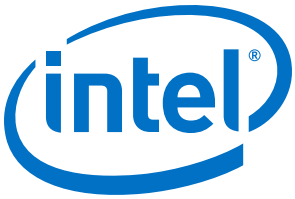


Board Management Controller User Guide

Intel FPGA Programmable Acceleration Card N3000

Subscribe
Send Feedback

UG-20222 | 2019.11.25
Latest document on the web: PDF | HTML

Contents

1. Introduction

1.1. About this Document

Reference the Intel FPGA Programmable Acceleration Card N3000 Board Management User Guide to learn more about the functions and features of the Intel® MAX® 10 BMC and to understand how to read telemetry data on the Intel FPGA PAC N3000 using PLDM over MCTP SMBus and I^2C SMBus. An introduction to Intel MAX 10 root of trust (RoT) and secure remote system update is included.

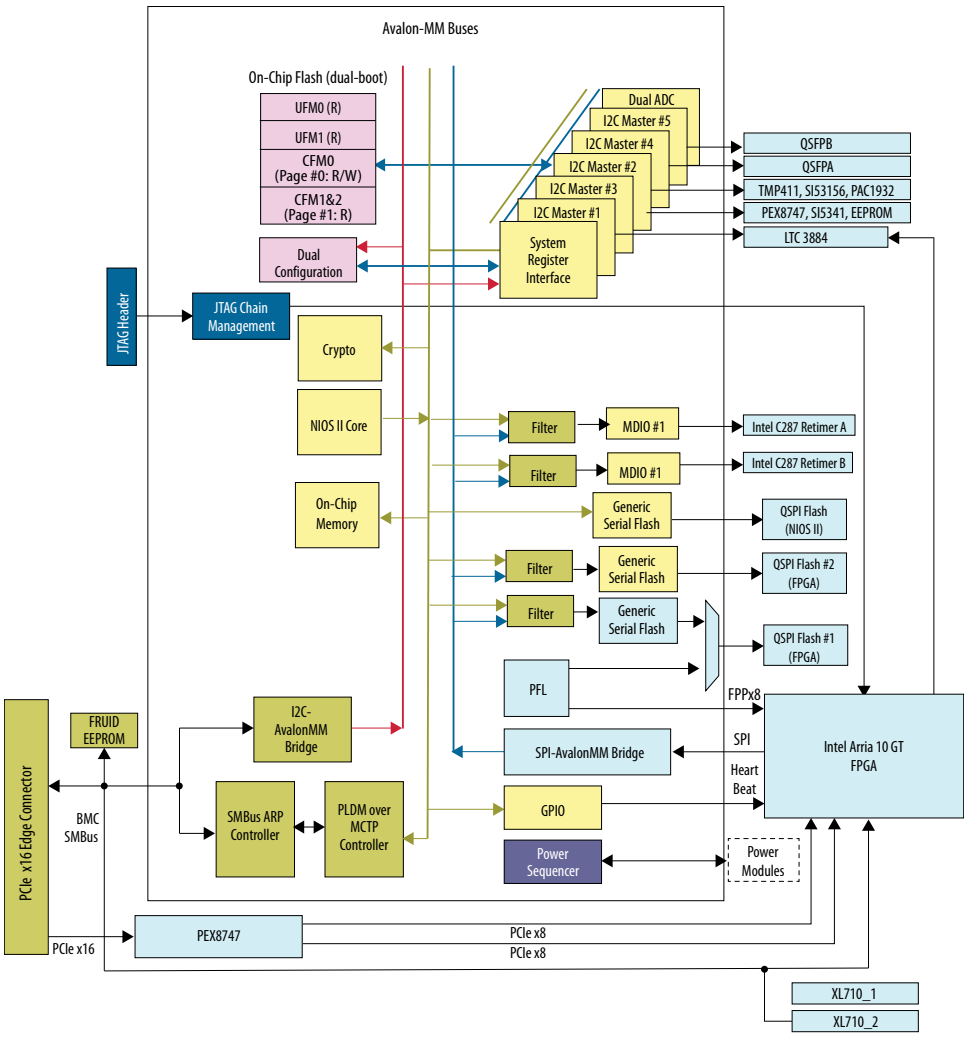
1.2. Overview

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 Platform Level Data Model (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.
- Controls firmware and FPGA flash updates over PCIe.
- Manages FPGA configuration.
- Configures the network settings for the C827 Ethernet re-timer device.
- Controls Power up and power down 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 outside of critical threshold.
  - Reports telemetry data to host BMC via Platform Level Data Model (PLDM) over MCTP SMBus or I^2C.
  - Supports PLDM over MCTP SMBus via PCIe SMBus. 0xCE is a 8-bit slave address.
  - Supports I^2C SMBus. 0xBC is the 8-bit slave address.
- Accesses the Ethernet MAC addresses in EEPROM and field replaceable unit identificiation (FRUID) EEPROM.

Figure 1. BMC High-Level 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. 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 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 *Intel FPGA Programmable Acceleration Card N3000 Security User Guide* for information on RoT and security features of the Intel FPGA PAC N3000.

Related Information

Intel FPGA Programmable Acceleration Card N3000 Security User Guide

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 user signed 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 *Intel FPGA Programmable Acceleration Card N3000 Security User Guide* for information on RoT and security features of the Intel FPGA PAC N3000.

Related Information

Intel FPGA Programmable Acceleration Card N3000 Security User Guide

1.5. Power Sequence Management

The BMC Power sequencer state machine manages Intel FPGA PAC N3000 power-on and power-off sequences for corner cases during the power-on process or normal operation. 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 runs into corner cases such as a Intel MAX 10 or FPGA factory build load failure or Nios boot up failure. The BMC protects the Intel FPGA PAC N3000 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 85 °C

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. Host BMC can access the telemetry data through PCIe SMBus. The PCIe SMBus between host BMC and Intel FPGA PAC N3000 Intel MAX 10 BMC is shared by both the PLDM over MCTP 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 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 slave address is 0xCE by default. It can be reprogrammed into corresponding section of external FPGA Quad SPI flash via in-band way if necessary.

The Intel FPGA PAC N3000 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: Platform Level Data Model (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,but all four device categories are supported and are field-reconfigurable.

ACK-Poll is supported

- Supported with SMBus default slave address 0xCE.
- Supported with a fixed or assigned slave address.

The BMC supports version 1.3.0 of the Management Component Transport Protocol (MCTP) Base Specification (DTMF specification DSP0236), version 1.1.1 of the PLDM for Platform Monitoring and Control standard (DTMF specification DSP0248), and version 1.0.0 of the PLDM for Message Control and Discovery (DTMF specification DSP0240).

Related Information

Distributed Management Task Force (DMTF) Specifications
For link to specific DMTF specifications

2.1. SMBus Interface Speed

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

2.2. MCTP Packetization Support

MCTP Definitions

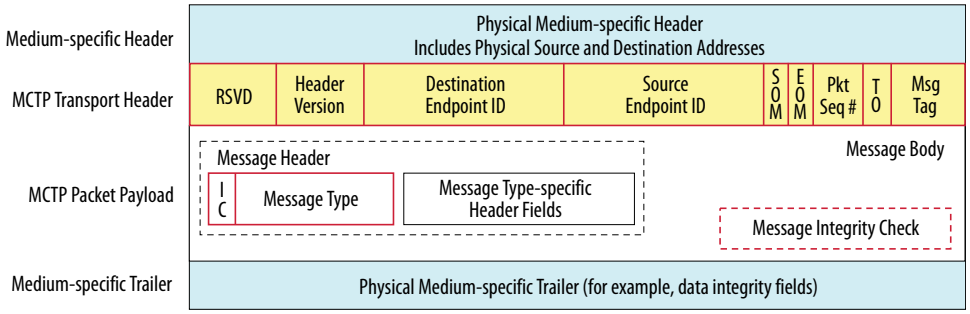
- The message body represents the payload of an MCTP message. The message body can span multiple MCTP packets.
- MCTP packet payload refers to the portion of the message body of an MCTP message that is carried in a single MCTP packet.
- Transmission Unit refers to the size of the portion of the MCTP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



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 BMC Nios II core polls for different telemetry data every 1 millisecond, and the polling duration takes about 500~800 milliseconds, hence the response message versus a corresponding request message of the command `GetSensorReading` or `GetSensorThresholds` accordingly updates every 500~800 milliseconds.

Note: `GetStateSensorReadings` is not supported.

2.4. PLDM Topology and Hierarchy

Defined Platform Descriptor Records

The Intel FPGA PAC N3000 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.

Table 1. PDRs Sensor Names and Record Handle

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	Calculate from PCIe fingers 12V Current and 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 (U44)	10	0	No
FPGA Core Current	FPGA Core Current	LTC3884 (U44)	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 (U65)	13	Upper Warning: 75 Upper Fatal: 85	Yes
QSFP0 Voltage	QSFP0 Voltage	External QSFP module (J4)	14	0	No
QSFP0 Temperature	QSFP0 Temperature	External QSFP module (J4)	15	Upper Warning: Value set by QSFP Vendor Upper Fatal: Value set by QSFP Vendor	No
PCIe Auxiliary 12V Current	12 V AUX	PAC1932 SENSE2	24	0	No
PCIe Auxiliary 12V Voltage	12 V AUX Voltage	PAC1932 SENSE2	25	0	No
QSFP1 Voltage	QSFP1 Voltage	External QSFP module (J5)	37	0	No
QSFP1 Temperature	QSFP1 Temperature	External QSFP module (J5)	38	Upper Warning: Value set by QSFP Vendor Upper Fatal: Value set by QSFP Vendor	No
PKVL A Core Temperature	PKVL A Core Temperature	PKVL chip (88EC055) (U18A)	44	0	No

continued...

Function	Sensor Name	Sensor Information	PLDM		
		Sensor Reading Source (Component)	PDR Record Handle	Thresholds in PDR	Threshold changes allowed via PLDM
PKVL A Serdes Temperature	PKVL A Serdes Temperature	PKVL chip (88EC055) (U18A)	45	0	No
PKVL B Core Temperature	PKVL B Core Temperature	PKVL chip (88EC055) (U23A)	46	0	No
PKVL B Serdes Temperature	PKVL B Serdes Temperature	PKVL 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 will read these threshold values and report them out.

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* for more information.

Note: Qualified OEM server systems should provide the required cooling for your workloads.

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

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 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.

Here is the telemetry data register memory map that you can use to access information through the I²C commands. The description column describes how the returned register values may be further processed to get the actual values. The units can be Celsius (°C), mA, mV, mW depending on what 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(U65) Register value is signed integer Temperature = register value * 0.5
Board Temperature High Warn	0x104	32	RW	[31:0]	32'h00000000	TMP411(U65) Register value is signed integer High Limit = register value * 0.5
Board Temperature High Fatal	0x108	32	RW	[31:0]	32'h00000000	TMP411(U65) Register value is signed integer High Critical = register value * 0.5
FPGA Core Temperature	0x110	32	RO	[31:0]	32'h00000000	TMP411(U65) Register value is signed integer Temperature = register value * 0.5
FPGA Die Temperature High Warn	0x114	32	RW	[31:0]	32'h00000000	TMP411(U65) Register value is signed integer High Limit = register value * 0.5

continued...

Register	Offset	Width	Access	Field	Default Value	Description
FPGA Die Temperature High Fatal	0x118	32	RW	[31:0]	32'h00000000	TMP411(U65) Register value is signed integer High Critical = register value * 0.5
QSFP0 Temperature	0x11C	32	RO	[31:0]	32'h00000000	QSFPA(J4) Register value is signed integer Temperature = register value * 0.5
QSFP0 Temperature High Fatal	0x120	32	RW	[31:0]	32'h00000000	QSFPA(J4) Register value is signed integer High Alarm = register value * 0.5
QSFP0 Temperature High Warn	0x124	32	RW	[31:0]	32'h00000000	QSFPA(J4) Register value is signed integer High Warning = register value * 0.5
QSFP0 Voltage	0x128	32	RO	[31:0]	32'h00000000	QSFPA(J4) Voltage(mv) = register value
QSFP1 Temperature	0x12C	32	RO	[31:0]	32'h00000000	QSFPB(J5) Register value is signed integer Temperature = register value * 0.5
QSFP1 Temperature High Fatal	0x130	32	RW	[31:0]	32'h00000000	QSFPB(J5) Register value is signed integer High Alarm = register value * 0.5
QSFP1 Temperature High Warn	0x134	32	RW	[31:0]	32'h00000000	QSFPB(J5) Register value is signed integer High Warning = register value * 0.5
QSFP1 Voltage	0x138	32	RO	[31:0]	32'h00000000	QSFPB(J5) Voltage(mV) = register value

continued...

Register	Offset	Width	Access	Field	Default Value	Description
FPGA Core Voltage	0x13C	32	RO	[31:0]	32'h00000000	LTC3884(U44) Voltage(mV) = register value
FPGA Core Current	0x140	32	RO	[31:0]	32'h00000000	LTC3884(U44) 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
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
PKVL A Core Temperature	0x168	32	RO	[31:0]	32'h00000000	PKVL1(U18A) Register value is signed integer Temperature = register value * 0.5
PKVL A Serdes Temperature	0x16C	32	RO	[31:0]	32'h00000000	PKVL1(U18A) Register value is signed integer Temperature = register value * 0.5
PKVL B Core Temperature	0x170	32	RO	[31:0]	32'h00000000	PKVL2(U23A) Register value is signed integer Temperature = register value * 0.5
PKVL B Serdes Temperature	0x174	32	RO	[31:0]	32'h00000000	PKVL2(U23A) Register value is signed integer Temperature = register value * 0.5

QSFP values are obtained by reading the QSFP module and reporting the read values in the appropriate register. If the QSFP module does not support Digital Diagnostics Monitoring or if the QSFP module is not installed, then ignore values read from QSFP registers.

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.

Command:

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

Output:

```
01110010 00000000 00000000 00000000
```

The output value in hexidecimal is: 0x72000000

0x72 is 114 in decimal.

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

Note: Not all servers support I²C bus directly access to PCIe slots. Please check your server datasheet for support information and I²C bus address.

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 fpga 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 obtain 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
```

Note: If you are using an ES Intel FPGA PAC N3000, the MAC EEPROM may not be programmed. If the MAC EEPROM is not programmed then the first MAC address read returns as FFFFFFFFFFFF.

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.

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

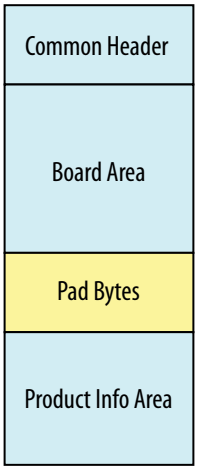


Table 4. FRUID EEPROM Board Area

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	8-bit ASCII + LATIN1 coded Intel® Corporation

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
		0x61 0x74 0x69 0x6F 0x6E	
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 0X43 0X20 0X4E 0X33 0X30 0X30 0X30	8-bit ASCII + LATIN1 coded Intel FPGA PAC N3000
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 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

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: 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 0x32 0x34 0x31 0x37 0x20 0x30 0x30 0x32 0x20 0x20 0x20 0x20	8-bit ASCII + LATIN1 coded with BOM ID. For 14 byte length, the coded board part number example is K82417-002 *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'. *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

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
M	MMID bytes	0x39 0x39 0x39 0x44 0x58 0x46	Formatted as 6 hex digits. Specific example in cell alongside Intel FPGA PAC N3000 MMID = 999DXF. 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)	0xB9	*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.

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 0xAE 0x20 0x43 0x6F 0x72 0x70 0x6F 0x72 0x61 0x74	8-bit ASCII + LATIN1 coded Intel Corporation

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
		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
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 0x2D 0x31	8-bit ASCII + LATIN1 coded OPN for the board BD-NVV-N3000-1 This field value varies with different Intel FPGA PAC N3000 OPNs.
1	Product Version type/length byte	0x01	8-bit binary 7:6 – 00b 5:0 – 000001b (1 byte of data)

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
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'.

continued...

Field Length in Bytes	Field Description	Field Values	Field Encoding
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)	0x9D	*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

Document Version	Changes
2019.11.25	Initial Release.