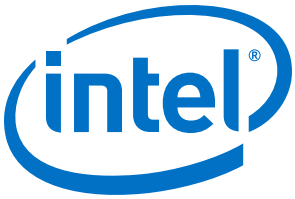


Board Management Controller User Guide

Intel FPGA Programmable Acceleration Card N3000-N

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.3.1**

Subscribe
Send Feedback

UG-20281 | 2020.09.08
Latest document on the web: **PDF** | **HTML**

Contents

1. Introduction

1.1. About this Document

This document describes:

- Features and functionality of the Intel® MAX® 10 Board Management Controller (BMC).
- How to read telemetry data on the Intel FPGA Programmable Acceleration Card N3000-N using the Platform Level Data Model (PLDM) over Management Component Transport Protocol (MCTP) SMBus and I^2C SMBus.
- Intel MAX 10 root of trust (RoT) and secure remote system update.

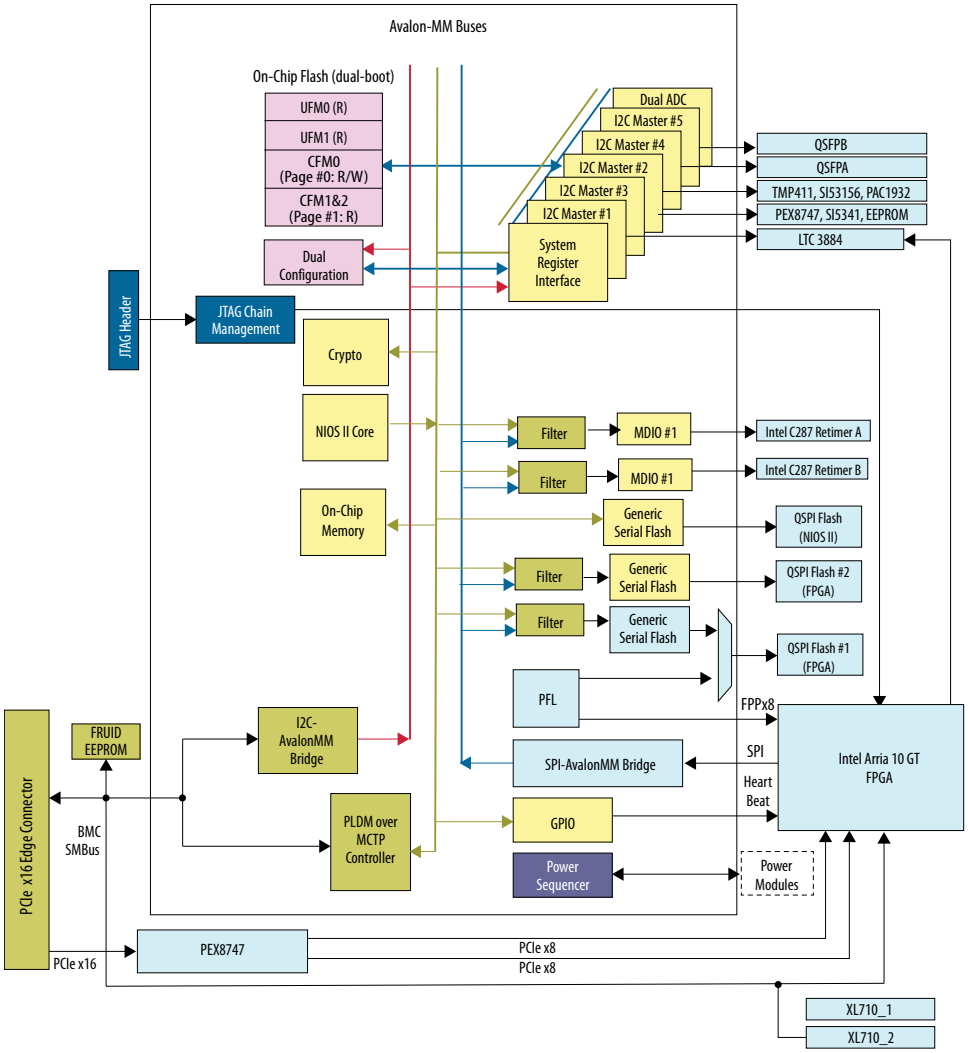
1.2. Overview

The Intel MAX 10 BMC is responsible for controlling, monitoring and granting access to board features. The Intel MAX 10 BMC interfaces with on-board sensors, the FPGA and the flash, and manages power-on/power-off sequences, FPGA configuration and telemetry data polling. You can communicate with the BMC using the PLDM version 1.1.1 protocol. The BMC firmware is field upgradeable over PCIe using the remote system update feature.

Features of BMC

- Acts as a Root of Trust (RoT) and enables the secure update features of the Intel FPGA PAC N3000-N.
- Controls firmware and FPGA flash updates over PCIe.
- Manages FPGA configuration.
- Updates the C827 Ethernet Retimer device firmware.
- Controls power sequencing and fault detection with automatic shut-down protection.
- Controls power and resets on the board.
- Interfaces with sensors, FPGA flash and QSFPs.
- Monitors telemetry data (board temperature, voltage, and current) and provides protective action when readings are beyond the critical threshold.
  - Reports telemetry data to host BMC via PLDM over MCTP via SMBus or through I^2C.
  - Supports PLDM over MCTP via PCIe SMBus. 0xCE is the 8-bit slave address.
  - Supports I^2C. 0xBC is the 8-bit slave address.
- Accesses the EEPROM containing the Ethernet MAC address of the Intel Ethernet Controller XL710-BM2.

Figure 1. Block Diagram



1.3. Root of Trust (RoT)

The Intel MAX 10 BMC acts as a Root of Trust (RoT) and enables the secure remote system update feature of the Intel FPGA PAC N3000-N. The RoT includes features that may help prevent the following:

- Loading or executing of unauthorized code or designs
- Disruptive operations attempted by unprivileged software, privileged software, or the host BMC
- Unintended execution of older code or designs with known bugs or vulnerabilities by enabling the BMC to revoke authorization

The Intel FPGA PAC N3000-N BMC also enforces several other security policies relating to access through various interfaces,as well as protecting the on-board flash through write rate limitation. Please refer to the *Security User Guide: Intel FPGA Programmable Acceleration Card N3000* for information on RoT and security features of the Intel FPGA PAC N3000-N.

Related Information

Security User Guide: Intel FPGA Programmable Acceleration Card N3000

1.4. Secure Remote System Update

The BMC supports Secure RSU for the Intel MAX 10 BMC Nios® firmware and RTL image and Intel Arria® 10 FPGA image updates with authentication and integrity checks. The Nios firmware is in charge of authenticating the image during the update process. The updates are pushed over the PCIe interface to the Intel Arria 10 GT FPGA, which in turn writes it over the Intel Arria 10 FPGA SPI master to Intel MAX 10 FPGA SPI slave. A temporary flash area called staging area stores any type of authentication bitstream through SPI interface.

The BMC RoT design contains the cryptographic module which implements SHA2 256 bit hash verification function and ECDSA 256 P 256 signature verification function to authenticate the keys and user image. Nios firmware uses the cryptographic module to authenticate the signed user image in the staging area. If authentication passes, the Nios firmware copies the user image to user flash area. If the authentication fails, the Nios firmware reports an error. Please refer to the *Security User Guide: Intel FPGA Programmable Acceleration Card N3000* for information on RoT and security features of the Intel FPGA PAC N3000-N.

Related Information

Security User Guide: Intel FPGA Programmable Acceleration Card N3000

1.5. Power Sequence Management

The BMC Power sequencer state machine manages the Intel FPGA PAC N3000-N power-on and power-off sequences. The Intel MAX 10 power-up flow covers the entire process including Intel MAX 10 boot-up, Nios boot-up, and power sequence management for FPGA configuration. The host must check the build versions of both the Intel MAX 10 and FPGA, as well as the Nios status after every power-cycle, and take corresponding actions in case the Intel FPGA PAC N3000-N runs into corner cases such as an Intel MAX 10 or FPGA factory build load failure or Nios boot up failure. The BMC protects the Intel FPGA PAC N3000-N by shutting down power to the card under the following conditions:

- 12 V Auxiliary or PCIe edge supply voltage is below 10.46 V
- FPGA core temperature reaches 100°C
- Board temperature reaches 100°C

Note: The LEDs blink yellow when the Intel FPGA PAC N3000-N shuts down.

1.6. Board Monitoring Through Sensors

The Intel MAX 10 BMC monitors voltage, current and temperature of various components on the Intel FPGA PAC N3000-N. Host BMC can access the telemetry data through PCIe SMBus. The PCIe SMBus between host BMC and Intel FPGA PAC N3000-N Intel MAX 10 BMC is shared by both the PLDM over MCTP via SMBus endpoint and Standard I^2C slave to Avalon-MM interface (read-only).

2. Board Monitoring through PLDM over MCTP SMBus

The BMC on the Intel FPGA PAC N3000-N communicates with a server BMC over the PCIe* SMBus.

The MCTP controller supports Platform Level Data Model (PLDM) over Management Component Transport Protocol (MCTP) stack. MCTP endpoint 8-bit slave address is 0xCE by default. This slave address can be modified and reprogrammed into a corresponding section of external FPGA Quad SPI flash via in-band way if necessary.

The Intel FPGA PAC N3000-N BMC supports a subset of the PLDM and MCTP commands to enable a server BMC to obtain sensor data such as voltage, current and temperature.

Note: PLDM over MCTP SMBus endpoint is supported. PLDM over MCTP via native PCIe is not supported.

SMBus device category:

- **Fixed not Discoverable** device is supported by default. Other device categories are not supported.

ACK-Poll is supported with SMBus default slave address 0xCE.

The BMC supports:

- 1.1.1 Version of the Management Component Transport Protocol (MCTP) Base Specification (DMTF specification DSP0236)
- 1.1.0 Version of the MCTP I2C/SMBus Transport Binding Specification (DSP0237)
- 1.0.0 Version of the MCTP I2C/SMBus Transport Binding Specification (DSP0240)
- 1.2.0 Version of the MCTP I2C/SMBus Transport Binding Specification (DSP0248)

Related Information

Distributed Management Task Force (DMTF) Specifications

2.1. SMBus Interface Speed

The Intel FPGA PAC N3000-N implementation supports SMBus transactions at 100 KHz by default.

2.2. MCTP Packetization Support

MCTP Definitions

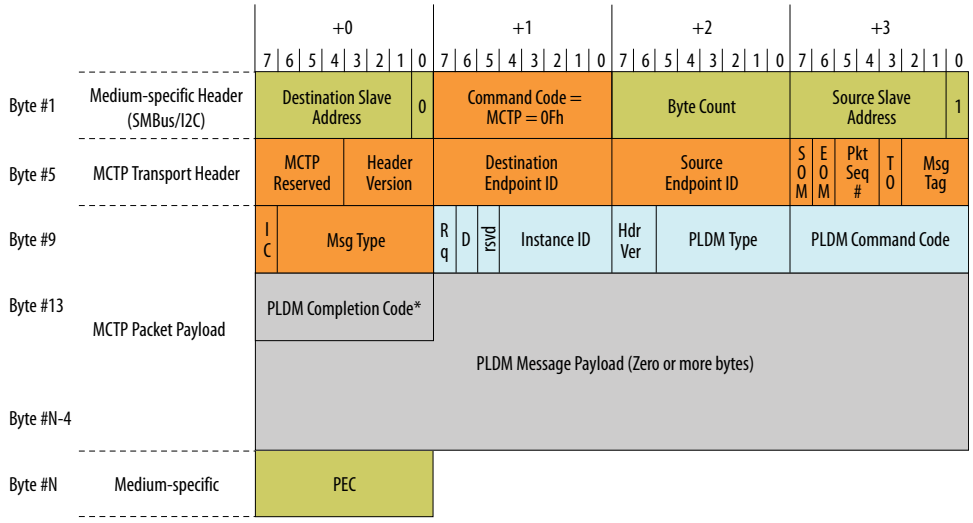
- The message body represents the payload of an MCTP message. The message body can span multiple MCTP packets.
- MCTP packet payload refers to the portion of the message body of an MCTP message that is carried in a single MCTP packet.
- Transmission Unit refers to the size of the portion of the MCTP packet payload.

Transmission Unit Size

- The baseline transmission unit (minimum transmission unit) size for MCTP is 64 bytes.
- All MCTP control messages are required to have a packet payload that is no larger than the baseline transmission unit without negotiation. (The negotiation mechanism for larger transmission units between endpoints is message type-specific and is not addressed in MCTP Base specification)
- Any MCTP message whose message body size is bigger than 64 bytes shall be split into multiple packets for a single message transmission.

MCTP Packet Fields

Figure 2. Generic Packet/Message Fields



Refer to the table below for more information on message fields.

Message Fields	Description
Byte Count	Byte counting from byte #4 to the byte right before PEC.
Header Version	Set to 0001b for MCTP devices that are conformant to the MCTP Base Specification DSP0237.
SOM, EOM, Pkt Seq #, TO, and Msg Tag	Refer to DSP0237 (MCTP SMB us/I2C Transport Binding Spec) and DSP0236 (MCTP Base Spec).

continued...

Message Fields	Description
IC	Message Integrity Check bit = 0b -> MCTP control and PLDM over MCTP messages do not include an overall Message Integrity check field according to DSP0236 and DSP0241 in respective.
Msg Type	0x00 for MCTP control and 0x01 for PLDM, others are not supported, refer to DSP0239 for more message types.
Rq, D, and Instance ID	Refer to the spec DSP0236.
Hdr Ver	0x00 according the spec DSP0240.
PLDM Type	0x00 for PLDM Messaging Control and Discovery, 0x02 for PLDM for Platform Monitoring and Control, only two types here are supported.
PLDM Command Code	Refer to the spec DSP0245 for more PLDM types.
PLDM Completion Code*	Present only in PLDM response message, refer to the spec DSP0240.
PLDM Message Payload	The fields within PLDM messages are transferred from lower offset (e.g. +0 byte) first. And Unless otherwise specified, byte ordering of multi-byte numeric fields or bit fields is "Big Endian" (that is, the lower byte offset holds the most significant byte, and higher offsets hold lesser significant bytes).

2.3. Supported Command Sets

Supported MCTP Commands

- `Get MCTP Version Support`
  - `Base Spec Version Info`
  - `Control Protocol Version Info`
  - `PLDM over MCTP Version`
- `Set Endpoint ID`
- `Get Endpoint ID`
- `Get Endpoint UUID`
- `Get Message Type Support`
- `Get Vendor Defined Message Support`

Note: For `Get Vendor Defined Message Support` command, the BMC responds with the completion code `ERROR_INVALID_DATA(0x02)`.

Supported PLDM Base Specification Commands

- `SetTID`
- `GetTID`
- `GetPLDMVersion`
- `GetPLDMTypes`
- `GetPLDMCommands`

Supported PLDM for Platform Monitoring and Control Specification Commands

- `SetTID`
- `GetTID`
- `GetSensorReading`
- `GetSensorThresholds`
- `SetSensorThresholds`
- `GetPDRRepositoryInfo`
- `GetPDR`

Note: The polling duration of the telemetry data by BMC NIOS is about 500~800 milliseconds (ms). Therefore, the duration between the start of the request message commands and getting the corresponding responses back is 500~800 ms. The interval between getting responses back from the previous command and the start of the next request message commands is 1 ms.

Note: `GetStateSensorReadings` is not supported.

2.4. PLDM Topology and Hierarchy

Defined Platform Descriptor Records

The Intel FPGA PAC N3000-N uses 20 Platform Descriptor Records (PDRs). Intel MAX 10 BMC only supports consolidated PDRs where the PDRs will not be added or removed dynamically when QSFP is plugged and unplugged. When unplugged the sensor operational status will simply be reported as unavailable.

Sensor Names and Record Handle

All PDRs are assigned an opaque numeric value called the Record Handle. This value is used for accessing individual PDRs within the PDR Repository via `GetPDR` (DTMF specification DSP0248).

The following table is a consolidated list of sensors monitored on Intel FPGA PAC N3000-N.

Table 1. PDRs Sensor Names and Record Handle

Please refer to *Intel FPGA Programmable Acceleration Card N3000-N ReadMe* for sensor output information.

Function	Sensor Name	Sensor Information	PLDM		
		Sensor Reading Source (Component)	PDR Record Handle	Thresholds in PDR	Threshold changes allowed via PLDM
Total Intel FPGA PAC input power	Board Power	Calculated using the formula: `(12 V Backplace Current * 12 V Backplane Voltage) + (12 V Aux Current * 12 V Aux Voltage)`	1	0	No
PCIe fingers 12 V Current	12 V Backplane Current	PAC1932 SENSE1	2	0	No
PCIe fingers 12 V Voltage	12 V Backplane Voltage	PAC1932 SENSE1	3	0	No
1.2 V Rail Voltage	1.2 V Voltage	MAX10 ADC	4	0	No
1.8 V Rail Voltage	1.8 V Voltage	MAX 10 ADC	6	0	No
3.3 V Rail Voltage	3.3 V Voltage	MAX 10 ADC	8	0	No
FPGA Core Voltage	FPGA Core Voltage	LTC3884	10	0	No
FPGA Core Current	FPGA Core Current	LTC3884	11	0	No
FPGA Core Temperature	FPGA Core Temperature	FPGA temp diode via TMP411	12	• Upper Warning: 90 • Upper Fatal: 100	Yes
Board Temperature	Board Temperature	TMP411	13	• Upper Warning: 85 • Upper Fatal: 100	Yes
QSFP A Voltage	QSFP A Voltage	External QSFP module	14	0	No

continued...

Function	Sensor Name	Sensor Information	PLDM		
		Sensor Reading Source (Component)	PDR Record Handle	Thresholds in PDR	Threshold changes allowed via PLDM
QSFP A Temperature	QSFP A Temperature	External QSFP module	15	• Upper Warning: Value set by QSFP Vendor • Upper Fatal: Value set by QSFP Vendor	No
PCIe Auxiliary 12V Current	12 V AUX Current	PAC1932 SENSE2	24	0	No
PCIe Auxiliary 12V Voltage	12 V AUX Voltage	PAC1932 SENSE2	25	0	No
QSFP B Voltage	QSFP B Voltage	External QSFP module	37	0	No
QSFP B Temperature	QSFP B Temperature	External QSFP module	38	• Upper Warning: Value set by QSFP Vendor • Upper Fatal: Value set by QSFP Vendor	No
Intel C827 Retimer A Core Temperature	Retimer A Core Temperature	Intel C827 Retimer chip (88EC055) (U18A)	44	0	No
Intel C827 Retimer A Serdes Temperature	Retimer A Serdes Temperature	Intel C827 Retimer chip (88EC055) (U18A)	45	0	No
Intel C827 Retimer B Core Temperature	Retimer B Core Temperature	Intel C827 Retimer chip (88EC055) (U23A)	46	0	No
Intel C827 Retimer B Serdes Temperature	Retimer B Serdes Temperature	Intel C827 Retimer chip (88EC055) (U23A)	47	0	No

Note: The Upper Warning and Upper Fatal values for QSFP are set by the QSFP vendor. Refer to vendor datasheet for the values. The BMC reads these threshold values and reports them out.

Note: If a sensor status shows as unavailable, please check if the card is still operational and not shut down due to over-heating. The LEDS blink yellow in case of card shut down. In such scenario restart the server and provide sufficient cooling to the card to keep it operational and read the sensor data. Refer to the *Cooling Requirements* section in the *Intel FPGA Programmable Acceleration Card N3000-N Data Sheet*.

Please refer to *Appendix A: Example PLDM Command and Response Messages* for an example PLDM command to read the board temperature.

fpgad is a service that can help you protect the server from crashing when the hardware reaches an upper non-recoverable or lower non-recoverable sensor threshold (also called as fatal threshold).

fpgad is capable of monitoring each of the 20 sensors reported by the Board Management Controller.

Please refer to the *Graceful Shutdown* topic from *Intel Acceleration Stack User Guide: Intel FPGA Programmable Acceleration Card N3000-N* for more information.

You can obtain the values of the sensors by running the following OPAE command as `root` or `sudo`:

```
$ sudo fpgainfo bmc
```

Related Information

Intel Acceleration Stack User Guide: Intel FPGA Programmable Acceleration Card N3000-N

3. Board Monitoring through I²C SMBus

The standard I²C slave to Avalon-MM interface (read-only) shares the PCIe SMBus between the host BMC and the Intel MAX 10 RoT. The Intel FPGA PAC N3000-N supports standard I²C slave interface and the slave address is 0xBC by default only for out-of-band access. Byte addressing mode is 2-byte offset address mode.

Reference the telemetry data register memory map below when accessing telemetry data through I²C. The description column includes the equations you can use to calculate actual temperature and hysteresis values. The units can be Celsius (°C), mA, mV, mW depending on which sensor you read.

Table 2. Telemetry Data Register Memory Map

Register	Offset	Width	Access	Field	Default Value	Description
Board Temperature	0x100	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer Temperature = register value * 0.5
Board Temperature High Warning	0x104	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer High Limit = register value * 0.5
Board Temperature High Fatal	0x108	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer High Fatal = register value * 0.5
Hysteresis	0x10C	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer Hysteresis = register value * 0.5
FPGA Core Temperature	0x110	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer Temperature = register value * 0.5
FPGA Core Temperature High Warning	0x114	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer High Limit = register value * 0.5
FPGA Core Temperature High Fatal	0x118	32	RO	[31:0]	32'h00000000	TMP411 Register value is signed integer

continued...

Register	Offset	Width	Access	Field	Default Value	Description
						High Fatal = register value * 0.5
QSFP A Temperature	0x11C	32	RO	[31:0]	32'h00000000	QSFP A Register value is signed integer Temperature = register value * 0.5
QSFP A Temperature High Fatal	0x120	32	RO	[31:0]	32'h00000000	QSFP A Register value is signed integer High Alarm = register value * 0.5
QSFP A Temperature High Warning	0x124	32	RO	[31:0]	32'h00000000	QSFP A Register value is signed integer High Warning = register value * 0.5
QSFP A Voltage	0X128	32	RO	[31:0]	32'h00000000	QSFP A Voltage(mv) = register value
QSFP B Temperature	0x12C	32	RO	[31:0]	32'h00000000	QSFP B Register value is signed integer Temperature = register value * 0.5
QSFP B Temperature High Fatal	0x130	32	RO	[31:0]	32'h00000000	QSFP B Register value is signed integer High Alarm = register value * 0.5
QSFP B Temperature High Warning	0x134	32	RO	[31:0]	32'h00000000	QSFP B Register value is signed integer High Warning = register value * 0.5
QSFP B Voltage	0x138	32	RO	[31:0]	32'h00000000	QSFP A Voltage(mv) = register value
FPGA Core Voltage	0x13C	32	RO	[31:0]	32'h00000000	LTC3884 Voltage(mV) = register value
FPGA Core Current	0x140	32	RO	[31:0]	32'h00000000	LTC3884 Current(mA) = register value
12v Backplane Voltage	0x144	32	RO	[31:0]	32'h00000000	Voltage(mV) = register value
12v Backplane Current	0x148	32	RO	[31:0]	32'h00000000	Current(mA) = register value
1.2v Voltage	0x14C	32	RO	[31:0]	32'h00000000	Voltage(mV) = register value

continued...

Register	Offset	Width	Access	Field	Default Value	Description
12v Aux Voltage	0x150	32	RO	[31:0]	32'h00000000	Voltage(mV) = register value
12v Aux Current	0x154	32	RO	[31:0]	32'h00000000	Current(mA) = register value
1.8v Voltage	0x158	32	RO	[31:0]	32'h00000000	Voltage(mV) = register value
3.3v Voltage	0x15C	32	RO	[31:0]	32'h00000000	Voltage(mV) = register value
Board Power	0x160	32	RO	[31:0]	32'h00000000	Power(mW) = register value
Retimer Link Status	0x164	32	RO	[0]	1'b0	Retimer A Port 0 link status
				[1]	1'b0	Retimer A Port 1 link status
				[2]	1'b0	Retimer A Port 2 link status
				[3]	1'b0	Retimer A Port 3 link status
				[4]	1'b0	Retimer B Port 0 link status
				[5]	1'b0	Retimer B Port 1 link status
				[6]	1'b0	Retimer B Port 2 link status
				[7]	1'b0	Retimer B Port 3 link status
				[31:8]	24'h00000000	Reserved
Retimer A Core Temperature	0x168	32	RO	[31:0]	32'h00000000	Retimer A Register value is signed integer Temperature = register value * 0.5
Retimer A Serdes Temperature	0x16C	32	RO	[31:0]	32'h00000000	Retimer A Register value is signed integer Temperature = register value * 0.5
Retimer B Core Temperature	0x170	32	RO	[31:0]	32'h00000000	Retimer B Register value is signed integer Temperature = register value * 0.5
Retimer B Serdes Temperature	0x174	32	RO	[31:0]	32'h00000000	Retimer B Register value is signed integer Temperature = register value * 0.5

The Intel MAX 10 BMC populates the QSFP voltage, temperature, high fatal temperature, and high warning temperature values in its telemetry data register by polling the QSFP module and copying the read values into the corresponding telemetry

data register. The BMC supports QSFP modules in compliance with SFF8436 or SFF8636 standard and SFP modules in compliance with SFF8472 standard. If the QSFP module does not support Digital Diagnostics Monitoring or if the QSFP module is not installed, you may see the following values for QSFP voltage, temperature, temperature high fatal, temperature high warning:

- `deadbeef` in the telemetry data registers
- `N/A` while running the fpgainfo bmc command
- `Device or resource busy` in the sysfs entries

The sysfs node entries for QSFP sensor data can be found at:

- For QSFPA:

```
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor14/value
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor15/value
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor15/high_warn
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor15/high_fatal
```

- For QSFPB:

```
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor37/value
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor38/value
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor38/high_warn
/sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0/spi-altera.0.auto/
spi_master/spi0/spi0.0/sensor38/high_fatal
```

Use the Intelligent Platform Management Interface (IPMI) tool to read the telemetry data through the I²C bus.

Example 1. I²C command to read the board temperatures at address 0x100:

In the command below:

- **0x20** is the I²C master bus address of your server that can access PCIe slots directly. This address varies with the server. Please refer to your server datasheet for the correct I²C address of your server.
- **0xBC** is the I²C slave address of the Intel MAX 10 BMC.
- **4** is the number of read data bytes
- **0x01 0x00** is the register address of the board temperature which is presented in the Table 2 on page 14.

Command:

```
$ sudo ipmitool i2c bus=0x20 0xBC 4 0x01 0x00
```

Output:

```
01110010 00000000 00000000 00000000
```

The output value returned is in big endian format i.e., the lower bytes are transferred first before the upper bytes. So the hexidecimal returned is: 0x00000072

0x72 is 114 in decimal.

To calculate the temperature in Celsius multiply by 0.5: 114 x 0.5 = 57 °C

Note: Please check with your server vendor if you encounter the following error when running the ipmitool command:

```
I2C Master Write-Read command failed: Bus Error
Unable to perform I2C Master Write-Read
```

The bus address and command could be different depending on the server vendor. Here are some IPMI tool commands for different servers to read board temperature:

- Dell PowerEdge R740 server:

  ```
  $ sudo ipmitool i2c bus=0 0xBC 4 0x01 0x00
  ```

- Intel Neoncity server:

  ```
  ipmitool i2c bus=2 0xBC 4 0x01 0x00
  ```

4. EEPROM Data Format

This section defines the data format of both the MAC Address EEPROM and the FRUID EEPROM and that can be accessed by the host and FPGA respectively.

4.1. MAC EEPROM

At the time of manufacturing, Intel programs the MAC address EEPROM with the Intel Ethernet Controller XL710-BM2 MAC addresses. The Intel MAX 10 accesses the addresses in the MAC address EEPROM through the I^2C bus.

Discover the MAC address using the following command:

```
$ sudo fpgainfo mac
```

The MAC Address EEPROM only contains the starting 6-byte MAC address at address `0x00h` followed by the MAC address count of 08. The starting MAC address is also printed on the label sticker on the back side of the Printed Circuit Board (PCB).

The OPAE driver provides sysfs nodes to obtain the starting MAC address from the following location: `/sys/class/fpga/intel-fpga-dev.*/intel-fpga-fme.*/spi-altera.*.auto/spi_master/spi*/spi*/mac_address`

Starting MAC Address Example:

```
644C360F4430
```

The OPAE driver obtains the count from the following location: `/sys/class/fpga/intel-fpga-dev.*/intel-fpga-fme.*/spi-altera.*.auto/spi_master/spi*/spi*/mac_count`

MAC count Example:

```
08
```

From the starting MAC address, the remaining seven MAC addresses are obtained by sequentially incrementing the Least Significant Byte (LSB) of the starting MAC Address by a count of one for each subsequent MAC address.

Subsequent MAC address example:

```
644C360F4431
644C360F4432
644C360F4433
644C360F4434
644C360F4435
644C360F4436
644C360F4437
```

4.2. Field Replaceable Unit Identification (FRUID) EEPROM Access

You can only read the field replaceable unit identification (FRUID) EEPROM (0xA0) from the host BMC through SMBus directly.

The structure in the FRUID EEPROM is based on the IPMI specification, *Platform Management FRU Information Storage Definition, v1.3, March 24, 2015*, from which a board information structure is derived. The FRUID EEPROM follows the common header format with Board Area and Product Info Area. Refer to the table below for what fields in the common header apply to the FRUID EEPROM.

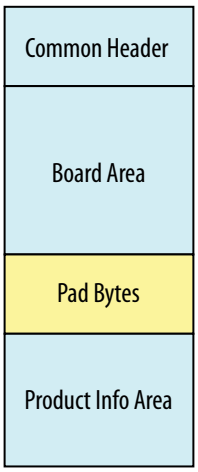
Table 3. Common Header of FRUID EEPROM

All the fields in the common header are mandatory.

Field Length in Bytes	Field Description	FRUID EEPROM Value
1	Common Header Format Version 7:4 - reserved, write as 0000b 3:0 - format version number = 1h for this specification	01h (Set as 00000001b)
1	Internal Use Area Starting Offset (in multiples of 8 bytes). 00h indicates that this area is not present.	00h (not present)
1	Chassis Info Area Starting Offset (in multiples of 8 bytes). 00h indicates that this area is not present.	00h (not present)
1	Board Area Starting Offset (in multiples of 8 bytes). 00h indicates that this area is not present.	01h
1	Product Info Area Starting Offset (in multiples of 8 bytes). 00h indicates that this area is not present.	0Ch
1	MultiRecord Area Starting Offset (in multiples of 8 bytes). 00h indicates that this area is not present.	00h (not present)
1	PAD, write as 00h	00h
1	Common Header Checksum (zero checksum)	F2h

The common header bytes are placed from the first address of the EEPROM. The layout looks like the figure below.

Figure 3. FRUID EEPROM Memory Layout Block Diagram



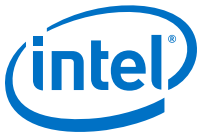


Table 4. FRUID EEPROM Board Area

Field Length in Bytes	Field Description	Field Values	Field Encoding
1	Board Area Format Version 7:4 - reserved, write as 0000b 3:0 - format version number	0x01	Set to 1h (0000 0001b)
1	Board Area Length (in multiples of 8 bytes)	0x0B	88 bytes (includes 2 pad 00 bytes)
1	Language Code	0x00	Set to 0 for English *Note:* No other languages supported at this time
3	Mfg. Date / Time: Number of minutes from 0:00 hrs 1/1/96. Least Significant byte first (little endian) 00_00_00h = unspecified (Dynamic field)	0x10 0x65 0xB7	Time difference between 12:00 AM 1/1/96 to 12 PM 11/07/2018 is 12018960 minutes = b76510h – stored in little endian format
1	Board Manufacturer type/length byte	0xD2	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 010010b (18 bytes of data)
P	Board Manufacturer bytes	0x49 0x6E 0x74 0x65 0x6C 0xAE 0x20 0x43 0x6F 0x72 0x70 0x6F 0x72 0x61 0x74 0x69 0x6F 0x6E	8-bit ASCII + LATIN1 coded
1	Board Product Name type/length byte	0xD5	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 010101b (21 bytes of data)
Q	Board Product Name bytes	0X49 0X6E 0X74 0X65 0X6C 0XAE 0X20 0X46 0X50 0X47 0X41 0X20 0X50 0X41	8-bit ASCII + LATIN1 coded Intel FPGA PAC N3000

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
		0X43 0X20 0X4E 0X33 0X30 0X30 0X30	
1	Board Serial Number type/length byte	0xCC	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 001100b (12 bytes of data)
N	Board Serial Number bytes (Dynamic field)	0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30	8-bit ASCII + LATIN1 coded 1st 6 hex digits are OUI: 000000 2nd 6 hex digits are MAC address: 000000 *Note:* This is coded as an example and needs to be modified in an actual device 1st 6 hex digits are OUI: 644C36 2nd 6 hex digits are MAC address: 00AB2E *Note:* To identify not programmed FRUID, set OUI and MAC address to "0000".
1	Board Part Number type/length byte	0xCE	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 001110b (14 bytes of data)
M	Board Part Number bytes	0x4B 0x38 0x37 0x35 0x38 0x34 0x2D 0x30 0x30 0x31 0x20 0x20 0x20 0x20	8-bit ASCII + LATIN1 coded with BOM ID. For 14 byte length, the coded board part number example is K87584-001 *Note:* This is coded as an example and needs to be modified in an actual device. This field value varies with different board PBA number. PBA Revision has been removed in FRUID. These last four bytes return blank and are reserved for future use.
1	FRU File ID type/length byte	0x00	8-bit ASCII + LATIN1 coded 7:6 – 00b 5:0 – 000000b (0 bytes of data) The FRU File ID bytes field that should follow this is not included as the field would be 'null'.

continued...

(1) These fields are always coded as if the language code is English.

Field Length in Bytes	Field Description	Field Values	Field Encoding
			Note: FRU File ID bytes. The FRU File version field is a pre-defined field provided as a manufacturing aid for verifying the file that was used during manufacture or field update to load the FRU information. The content is manufacturer-specific. This field is also provided in the Board Info area. Either or both fields may be 'null'.
1	MMID type/length byte	0xC6	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 000110b (6 bytes of data) *Note:* This is coded as an example and needs to be modified in an actual device
M	MMID bytes	0x39 0x39 0x39 0x57 0x43 0x46	Formatted as 6 hex digits. Specific example in cell alongside Intel FPGA PAC N3000-N MMID = 999PJD. This field value varies with different SKUs fields like MMID, OPN, PBN etc.
1	C1h (type/length byte encoded to indicate no more info fields).	0xC1	
Y	00h - any remaining unused space	0x00	
1	Board Area Checksum (zero checksum)	0x35	*Note:* The checksum in this table is a zero checksum computed for the values used in the table. It must be recomputed for the actual values of a Intel FPGA PAC N3000-N.

Table 5. FRUID EEPROM Product Info Area

Field Length in Bytes	Field Description	Field Values	Field Encoding
1	Product Area Format Version 7:4 - reserved, write as 0000b 3:0 - format version number = 1h for this specification	0x01	Set to 1h (0000 0001b)
1	Product Area Length (in multiples of 8 bytes)	0x0A	Total of 80 bytes
1	Language Code	0x00	Set to 0 for English *Note:* No other languages supported at this time
1	Manufacturer Name type/length byte	0xD2	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 010010b (18 bytes of data)
N	Manufacturer Name bytes	0x49 0x6E 0x74 0x65 0x6C	8-bit ASCII + LATIN1 coded Intel Corporation

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
		0xAE 0x20 0x43 0x6F 0x72 0x70 0x6F 0x72 0x61 0x74 0x69 0x6F 0x6E	
1	Product Name type/length byte	0xD5	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 010101b (21 bytes of data)
M	Product Name bytes	0x49 0x6E 0x74 0x65 0x6C 0xAE 0x20 0x46 0x50 0x47 0x41 0x20 0x50 0x41 0x43 0x20 0x4E 0x33 0x30 0x30 0x30	8-bit ASCII + LATIN1 coded Intel FPGA PAC N3000-N
1	Product Part/Model Number type/length byte	0xCE	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 001110b (14 bytes of data)
O	Product Part/Model Number bytes	0x42 0x44 0x2D 0x4E 0x56 0x56 0x2D 0x4E 0x33 0x30 0x30 0x30	8-bit ASCII + LATIN1 coded OPN for the board BD-NVV-N3000-3 This field value varies with different Intel FPGA PAC N3000-N OPNs.

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
		0x2D 0x56	
1	Product Version type/length byte	0x01	8-bit binary 7:6 – 00b 5:0 – 000001b (1 byte of data)
R	Product Version bytes	0x00	This field is encoded as family member
1	Product Serial Number type/length byte	0xCC	8-bit ASCII + LATIN1 coded 7:6 – 11b 5:0 – 001100b (12 bytes of data)
P	Product Serial Number bytes (Dynamic field)	0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30 0x30	8-bit ASCII + LATIN1 coded 1st 6 hex digits are OUI: 000000 2nd 6 hex digits are MAC address: 000000 *Note:* This is coded as an example and needs to be modified in an actual device. 1st 6 hex digits are OUI: 644C36 2nd 6 hex digits are MAC address: 00AB2E *Note:* To identify not programmed FRUID, set OUI and MAC address to "0000".
1	Asset Tag type/length byte	0x01	8-bit binary 7:6 – 00b 5:0 – 000001b (1 byte of data)
Q	Asset Tag	0x00	Not supported
1	FRU File ID type/length byte	0x00	8-bit ASCII + LATIN1 coded 7:6 – 00b 5:0 – 000000b (0 bytes of data) The FRU File ID bytes field that should follow this is not included as the field would be 'null'. *Note:* FRU file ID bytes. The FRU File version field is a pre-defined field provided as a manufacturing aid for verifying the file that was used during manufacture or field update to load the FRU information. The content is manufacturer-specific. This field is also provided in the Board Info area. Either or both fields may be 'null'.
1	C1h (type/length byte encoded to indicate no more info fields).	0xC1	
Y	00h - any remaining unused space	0x00	
1	Product Info Area Checksum (zero checksum) (Dynamic Field)	0xDC	*Note:* the checksum in this table is a zero checksum computed for the values used in the table. It must be recomputed for the actual values of a Intel FPGA PAC.

5. Document Revision History for Board Management Controller User Guide: Intel FPGA Programmable Acceleration Card N3000-N

Document Version	Intel Acceleration Stack Version	Changes
2020.09.08	1.3.1	Updated in accordance with the Intel Acceleration Stack 1.3.1 Version for Intel FPGA Programmable Acceleration Card N3000-N.
2020.06.15	1.3	Initial release.

A. Example PLDM Command and Response Messages

This section provides an example command to read the board temperatures through PLDM over MCTP SMBus.

Use the following commands to read the board temperatures:

- `GetPDR`
- `GetSensorReading`

`GetPDR`

Run the `GetPDR` command with the board temperature PDR record handle (13) to get the board sensor ID, and temperature conversion formula. You can also observe threshold values set for the board temperature sensor.

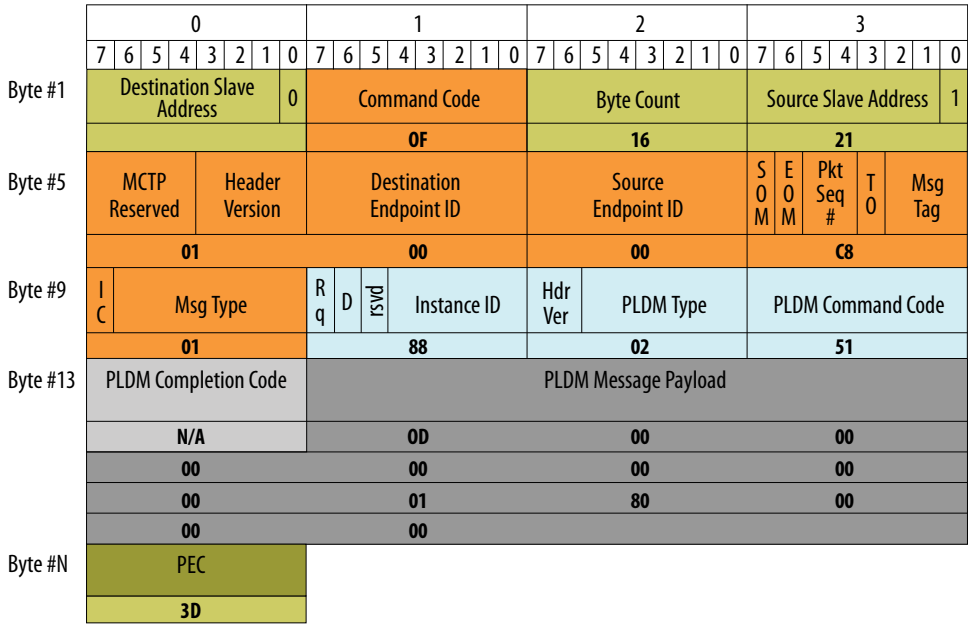
`GetPDR` Request Command (0x51):

The length of the write data is 25.

```
0x0F 0x16 0x21
0x01 0x00 0x00 0xC8
0x01 0x88 0x02 0x51 0x0D 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x80 0x00 0x00
0x00
0x3D
```

Below is the representation of the `GetPDR` command arranged in their respective SMBUS, MCTP, PLDM header location.

Figure 4. GetPDR Command Packet Fields



To decode the PLDM message payload, refer to DSP0248_1.2.0 spec.

GetPDR Request Data	Value
recordHandle	0x0000_000D – PDR handle for board temperature is 13
dataTransferHandle	0x0000_0000
transferOperationFlag	0x01
requestCount	0x0080 (128 bytes)
recordChangeNumber	0x0000

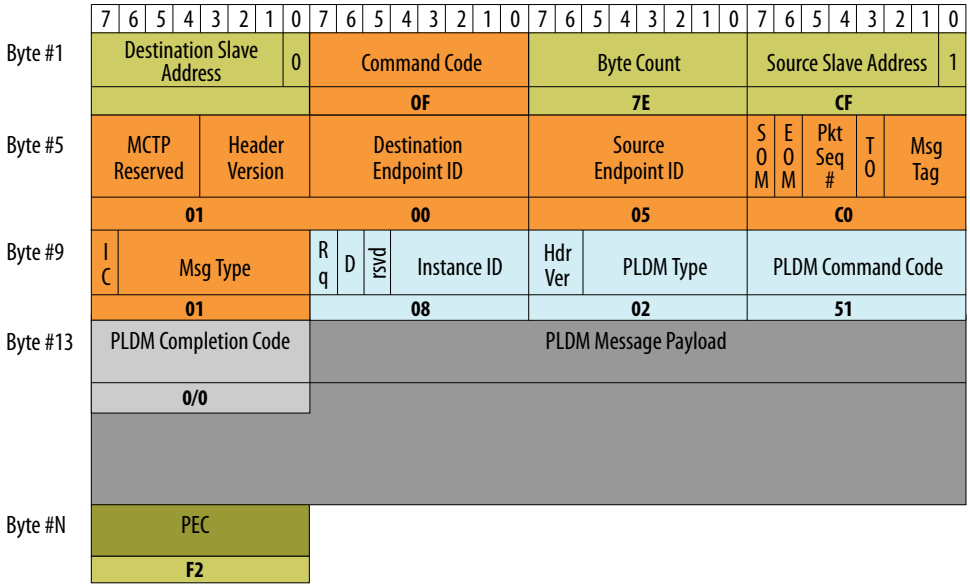
GetPDR Response:

The length of the received data is 131 bytes.

```
0x20 0x0F 0x7E 0xCF 0x01 0x00 0x05 0xC0 0x01 0x08 0x02 0x51 0x00 0x02 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x05 0x69 0x00 0x01 0x00 0x00 0x00 0x01 0x02 0x00 0x00
0x5F 0x00 0x34 0x12 0x01 0x00 0x3E 0x00 0x01 0x00 0x23 0x01 0x00 0x01 0x02 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x05 0x00 0x00 0x00 0x3F 0x00 0x00 0x00
0x00 0x00 0x00 0x02 0x02 0x14 0x00 0x00 0x00 0x05 0x06 0x00 0x00 0x40 0x40 0x00
0x00 0x40 0x40 0xFE 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x05 0x20 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x55 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x64 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0xF2
```

Below is the presentation of the of the above command arranged in their respective SMBUS, MCTP, PLDM header location

Figure 5. GetPDR Response Message Packet Fields



PLDM response message payload for `GetPDR`:

```
0x00 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x05 0x69 0x00
0x01 0x00 0x00 0x00 0x01 0x02 0x00 0x00 0x5F 0x00
0x34 0x12 0x01 0x00 0x3E 0x00 0x01 0x00 0x23 0x01 0x00 0x01 0x02 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x01 0x05 0x00 0x00 0x00 0x00 0x3F 0x00 0x00 0x00 0x00
0x00 0x02 0x02 0x14 0x00 0x00 0x00 0x05 0x06 0x00 0x00 0x40 0x40 0x00 0x00 0x40
0x40 0xFE 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x05 0x20 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x55 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x64 0x00 0x00 0x00 0x00 0x00 0x00 0xF2
```

To decode the PLDM message payload, refer to DSP0248_1.2.0 specification:

GetPDRResponse Data	Value
Completion code	0x00
nextRecordHandle	0x0000_0002
nextDataTransferHandle	0x0000_0000
transferFlag	0x05
responseCount	0x0069 = 105 bytes
recordData	0x00

Common PDR Header Format	Value
recordHandle	0x0000_0001
PDRHeaderVersion	0x01
PDRType	0x02
recordChangeNumber	0x0000
dataLength	0x005F = 95 bytes

Numeric Sensor PDR Format	Value
PLDMTerminusHandle	0x1234
sensorID	0x0001
entityType	0x003E
entityInstanceNumber	0x0001
containerID	0x0123
sensorInit	0x00- noInit
sensorAuxiliaryNamesPDR	0x01 - False
baseUnit	0x02 - units -Degrees C
unitModifier	0x00
rateUnit	0x00
baseOEMUnitHandle	0x00
auxUnit	0x00
auxUnitModifier	0x00
auxrateUnit	0x00
rel	0x00
auxOEMUnitHandle	0x00
isLinear	0x01
sensorDataSize	0x05 -Sint32
resolution	0x3F00_0000 (IEEE754 conversion is 0.5)
offset	0x00 (IEEE 754 conversion is 0.0)
accuracy	0x00
plusTolerance	0x02
minusTolerance	0x02
hysteresis	0x0000_0014
supportedThresholds	0x05
thresholdAndHysteresisVolatility	0x06
stateTransitionInterval	0x4040_0000 (IEEE 754 conversion is 3.0)
updateInterval	0x4040_0000 (IEEE 754 conversion is 3.0)
maxReadable	0x0000_00FE
minReadable	0x0000_0000
rangeFieldFormat	0x05 - Sint32
rangeFieldSupport	0x0010_0000
nominalValue	0x0000_0000
normalMax	0x0000_0000
normalMin	0x0000_0000
warningHigh	0x0000_0055 = 85 units

continued...

Numeric Sensor PDR Format	Value
warningLow	0x0000_0000
criticalHigh	0x0000_0000
criticalLow	0x0000_0000
fatalHigh	0x0000_0064 = 100 units
fatalLow	0x0000_0000
PEC	0xF2

You can observe the following values from the above table:

- The sensor ID for board temperature is 0x0001
- Upper Warning high value is 85 °C
- Upper Fatal high value is 100 °C
- The reading conversation formula is Y= (m*X+B) where,
  - Y= Converted reading in units
  - X= Reading from sensor obtained by running GetSensorReading command
  - m= Resolution from PDR in units (0.5)
  - B = Offset from PDR in units (0.0)

`GetSensorReading`

Run the `GetSensorReading` command to get the raw sensor reading of the board temperature.

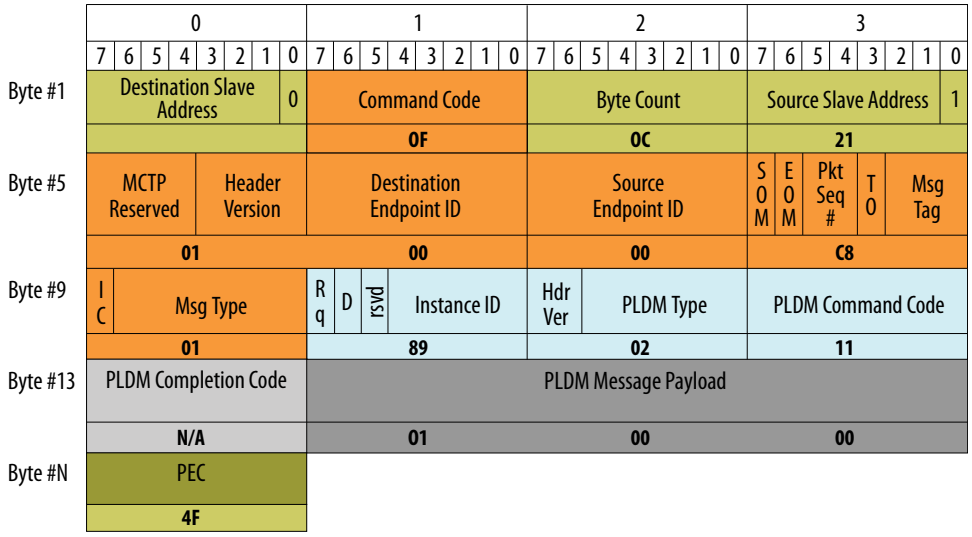
`GetSensorReading` Command (0x11):

the length of the write data is 15.

```
0x0F 0x0C 0x21 0x01 0x00 0x00 0xC8 0x01 0x89 0x02 0x11 0x01 0x00 0x00 0x4F
```

Below is the of the above command arranged in their respective SMBUS, MCTP, PLDM header location.

Figure 6. GetSensorReading Command Packet Fields



To decode the PLDM message payload, refer to DSP0248_1.2.0 specification.

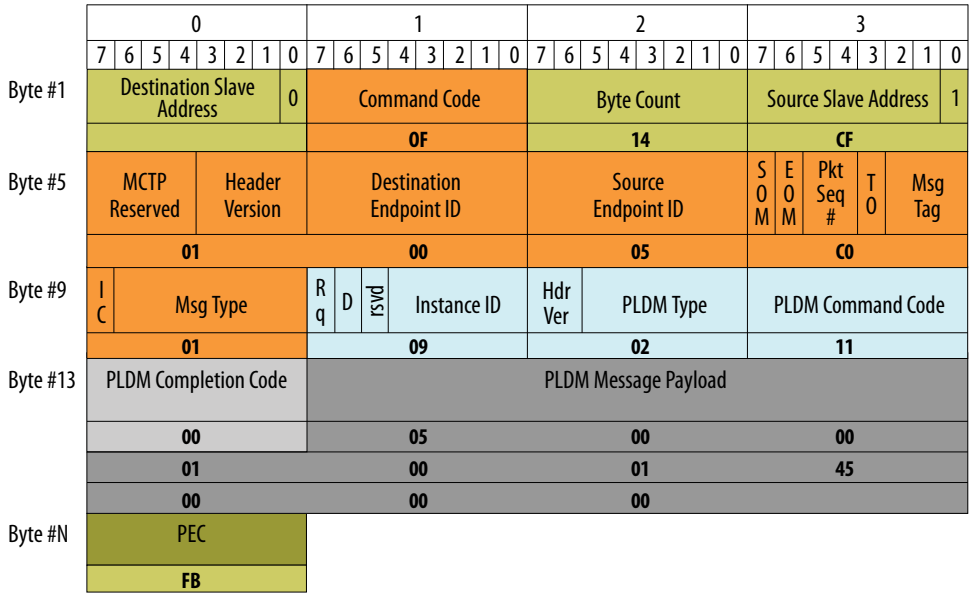
GetSensorReading Requestdata	Value
sensorID	0x0001
rearmEventState	0x00

`GetSensorReading` Response:

The length of the received data is 24.

```
0x20 0x0F 0x14 0xCF 0x01 0x00 0x05 0xC0 0x01 0x09 0x02 0x11 0x00 0x05 0x00 0x00
0x01 0x00 0x01 0x45 0x00 0x00 0x00 0xFB
```

Figure 7. GetSensorReading Response Message Packet Fields



To decode the PLDM message Payload, refer to DSP0248_1.2.0 specification.

GetSensorReading Response	Value
CompletionCode	0x00
SensorDataSize	0x05 - sint32
SensorOperationalState	0x00 -enabled
SensorEventMessageEnable	0x00
presentState	0x01 - Normal
previousState	0x00 - unknown
eventState	0x01- Normal
presentReading	0x0000_0045 = 69 units

You can observe the following values from the above table:

- The raw data value of board temperature is 69 °C
- Find the actual temperature using the conversion formula, Y=(m*X+B)
- Actual Board temperature value = (69*0.5+0.0)= 34.5 °C