1.1. Feature Description.................................................................3
1.2. Commands and Error Codes......................................................4
   1.2.1. Commands........................................................................5
   1.2.2. Error Code Responses......................................................8
1.3. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-MM Interface................................. 9
1.4. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-Memory Map................................. 9
   1.4.1. Interrupt Enable Register.................................................9
   1.4.2. Interrupt Status Register.................................................10
1.5. Using the Mailbox Client Intel Stratix 10 FPGA IP Core.................................................... 10
1.6. Mailbox Client Intel Stratix 10 FPGA IP Core Use Case Examples.................................... 11
   1.6.1. Example 1: Reading Intel Stratix 10 IDCODE and Voltage.........................11
   1.6.2. Example 2: Read and Write EPCQ-L or QSPI Devices..........................12
1.7. Document Revision History for the Mailbox Client Intel Stratix 10 FPGA IP Core
    User Guide....................................................................................15
1. Mailbox Client Intel® Stratix® 10 FPGA IP Core User Guide

The Mailbox Client Intel® Stratix® 10 FPGA IP core converts the Avalon-ST interface to Avalon-MM interface for clients such as JTAG, FPGA mailbox, HPS mailbox, to communicate with the secure device manager (SDM) in Intel Stratix 10 devices.

Related Information

• Avalon Interface Specifications
• Secure Device Manager in Intel Stratix 10 Devices

1.1. Feature Description

The Mailbox Client Intel Stratix 10 FPGA IP core features commands and responses for communication with the SDM.

The Mailbox Client Intel Stratix 10 FPGA IP core supports:

• Command and response access
• Urgent access (1)
• Depth-adjustable FIFO to buffer command, response and urgent packet
• Configurable interrupt source:
  — Indication that command FIFO is not full and client can send more commands
  — Indication that there is valid response in FIFO and client can begin to read out the response.

(1) This feature will be available in the future releases of Intel Quartus® Prime software.
Figure 1. **Mailbox Client Intel Stratix 10 FPGA IP Core Block Diagram**

Note: Refer to the Secure Device Manager topic of the Intel Stratix 10 Configuration User Guide for the SDM block diagram showing the connection of the Mailbox Client Intel Stratix 10 FPGA IP core with the mailbox block in SDM.

![Mailbox Client Intel Stratix 10 FPGA IP Core Block Diagram](image)

**Related Information**
Secure Device Manager of the Intel Stratix 10 Devices

### 1.2. Commands and Error Codes

The remote system upgrade host communicates with the SDM using command and response packets via the Mailbox Client Intel Stratix 10 FPGA IP.

**Figure 2. Command and Error Code Header Format**

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

The following table describes the fields of the header command.

**Table 1. Mailbox Client Intel Stratix 10 FPGA IP Command and Error Code Header Description**

<table>
<thead>
<tr>
<th>Header</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>[27:24]</td>
<td>The command ID. The response header returns the ID specified in the command header. Set different IDs in each command to match responses with commands.</td>
</tr>
<tr>
<td>0</td>
<td>[23]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Length</td>
<td>[22:12]</td>
<td>Number of words of arguments following the header.</td>
</tr>
</tbody>
</table>
# 1.2.1. Commands

## Table 2. Command List and Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Number of Command Word (2)</th>
<th>Number of Response Word (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOOP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Sends an OK status response.</td>
</tr>
<tr>
<td>GET_IDCODE</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>The response contains one argument which is the JTAG IDCODE for the device.</td>
</tr>
<tr>
<td>GET_CHIPID</td>
<td>12</td>
<td>0</td>
<td>2</td>
<td>The response contains 64-bit CHIPID value with the least significant word first.</td>
</tr>
<tr>
<td>GET_USERCODE</td>
<td>13</td>
<td>0</td>
<td>1</td>
<td>The response contains one argument which is the 32-bit JTAG USERCODE provided to the device by the configuration bitstream.</td>
</tr>
<tr>
<td>GET_VOLTAGE</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>Command has a single argument which is a bitmask of which channels to read. Bit 0 is set to read channel 0, bit 1 to read channel 1 and so on. The response contains one word argument for each bit set in the bitmask. Each return value is a 32-bit value. The voltage returned is an unsigned fixed point number with 16 bits below the binary point. Example: A voltage of 0.75V returns 0x0000C000.</td>
</tr>
<tr>
<td>GET_TEMPERATURE</td>
<td>19</td>
<td>1</td>
<td>1</td>
<td>Command has a single argument which is a bitmask indicating which temperature sensors to read. The response contains one word for each channel temperature requested. The temperature returned as a signed fixed value with 8 bits below the binary point.  • Channel 0: Samples the temperature value from the core fabric. • Channels 1 to 6: Samples the temperature value from the specified transceiver tile. • Channels 7 to 8: Samples the temperature value from the high-bandwidth DRAM memory (HBM2) stacks. Example: A Temperature of 10°C returns 0x00000A00 and a of temperature -1.5°C returns 0xFFFFFE80.</td>
</tr>
<tr>
<td>RSU_IMAGE_UPDATE</td>
<td>5C</td>
<td>2</td>
<td>0</td>
<td>Triggers reconfiguration from the data source selected by MSEL setting for initial configuration. This command takes an optional 64-bit argument to specify the reconfiguration data address in the flash. If the argument is not provided then its value is assumed to be 0. • Bit [63:32]: Reserved (write as 0). • Bit [31:0]: The start address of an application image. Returns non-zero response if the device is already processing a configuration. Note: You can use this command to check the configuration status during configuration and after it has completed.</td>
</tr>
<tr>
<td>CONFIG_STATUS</td>
<td>4</td>
<td>0</td>
<td>1</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:</td>
</tr>
</tbody>
</table>

(2) The number does not include the command and response header.

(3) The availability of each transceiver tile varies among devices. For the temperature sensor channel numbers, refer to the related information.
<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Number of Command Word (2)</th>
<th>Number of Response Word (2)</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>State</td>
<td>Most recent configuration related error. 0 if there has been no configuration errors.</td>
</tr>
<tr>
<td>1</td>
<td>Version</td>
<td>0 for this version.</td>
</tr>
</tbody>
</table>
| 2    | Pin status | • Bit [31]: Current nSTATUS output value (active low).  
• Bit [30]: Detected nCONFIG input value (active low).  
• Bit [29:8]: Reserved.  
• Bit [7:0]: The MSEL value latched by the device at power up. |
| 3    | Soft function status | Contains the value of each of the soft functions, regardless if the function has been assigned to an SDM pin.  
• Bit [31:4]: Reserved  
• 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 is no error. |
| 5    | Error details | Contains the error details. Returns 0 if there is no error. |

**RSU_STATUS**  
5B  0  6  
Reports the current remote system upgrade status. This command returns the following responses:

<table>
<thead>
<tr>
<th>Word</th>
<th>Summary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Current image</td>
<td>Flash offset of the currently running application image.</td>
</tr>
<tr>
<td>2-3</td>
<td>Last failing image</td>
<td>Flash offset of the last failing application image. The value of all 1s indicates no failing images. If no failing images, the following words do not contain meaningful data.</td>
</tr>
<tr>
<td>4</td>
<td>State</td>
<td>Failure code of the last failing image.</td>
</tr>
<tr>
<td>5</td>
<td>Version</td>
<td>Contains the value of each of the soft functions, whether that function has been put on an SDM pin or not.</td>
</tr>
<tr>
<td>6</td>
<td>Error location</td>
<td>Contains the error location of the last failing image. Returns 0 if there is no error.</td>
</tr>
<tr>
<td>7</td>
<td>Error details</td>
<td>Contains the error details of the last failing image. Returns 0 if there is no error.</td>
</tr>
</tbody>
</table>

(2) The number does not include the command and response header.
<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Number of Command Word (2)</th>
<th>Number of Response Word (2)</th>
<th>Description</th>
</tr>
</thead>
</table>
| QSPI_OPEN     | 32         | 0                          | 1                          | Requests exclusive access to the quad-serial peripheral interface (QSPI). If the SDM accepts the request (if the QSPI is not already in use or the SDM is not in the process of a device configuration), it returns the OK response, else it returns the error response.  
Note: The exclusive access is granted only to the client using this mailbox. Other clients are not be able to access the QSPI until it is closed by this client. |
| QSPI_CLOSE    | 33         | 0                          | 1                          | Closes the exclusive access to the QSPI interface.                                                                                                                                                        |
| QSPI_SET_CS   | 34         | 1                          | 1                          | Select which of the attached QSPI device via the chip select lines. Takes on one word argument as described below:  
• Bit [31:28]: Flash device to be selected. Bit setting 0000 is used to select flash that attached to nCSO[0].  
• Bit [27:0]: Reserved (write as 0).                                                                                                           |
| QSPI_READ     | 3A         | 2                          | N                          | Reads the attached QSPI device and takes two parameters:  
• The flash address offset from where you want to start reading from the QSPI device (one word).  
• Number of words to read (one word).  
A successful response returns an OK response code followed by the data read from the QSPI device. A failure response is either:  
• Returns an error code  
• Returns OK but partial of the data read from QSPI device is incorrect.  
Note: The maximum transfer size is limited to 4K bytes and cannot be called while a configuration is in progress. |
| QSPI_WRITE    | 39         | 2+N                        | 0                          | Writes data on the attached QSPI device and takes three parameters:  
• The flash address offset from where you want the command to start writing to the QSPI device (one word).  
• The number of words to write (one word).  
• The data to be written (one or more words).  
A successful write returns an OK response code.  
The client may need to issue QSPI_ERASE command before issuing this command to prepare the memory for writing.  
Note: The maximum transfer size is limited to 4K bytes and cannot be called while a configuration is in progress. |
| QSPI_ERASE    | 38         | 2                          | 0                          | Erases sector on the attached QSPI device and takes two parameters:  
• The flash address offset within the device to start erasing from (one word). The address must be the start address of a sector within the flash memory.  
• The number of bytes to erase. The erase size is the multiple of 64K bytes.  
A successful erase returns an OK response code.                                                                                                     |
| QSPI_READDEVICE_REG | 35    | 2                          | N                          | Reads registers from the attached QSPI device and takes two parameters:  
• The opcode for the read command.  
• The number of bytes to read (the maximum size is 8 bytes). |

(2) The number does not include the command and response header.
A successful read returns an OK response code followed by the data read from the device. If the data is not an exact multiple of 4 bytes then it is padded with 0 bytes until the next word boundary.

<table>
<thead>
<tr>
<th>Command</th>
<th>Code (Hex)</th>
<th>Number of Command Word (2)</th>
<th>Number of Response Word (2)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSPI_WRITE_DEVICE_REG</td>
<td>36</td>
<td>2+N</td>
<td>0</td>
<td>Writes to registers on the attached QSPI and takes three arguments:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The opcode for the write command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The number of bytes to write (the maximum size is 8 bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The data to write (maximum 2 words, padded with 0 to word boundary).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A successful write returns an OK response code.</td>
</tr>
<tr>
<td>QSPI_SEND_DEVICE_OP</td>
<td>37</td>
<td>1</td>
<td>0</td>
<td>Sends a command opcode to the QSPI and takes one argument:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The opcode to send the attached QSPI device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A successful command returns an OK response code.</td>
</tr>
</tbody>
</table>

### 1.2.2. Error Code Responses

#### Table 3. Mailbox Client Intel Stratix 10 FPGA IP Error Code Responses and Description

<table>
<thead>
<tr>
<th>Value (Hex)</th>
<th>Error Code Response</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
<td>Indicates that the command completed successfully. Depending on the command delivered to the Mailbox Client, the response error code may not be sufficient to ensure that the operation completed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>INVALID_COMMAND</td>
<td>Indicates that the command is in an incorrect format.</td>
</tr>
<tr>
<td>2</td>
<td>UNKNOWN_BR</td>
<td>Indicates that the command code is not understood. This error may occur if you have deselected the Use the factory default helper image on the Programmer Tools -&gt; Options menu.</td>
</tr>
<tr>
<td>3</td>
<td>UNKNOWN</td>
<td>Indicates that the command code is not understood by the currently loaded firmware.</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.</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>
</tbody>
</table>

(2) The number does not include the command and response header.
1.3. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-MM Interface

Table 4. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-MM Interface

<table>
<thead>
<tr>
<th>Signal Role</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avmm_address</td>
<td>4</td>
<td>Avalon-MM address</td>
</tr>
<tr>
<td>avmm_write</td>
<td>1</td>
<td>Avalon-MM write request</td>
</tr>
<tr>
<td>avmm_read</td>
<td>1</td>
<td>Avalon-MM read request</td>
</tr>
<tr>
<td>avmm_writedata</td>
<td>32</td>
<td>Avalon-MM writedata bus</td>
</tr>
<tr>
<td>avmm_readdata</td>
<td>32</td>
<td>Avalon-MM readdata bus</td>
</tr>
<tr>
<td>avmm_readdatavalid</td>
<td>1</td>
<td>Avalon-MM readdata valid</td>
</tr>
</tbody>
</table>

1.4. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-Memory Map

Table 5. Mailbox Client Intel Stratix 10 FPGA IP Core Avalon-Memory Map

<table>
<thead>
<tr>
<th>Offset (word)</th>
<th>R/W</th>
<th>31</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base + 0</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 1</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 2</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 3</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 4</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 5</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 6</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 7</td>
<td>R/W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base + 8</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.4.1. Interrupt Enable Register

Table 6. Interrupt Enable Register

Having the enable bit cleared disregards the corresponding interrupt status bit from causing interrupt output assertion (IRQ).

Note: These enable bits does not prevent the value of interrupt status bit from showing up in ISR, it only prevents the interrupt status bit from causing interrupt output assertion.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Fields</th>
<th>Access</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>EN_CMD_FIFO_NOT_FU LL</td>
<td>R/W</td>
<td>0x0</td>
<td>The enable bit interrupt of command FIFO is not full.</td>
</tr>
</tbody>
</table>
## 1.4.2. Interrupt Status Register

### Table 7. Interrupt Status Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Fields</th>
<th>Access</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CMD_FIFO_NOT_FULL</td>
<td>R</td>
<td>0x0</td>
<td>Command FIFO is not full interrupt. (4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Indicates command FIFO is not full, the client can send more data into it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Indicates FIFO is full.</td>
</tr>
<tr>
<td>0</td>
<td>DATA_VALID</td>
<td>R</td>
<td>0x0</td>
<td>Data valid interrupt. (4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Indicates data exist in FIFO, master can read it out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Indicates the FIFO is full.</td>
</tr>
</tbody>
</table>

## 1.5. Using the Mailbox Client Intel Stratix 10 FPGA IP Core

### Writing Command Packet

Command with one word (header only): Write to base + 1.

Command with more than one word (header + argument):

1. Write the header & argument (except the last one) to base + 0
2. Write last argument to base + 1

You can read from base + 2 which shows the remaining available free space in the FIFO for command. The FIFO fills up when the SDM is busy.

The behavior of the IP is undefined if you write to base + 0 and base + 1 while the FIFO is full. The write data is discarded.

### Reading Response packet

1. Read base + 6 to check SOP, EOP & fill level of response FIFO. Proceed to step 2 if SOP = 1.

   For example, a return value of 0x00000007 indicates that a word is returned by SDM to fill up the response FIFO where the word represents the start of packet and also is the end of packet.

2. Read base + 5 to read response header. The length value contains the number (eg: n) of argument a response has.
3. Read base + 5 n times to retrieve all the response data.

(4) This bit is cleared by operations on the FIFO. You do not need to clear this bit manually.
1.6. Mailbox Client Intel Stratix 10 FPGA IP Core Use Case Examples

To use the Mailbox Client Intel Stratix 10 FPGA IP Core, an Avalon master is required and the simplest Avalon master is JTAG-to-Avalon Master.

The following examples are based on a Platform Designer system consist of JTAG-to-Avalon Master and the Mailbox Client Intel Stratix 10 FPGA IP core.

1.6.1. Example 1: Reading Intel Stratix 10 IDCODE and Voltage

```bash
#set base address according to Platform Designer system
set base 0x00000000
#set the variables to their respective offset
set b0 [expr {$base + 0x0}]
set b1 [expr {$base + 0x4}]
set b2 [expr {$base + 0x8}]
set b3 [expr {$base + 0xc}]
set b4 [expr {$base + 0x10}]
set b5 [expr {$base + 0x14}]
set b6 [expr {$base + 0x18}]
set b7 [expr {$base + 0x1c}]
set b8 [expr {$base + 0x20}]

#assign variable mp to the string that is the 0th element in the list returned by get_service_paths master
set mp [lindex [get_service_paths master] 0]

#procedure to open the connection to the master module
proc start_service_master {} {
    global omp
    set omp [claim_service master $mp demo]
}

#procedure to close the connection to the master module
proc stop_service_master {} {
    global omp
    close_service master $omp
    set omp {}
}

#calling the start_service_master procedure
start_service_master

#writing a command without argument
#writing the command header to offset 1 of the SDM Mailbox IP (eg Get_IDCODE)
master_write_32 $omp $b1 0x00000010

#read offset 8 (Interrupt service register) to determine if there is data in the FIFO
master_read_32 $omp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $omp $b6 1

#read offset 5 for the first packet of data (this will be the response header), the length field will notify user how many packets of argument that is to follow
master_read_32 $omp $b5 1

#read offset 5 again to retrieve the response argument (in this case, this command only has one response argument, which is the IDCODE)
master_read_32 $omp $b5 1

#writing a command with argument (eg: GET_VOLTAGE)
```
#writing the command header to offset 0 of the SDM Mailbox IP
master_write_32 $omp $b0 0x00001018

#writing the command argument to offset 1 (the voltage of interest is of channel 0)
master_write_32 $omp $b1 0x00000001

#read offset 8 (Interrupt service register) to determine if there is data in the FIFO
master_read_32 $omp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $omp $b6 1

#read offset 5 for the first packet of data (this will be the response header), the length field will notify user how many packets of argument that is to follow
master_read_32 $omp $b5 1

#read offset 5 again to retrieve the response argument (in this case, this command only has one response argument, which is the voltage of channel 0)
master_read_32 $omp $b5 1

stop_service_master

### 1.6.2. Example 2: Read and Write EPCQ-L or QSPI Devices

The following example shows the sequences and commands to read 1 word of data stored in the EPCQ-L or QSPI device.

1. Request an access to QSPI interfaces using QSPI_OPEN command:

   #writing the command header to offset 1 of the SDM Mailbox IP by using QSPI_OPEN command code
   master_write_32 $mp $b1 0x00000032

   #read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
   master_read_32 $mp $b8 1

   #read offset 6 for SOP and EOP of response packet
   master_read_32 $mp $b6 1

   #read offset 5 for the first packet of data (this will be the response header). You are expecting to get the response packet to return OK(0x0000000)
   master_read_32 $mp $b5 1

2. Select the QSPI device using QSPI_SET_CS command:

   #writing the command header to offset 0 of the SDM Mailbox IP by using the QSPI_CS command code
   master_write_32 $mp $b0 0x000001034

   #writing the command argument to offset 1 (select QSPI flash attached to nCSO[0])
   master_write_32 $mp $b1 0x00000000

   #read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
   master_read_32 $mp $b8 1

   #read offset 6 for SOP and EOP of response packet
   master_read_32 $mp $b6 1

   #read offset 5 for the first packet of data (this will be the response header), the length field will notify user how many packets of argument that is to follow.
   master_read_32 $mp $b5 1
3. Start to read from selected QSPI device using QSPI_READ command:

```plaintext
#read offset 5 again to retrieve the response argument (in this case, this command only has one response argument, which is one word of data 0x62294895 stored in address offset 0x00000000)
master_read_32 $mp $b5 1

#writing the command header to offset 0 of the SDM Mailbox IP (Specify the QSPI_READ command code)
master_write_32 $mp $b0 0x0000203A

#writing the command argument to offset 0 (Specify the flash address offset in one word)
master_write_32 $mp $b0 0x00000000

#writing the command argument to offset 1 (reading one word from flash address offset 0x00000000)
master_write_32 $mp $b1 0x00000001

#read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
master_read_32 $mp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $mp $b6 1

#read offset 5 for the first packet of data (this will be the response header), the length field will notify user how many packets of argument that is to follow. You are expecting to get the response packet to return OK(0x0000000)
master_read_32 $mp $b5 1

#read offset 5 again to retrieve the response argument (in this case, this command only has one response argument, which is one word of data 0x62294895 stored in address offset 0x00000000)
master_read_32 $mp $b5 1
```

4. Close the access to the QSPI interfaces using QSPI_CLOSE command:

```plaintext
#writing the command header to offset 1 of the SDM Mailbox IP by using QSPI_CLOSE command code
master_write_32 $mp $b1 0x00000033

#read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
master_read_32 $mp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $mp $b6 1
```

The following example shows the sequences and commands to write 1 word of data to an EPCQ-L or QSPI flash device.

1. Repeat step 1 and 2 in the example above to request exclusive access to QSPI interface and select QSPI flash attached to nCSO[0].

2. Ensure you erase the content stored in the address that you would like to write to:
   a. Set the Write enable latch bit to 1:

```plaintext
#writing the command header to offset 0 of the SDM Mailbox IP by using QSPI_SEND_DEVICE_OP command code
master_write_32 $mp $b1 0x00001037

#writing the command argument to offset 1 (the opcode for write enable 0x00000006)
master_write_32 $mp $b1 0x00000006
```
b. Start erasing the contents:

```plaintext
#writing the command header to offset 0 of the SDM Mailbox IP (Specify the QSPI_ERASE command code)
master_write_32 $mp $b0 0x00002038

#writing the command argument to offset 0 (Specify the flash address offset in one word)
master_write_32 $mp $b0 0x00800000

#writing the command argument to offset 1 (Specify the number bytes to erase in the multiple of 64K bytes)
master_write_32 $mp $b1 0x00004000

#read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
master_read_32 $mp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $mp $b6 1

#read offset 5 for the first packet of data (this will be the response header). You are expecting to get the response packet to return OK(0x0000000)
master_read_32 $mp $b5 1
```

3. Start to write to the selected QSPI device using QSPI_WRITE command.

```plaintext
#writing the command header to offset 0 of the SDM Mailbox IP (Specify the QSPI_WRITE command code)
master_write_32 $mp $b0 0x00003039

#writing the command argument to offset 0 (Specify the flash address offset in one word)
master_write_32 $mp $b0 0x00800000

#writing the command argument to offset 0 (To write a single word, specify to 1)
master_write_32 $mp $b0 0x00000001

#writing the command argument to offset 1 (Specify the data to be written for example like 0xdeadbeef)
master_write_32 $mp $b1 0xdeadbeef

#read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
master_read_32 $mp $b8 1

#read offset 6 for SOP and EOP of response packet
master_read_32 $mp $b6 1

#read offset 5 for the first packet of data (this will be the response header). You are expecting to get the response packet to return OK(0x0000000). This means that the data is written into the flash.
master_read_32 $mp $b5 1
```
Alternatively, you can do a QSPI_READ at address 0x0080000 to verify the data has been written properly into the targeted flash address.

4. Close the access to the QSPI interfaces using QSPI_CLOSE command:

```plaintext
# writing the command header to offset 1 of the SDM Mailbox IP by using QSPI_CLOSE command code
master_write_32 $mp $b1 0x00000033

# read offset 8 (Interrupt service register) to determine if there is valid data in the FIFO
master_read_32 $mp $b8 1

# read offset 6 for SOP and EOP of response packet
master_read_32 $mp $b6 1
```

1.7. Document Revision History for the Mailbox Client Intel Stratix 10 FPGA IP Core User Guide

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 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_SD_MBOX_RESP_DEVICE_BUSY  
|                  |   — ALT_SD_MBOX_RESP_NO_VALID_RESP_AVAILABLE  
|                  |   — ALT_SD_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. |