature sensing diode at each location. Consequently, you set the `sensor_mask` to `0x0001` which is the default setting.



**Figure 10. Bitmask for GET\_TEMPERATURE Command**



**Table 8. Intel Agilex Temperature Sensor Channels and Locations**

Sensor Location	Sensor Location Bit Position <sup>(1)</sup>	AGF004 AGF006 AGF008 AGF012 AGF014	AGF022 AGF027 AGI022 AGI027
Core fabric, bottom center right	16	Supported	Supported
Core fabric, bottom left corner	17	Supported	Supported
Core fabric, top left corner	18	Supported	Supported
Core fabric, bottom right corner	19	Supported	Supported
Core fabric, top right corner	20	Supported	Supported
XCVR, bottom left corner	21	Supported	Supported
XCVR, top left corner	23	Not supported	Supported
XCVR, bottom right corner	24	Supported	Supported
XCVR, top right corner	26	Not supported	Supported

## 1.5. Error Codes

The response packet header includes the error code when the command fails.

**Table 9. 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 command is incorrectly formatted.
2	UNKNOWN_BR	Indicates that the command code is not understood.
3	UNKNOWN	Indicates that the currently loaded firmware cannot decode the command code.
4	INVALID_LENGTH	Indicates that 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 requesting quad SPI or SD MMC access does not have exclusive access.
9	INVALID_ADDRESS	The address is invalid. This error indicates one of the following conditions:

*continued...*

<sup>(1)</sup> Bit 22 and bit 25 are reserved and set to 0.



Value (Hex)	Error Code Response	Description
		<ul style="list-style-type: none"> <li>An unaligned address</li> <li>An address range problem</li> <li>A read permission problem</li> </ul>
A	AUTHENTICATION_FAIL	Indicates the configuration bitstream signature authentication failure.
B	TIMEOUT	The command timed out.
C	HW_NOT_READY	The hardware is not ready. Can indicate either an initialization or configuration problem.
100	NOT_CONFIGURED	Indicates that the device is not configured.
1FF	ALT_SDM_MBOX_RESP_DEVICE_BUSY	Indicates that the device is busy.
2FF	ALT_SDM_MBOX_RESP_NO_VALID_RESP_AVAILABLE	Indicates that there is no valid response available.
3FF	ALT_SDM_MBOX_RESP_ERROR	General Error.

## 1.6. Mailbox Avalon ST Client Intel FPGA IP User Guide Document 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
19.3	1.0.0	<a href="#">Mailbox Avalon ST Client Intel FPGA IP User Guide</a>

## 1.7. Document Revision History for the Mailbox Avalon ST Client Intel FPGA IP User Guide

Document Version	Intel Quartus Prime Version	Changes
2020.04.13	20.1	<p>Made the following changes:</p> <ul style="list-style-type: none"> <li>Added information about the temperature sensors for the GET_TEMPERATURE command, including figures illustrating TSD locations.</li> <li>Added RSU_NOTIFY command in the <i>Command Code List and Description</i> table.</li> <li>Updated the <i>Error Codes</i> table: <ul style="list-style-type: none"> <li>Renamed INVALID_COMMAND_PARAMETERS to INVALID_LENGTH.</li> <li>Changed COMMAND_INVALID_ON_SOURCE hex value from 5 to 6.</li> <li>Changed CLIENT_ID_NO_MATCH hex value from 6 to 8.</li> <li>Changed INVALID_ADDRESS hex value from 7 to 9.</li> <li>Added AUTHENTICATION_FAIL command.</li> <li>Changed TIMEOUT hex value from 8 to B.</li> <li>Changed HW_NOT_READY hex value from 9 to C.</li> </ul> </li> </ul>
2019.09.30	19.3	Initial release.