OCP Debug Card with LCD
Spec v1.0

Author:
Whitney Zhao, Hardware Engineer, Facebook
## 1. Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/17/2016</td>
<td>Whitney Zhao</td>
<td>Version 0.1 draft</td>
</tr>
<tr>
<td>1/18/2017</td>
<td></td>
<td>V0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add HW section for both base board and debug card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Change section sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add section 9 and 10</td>
</tr>
<tr>
<td>2/10/2017</td>
<td></td>
<td>- Add pics to section 5 and 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Minor contents update</td>
</tr>
<tr>
<td>6/12/2017</td>
<td>Whitney Zhao</td>
<td>V0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update critical sensor frame information</td>
</tr>
<tr>
<td>6/20/2017</td>
<td></td>
<td>V0.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add command 06h for user setting frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add section for ANSI escape codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update system info frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Type updates</td>
</tr>
<tr>
<td>6/22/2017</td>
<td></td>
<td>V0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add debug card cable info</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update “get frame from BMC” command to 05h</td>
</tr>
<tr>
<td>6/23/2017</td>
<td></td>
<td>V0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update post code frame information to support 5 pages post code history</td>
</tr>
<tr>
<td>7/21/2017</td>
<td></td>
<td>V0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update command 05h</td>
</tr>
<tr>
<td>9/8/2017</td>
<td></td>
<td>V0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Update sensor frame info and critical SEL frame info</td>
</tr>
<tr>
<td>12/30/2017</td>
<td></td>
<td>V1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Correct the description for section 7 drawing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add description for “BMC disconnected” in section 6.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Add some examples for critical SEL messages</td>
</tr>
<tr>
<td>1/18/2018</td>
<td></td>
<td>- Update the license section to updated version</td>
</tr>
</tbody>
</table>
Contributions to this Specification are made under the terms and conditions set forth in Open Compute Project Contribution License Agreement ("OCP CLA") ("Contribution License") by:

Facebook Inc.

You can review the signed copies of the applicable Contributor License(s) for this Specification on the OCP website at http://www.opencompute.org/products/specsanddesign

Usage of this Specification is governed by the terms and conditions set forth in Open Compute Project Hardware License – Copyleft ("OCPHL Reciprocal") ("Specification License").

You can review the applicable Specification License(s) executed by the above referenced contributors to this Specification on the OCP website at http://www.opencompute.org/participate/legal-documents/

**Note:** The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

None

**NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.**
2. Scope

The specification defines the OCP Debug Card with LCD for a server system debug.

3. Table of Contents

1. Revision History .......................................................................................................................... 2
2. Scope ....................................................................................................................................... 4
3. Table of Contents ....................................................................................................................... 4
4. Hardware Overview ................................................................................................................... 6
   4.1 Block Diagram ....................................................................................................................... 6
   4.2 Debug card-baseboard Interface/cable ................................................................................. 7
      4.2.1 Remapped Pin ................................................................................................................... 8
      4.2.2 Signal Voltage level ......................................................................................................... 8
   4.3 LCD Debug Card HW ............................................................................................................ 8
      4.3.1 Components ..................................................................................................................... 8
      4.3.2 Placement .......................................................................................................................... 10
   4.4 Baseboard HW ....................................................................................................................... 10
      4.4.1 UART ............................................................................................................................... 10
      4.4.2 I2C ................................................................................................................................... 11
      4.4.3 Present Pin ....................................................................................................................... 11
5. FW Design ............................................................................................................................... 12
   5.1 Roles and communication .................................................................................................... 12
      5.1.1 BMC (0x20h) .................................................................................................................... 12
      5.1.2 MCU (0x60h) ................................................................................................................... 13
      5.1.3 GPIO-I2C Expander (0x4Eh) ............................................................................................ 13
   5.2 Communication process ....................................................................................................... 13
      5.2.1 Initial process when debug card is plugged in to system ............................................... 13
      5.2.2 MCU polling BMC run time process: (Loop) ................................................................. 13
6. LED Panel Content and Display ............................................................................................... 13
   6.1 Display Frames ..................................................................................................................... 15
      6.1.1 POST Code Frame .......................................................................................................... 15
      6.1.2 System Info Frame .......................................................................................................... 16
      6.1.3 Critical SEL Frame ......................................................................................................... 17
      6.1.4 Critical Sensor Frame .................................................................................................... 18
      6.1.5 GPIO Status Frame ......................................................................................................... 19
      6.1.6 User Setting Frame ......................................................................................................... 20
      6.1.7 BMC Error Code Frame ................................................................................................. 21
   6.2 ANSI Escape codes .............................................................................................................. 21
7. OEM Command Definition ...................................................................................................... 22
   7.1.1 Request .............................................................................................................................. 22
   7.1.2 Response ............................................................................................................................ 22
8. Debug Card ME design................................................................................................................. 26
9. Appendix ....................................................................................................................................... 29
4. Hardware Overview

The LCD Debug Card is intended to improve and replace the existing Bluetooth Debug card. It carries all the features of the existing Bluetooth debug card. The major improvements are:

- A text-rich user interface with an LCD
- More baseboard front I/O space
- Improved mechanical design for plug-in and removal
- The electric interface is serialized

4.1 Block Diagram

Figure 2-1 below is the diagram for the baseboard to the LCD Debug Card. The baseboard connects to the LCD Debug Card through a USB interface. Figure 2-2 illustrates the functional block diagram of the LCD Debug Card.
4.2 Debug card-baseboard Interface/cable

LCD Debug Card will be plugged to baseboard remapped USB3.0 connector through a ribbon cable (see the picture below).

LXP L64U3005-SD-R or equivalent should be used as the USB cable.
4.2.1. Remapped Pin

The USB debug connector remaps 5X USB3.0 signals to UART, Present and I2C signals. UART will pass the console data to debug card; Present pin will tell it’s the debug card or USB device plugged in; The I2C allows MCU to communicate with either BMC or access post code through an I2C GPIO expander located on the baseboard. The detailed information for the remapping is as below:

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Signal Name</th>
<th>New Remapped Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VBUS</td>
<td>VBUS</td>
</tr>
<tr>
<td>2</td>
<td>D-</td>
<td>D-</td>
</tr>
<tr>
<td>3</td>
<td>D+</td>
<td>D+</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>Ground for power return</td>
</tr>
<tr>
<td>5</td>
<td>StdA_SSRX-</td>
<td>SCL</td>
</tr>
<tr>
<td>6</td>
<td>StdA_SSRX+</td>
<td>SDA</td>
</tr>
<tr>
<td>7</td>
<td>GDN_DRAIN</td>
<td>PRSNT</td>
</tr>
<tr>
<td>8</td>
<td>StdA_SSTX-</td>
<td>UART TX</td>
</tr>
<tr>
<td>9</td>
<td>StdA_SSTX+</td>
<td>UART RX</td>
</tr>
</tbody>
</table>

The USB3.0 connector will downgrade to support USB2.0 speed only in order to support the debug card.

4.2.2. Signal Voltage level

<table>
<thead>
<tr>
<th>Signal</th>
<th>Voltage level</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB3 RX/TX</td>
<td>0.8-1.2v</td>
</tr>
<tr>
<td>UART</td>
<td>3.3v</td>
</tr>
<tr>
<td>SCL/SDA</td>
<td>3.3v</td>
</tr>
</tbody>
</table>

4.3 LCD Debug Card HW

4.3.1 Components

- Modules
- 7 segment LED
- LCD panel: VO12864T-GFW-B601A from Vitek display is used for the debug card. It has 128x64 dots and can display 8 rows and 16 letters on each row.
- Bluetooth module RN42

**Chips**
- Micro controller. Micro controller communicates with BMC and GPIO expander on baseboard
- FDTI, UART to USB chip

**Human interface**
- Power/Reset/UART select button
- 5-way switch: The 5 way switch allows the user to page up or page down through the debug information on LED panel as well as switch to different debug information frames (for example, post code details → system information → BMC critical SEL → critical sensor → user settings).
- Bluetooth on/off switch: turn the battery power on/off to enable/disable Bluetooth module.

**LED**
- MCU HB
  - Green blink, Heartbeat for the micro controller on debug card
- Bluetooth LED
  - Green, blink at 2Hz if Bluetooth module enabled and no link
  - Solid Green when Bluetooth connecting or when data transfer
- Low Battery LED: red LED on when battery lower than 10%; it’s off otherwise

**Connectors**
- Micro USB
- USB 2.0 type A connector
- USB3.0 connector to baseboard

**Coin battery:** The LCD Debug Card also have chargeable coin battery to supply power for bluetooth module while debug card is not plugged on a system. The battery can sustain ~2 hours if fully charged. The LCD Debug Card battery can be charged through any usb2.0/3.0 port.
4.3.2 Placement

UART signal goes to baseboard BMC or Host UART. UART select button is used to switch
between host or BMC UART.

4.4.2 I2C

The I2C shall connects to baseboard one of BMC I2C ports and the GPIO expander I2C.

4.4.3 Present Pin

4.4.3.1 Support LCD Debug Card Only

The present pin has weak pull down at 100K on the baseboard side. Present pin status in the table below describes which device is plugged to baseboard.

<table>
<thead>
<tr>
<th>Present pin Status</th>
<th>Device plugged to baseboard</th>
<th>Implementation on device side</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>USB3</td>
<td>GND</td>
</tr>
<tr>
<td>1</td>
<td>LCD Debug Card</td>
<td>PU to 10K</td>
</tr>
<tr>
<td>0</td>
<td>USB2 or None</td>
<td>Weak PD on baseboard side</td>
</tr>
</tbody>
</table>

The baseboard design should disconnect of I2C/UART when the USB3.0 device is plugged in so different voltage of UART/I2C will not affect the USB3 device. A reference design is below by using debug card present pin:
4.4.3.2 Support Both Debug Cards

When the baseboard is designed to support both Bluetooth Debug Card V1 and the debug card with LCD, besides the usage of the present pin to disconnect the I2C/UART, the baseboard design should also consider to prioritize the UART paths to/from both debug cards in case of any confliction. A reference design is below:

![Reference Design Diagram]

5. FW Design

5.1 Roles and communication

There are three roles in this scope: BMC, Debug Card MCU, and GPIO-I2C expander.

![Role Diagram]

5.1.1. BMC (0x20h)

1. Response the message after receiving the request from MCU (via IPMB).
2. Send Post Code to GPIO-I2C expander (via GPIOs).
3. Configure GPIO-I2C expander to input when Debug Card is removed.
*All I2C addresses in this document are stated as 8-bit address.*

5.1.2. **MCU (0x60h)**

1. Handshake with BMC (via IPMB) to get the post code description, GPIO signal description, number of Frames.
2. Query System Post Code from GPIO-I2C expander (via I2C, be an I2C master) and update ‘POST Frame’ content.
3. Query GPIO status and update ‘GPIO Frame’ content.
4. Read the content for all available frames from BMC, polling BMC for updates every 5s.
5. Periodically(every 1s) check with BMC on availability.
6. MCU will retry 5 times if BMC has no response to the availability query. After 5 times, it will time out and MCU will have System Info/Critical SEL/Critical Sensor Frames/user settings etc. to display “BMC disconnected” under the frame title.

5.1.3. **GPIO-I2C Expander (0x4Eh)**

2. Response Post Code or GPIOs status to MCU (via I2C, be an I2C slave).
3. When debug card is removed, BMC will reset expander and configure all GPIO pins to input. Only when there is activity for UART select/ Reset / Power buttons, these 3 GPIOs will be configured as output and send the signals out. After that, GPIO pins will be configured as input again.

5.2 Communication process

5.2.1. **Initial process when debug card is plugged in to system**

1. Get Frame Information (01h).
2. Get GPIO expander IOs description (04h).
3. Get POST code description (03h).
4. Get Frame content. **Note:** Might have to send multiple times for multi-page frames.
5. Get platform information.

5.2.2. **MCU polling BMC run time process: (Loop)**

1. Get update frame status (02h)
   a. If it has update frame, send command to get latest content.
2. Back to 1.

6. **LED Panel Content and Display**

The MCU maintains cached content for all the frames that needs to be displayed on the LED panel. The user can cycle through all frames by using Detector Switch’s left and right buttons. Some of the frames might have multiple pages of content, and the user can go
through these pages by using up and down buttons.

The MCU handles the pre-defined frames as described below:

![Figure 3 Pre-defined frames]

The user can view various frames by onboard detector switch that provides four direction functions: **up, down, left, right** and one selection function: **Select**.

**Left or Right**

The left or right function uses to rotate different frame information to be shown on LED panel: POST code -> System Info -> BMC Critical SEL -> Critical sensors -> GPIO status -> User Settings.

**Up or Down**

The up or down function uses to view multiple pages (if available) in the same frame for more content.

**Select**

The select function will be only useful in “User Settings” page to identify user selection. “User Settings” frame is described in more detail in a later section. In Figure 1 (above), the arrow from A frame to B frame means that user can switch frame from A to B by controlling correspond function of detector switch.
The following list shows proposed property and display format of the frame:

- Maximum rows: 8
- First row displays title of subjects and page information with bold character
- Other rows display content

6.1 Display Frames

Post code frame and GPIO status frame is defined through MCU as MCU will get the post code and GPIO status from GPIO expander.

User Setting frame is the only frame that MCU will get the content from BMC to see which user settings that will be open to user to change.

This spec defines the other frame examples such as Sys-info frame, Critical SEL frame and Critical Sensor frame. The baseboard BMC FW can define any frames by the needs of each project.

6.1.1. POST Code Frame

The content of this frame contains 5 pages of POST codes and user readable string representation for those codes. Since the POST codes are specific to a given platform, MCU sends IPMB request to BMC to get the look table that shows the association of POST code and corresponding user readable string. Based on number of POST codes to be shown, this frame might contain multiple pages of information. MCU will periodically poll the GPIO-I2C expander and update this frame content (with possibly multiple pages). To provide better use case, it is suggested to keep the recent POST code at the top of the frame i.e. maintain reverse-chronological list of POST codes.
Some post code may have different description during different post phases. When debug card MCU gets post code description from BMC, byte4 will indicate the post code is for which phase. Pls refer to the command section for the details. MCU will based on the phase information to decide which description should be shown on LCD.

Post Code Frame and buffer will be cleared when the system powered off. MCU gets system power status by polling Get_Chassis_Status command to BMC. If system status is on MCU will turn on 7 segment LED otherwise 7 segment LED will be off. If BMC somehow does not response power status to MCU, MCU will consider the system power status is on by default.

6.1.2. System Info Frame

The content of this frame contains pre-defined key information as below:

- Serial Number
- Part Number
- BMC IP
- BMC FW ver.
- BIOS FW ver.
- ME status
- Board ID
- Sys Conf. info: CPU/Mem/HDD etc. info. An example is:
  o CPU: Type/Frequency/Cores
  o MEM: Vendor/Frequency/Total memory Capacity
  o HDD: Vendor/Model number
- Battery charging status and percentage
- MCU boot loader version
- MCU FW version
### 6.1.3. Critical SEL Frame

The content of this frame contains all pre-defined BMC critical SELs. The format will not be same as BMC SEL. It will be short and with extra information for debug. Some pre-defined critical SEL examples is shown below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 Temp UCR</td>
<td>P0 Temp UCR XRC – Assert/Deassert</td>
</tr>
<tr>
<td>P1 Temp UCR</td>
<td>P1 Temp UCR XRC – Assert/Deassert</td>
</tr>
<tr>
<td>Memory loop code</td>
<td>DIMM XX initial fails</td>
</tr>
<tr>
<td>CATERR/IERR</td>
<td>CPUx IERR</td>
</tr>
<tr>
<td>Machine Check Error</td>
<td>MACHINE_CHK_ERR,Uncorrectable/Correctable bank Number XX</td>
</tr>
<tr>
<td>Other IIO Error</td>
<td>CPU X, Error ID 0xYY – Source</td>
</tr>
<tr>
<td>PROCHOT#</td>
<td>CPU FPH by XXX Assert</td>
</tr>
<tr>
<td></td>
<td>CPU FPH Deassert</td>
</tr>
<tr>
<td>Memory ECC/UECC</td>
<td>DIMMxx ECC/UECC err</td>
</tr>
<tr>
<td>Fan Fail</td>
<td>Fan0/1 fail</td>
</tr>
<tr>
<td>PCIe ECC error</td>
<td>XXXX PCIe err</td>
</tr>
<tr>
<td>Power Fail</td>
<td>XXXX power rail fails</td>
</tr>
</tbody>
</table>

XX: refer to each platform BIOS spec. Ex: MACHINE_CHK_ERR,Correctable bank Number 5

Ex: CPU0/1 IERR/CATERR or MSMI

Ex: IIO_ERR CPU 0, Error ID 0x41 – Reserved

Trigger source XXX: UV, OC, timer exp, pmbus alert
6.1.4. Critical Sensor Frame

The content of this frame contains all pre-defined BMC critical sensors. Some pre-defined critical sensor examples is shown below:

<table>
<thead>
<tr>
<th>Sensor Name</th>
<th>Messages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0 temp</td>
<td>P0_TEMP:XXC</td>
<td>show “P0_temp XXC/UCT” If out of range and invert the font</td>
</tr>
<tr>
<td>CPU1 temp</td>
<td>P1_TEMP:XXC</td>
<td>show “P1_temp XXC/UCT” If out of range and invert the font</td>
</tr>
<tr>
<td>HSC Power</td>
<td>HSC_PWR:XXX.XW</td>
<td>Show “HSC Pwr XXX.Xw/UCT” If out of range and invert the font</td>
</tr>
<tr>
<td>HSC voltage</td>
<td>HSC_VOL:XX.XXV</td>
<td>show “XX.XXV/LCT” or “XX.XXV/UCT” If out of range and invert the font</td>
</tr>
<tr>
<td>Fan0/1 speed</td>
<td>Fan0/1:XXXXRPM</td>
<td>Show “XXXXRPM/UCT” or “XXXXRPM/LCT” if out of range and invert the font</td>
</tr>
<tr>
<td>Inlet Temp</td>
<td>Intel_TEMP:XXC</td>
<td>Show “XXC/UCT” if out of range and invert the font</td>
</tr>
<tr>
<td>CPU0/1 VR temp</td>
<td>P0/P1_VR_TEMP:XXXC</td>
<td>Show “XXC/UCT” if out of range and invert the font</td>
</tr>
<tr>
<td>CPU0/1 VR PIN</td>
<td>P0/P1_VR_Pwr:XX.XW</td>
<td>Show “XXW/UCT” if out of range and invert the font</td>
</tr>
<tr>
<td>DIMM Temp</td>
<td>DIMMXX_TEMP:XXC</td>
<td>Show “XXC/UCT” if out of range and invert the font</td>
</tr>
</tbody>
</table>
If any sensor is out of the threshold, the whole screen should blink and invert the color for the sensors which out of the threshold. An example is as below:

If any sensor is out of the threshold, the whole screen should blink and invert the color for the sensors which out of the threshold. An example is as below:

6.1.5. GPIO Status Frame

The content of this frame contains all the GPIO signal information with user readable GPIO signal name and its status i.e. ‘CPU_CARERR_MSMI_LVT3_N’. Since GPIO signals are specific to a given platform, MCU sends IPMB request to BMC to get the look up table that shows the association of GPIO signal and corresponding user readable string. Based on number GPIO signals to be shown, this frame might contain multiple pages of information. MCU will periodically poll GPIO-I2C expander and update this frame content (with possibly multiple pages).

The following table shows the GPIO pin definitions on I2C expander on platform Tioga Pass:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Usage</th>
<th>Messages</th>
<th>Direction (In the perspective of PCA9555)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO1_0</td>
<td>RST_BTN_N</td>
<td>IO1_0: FM_DBG_RST_BTN</td>
<td>Output</td>
</tr>
<tr>
<td>IO1_1</td>
<td>PWR_BTN_N</td>
<td>IO1_1: FM_PWR_BTN</td>
<td>Output</td>
</tr>
<tr>
<td>IO1_2</td>
<td>PWRGD_SYS_PWROK</td>
<td>IO1_2: SYS_PWROK</td>
<td>Input</td>
</tr>
<tr>
<td>IO1_3</td>
<td>RST_PLTRST_N</td>
<td>IO1_3: RST_PLTRST</td>
<td>Input</td>
</tr>
<tr>
<td>IO1_4</td>
<td>PWRGD_DSW_PWROK</td>
<td>IO1_4: DSW_PWROK</td>
<td>Input</td>
</tr>
<tr>
<td>IO1_5</td>
<td>FM_CPU_CATERR_MSMI_LVT3_N</td>
<td>IO1_5: FM_CATERR_MSMI</td>
<td>Input</td>
</tr>
</tbody>
</table>
6.1.6. User Setting Frame

The content of this frame contains some boot options for user configuration. Tioga Pass implements two boot options: power policy and boot order:

Besides the above pre-defined frames, MCU will check with BMC on any other frames to be displayed. This process involves MCU to get the number of different frames to be displayed.
and requesting content for each frame. Since each frame can contain multiple pages, the
MCU shall request the content for all pages for a multi-page frame.

6.1.7. BMC Error Code Frame

If BMC defines some error code for debugging, BMC can also add BMC error code Frame
and let MCU to display. It will show current BMC error code and the description of the
code. It is similar to the Post code frame for host. Each project may implement this frame
accordingly.

6.2 ANSI Escape codes

ANSI escape codes (or escape sequences) are a method using in-band signaling to control
the formatting, color, and other output options on video text terminals. To encode this
formatting information, certain sequences of bytes are embedded into the text, which the
terminal looks for and interprets as commands, not as character codes.

The below table shows the detail information of this debug card’s escape sequences (ESC,
ASCII decimal 27 / hex 0x1B) and CSI codes that will be used in this debug:

<table>
<thead>
<tr>
<th>Escape sequence</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LCD FW Non-CSI codes</strong>&lt;br&gt;Start with the characters ESC the final byte is technically any character in the range 64–95 (hex 0x40–0x5F, ASCII @ to ~)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC 'B'</td>
<td>Battery power percentage; overwrite the following data</td>
<td></td>
</tr>
<tr>
<td>ESC 'U'</td>
<td>MCU Bootloader version; overwrite the following data</td>
<td></td>
</tr>
<tr>
<td>ESC 'R'</td>
<td>MCU Runtime firmware version; overwrite the following data</td>
<td></td>
</tr>
<tr>
<td><strong>CSI codes - reference: <a href="https://en.wikipedia.org/wiki/ANSI_escape_code">https://en.wikipedia.org/wiki/ANSI_escape_code</a></strong>&lt;br&gt;Start with the characters ESC and [ (left bracket/0x5B), the final byte is technically any character in the range 64–126 (hex 0x40–0x7E, ASCII @ to ~)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESC '[' n 'm'</td>
<td>n can be zero or more SGR parameters separated with ','; With no parameters, n is treated as 0 (Reset)</td>
<td>ESC [ 5 ; 7 m represent the following data are Blink and Reversed; ESC [ m represent the following data are Reset to normal</td>
</tr>
</tbody>
</table>
7. OEM Command Definition

The Debug Card sends the IPMB request to BMC, and BMC response the brief message back according to the request command.

IPMB data format between BMC and MCU:

### 7.1.1 Request

<table>
<thead>
<tr>
<th>rsAddr</th>
<th>NetFn / rsLun</th>
<th>Header checksum</th>
<th>rqAddr</th>
<th>rqSeq</th>
<th>cmd</th>
<th>Request data bytes (0 or more)</th>
<th>Data checksum</th>
</tr>
</thead>
</table>

### 7.1.2 Response

<table>
<thead>
<tr>
<th>rqAddr</th>
<th>NetFn / rqLun</th>
<th>Header checksum</th>
<th>rsAddr</th>
<th>rqSeq</th>
<th>cmd</th>
<th>Completion code</th>
<th>Response data bytes (0 or more)</th>
<th>Data checksum</th>
</tr>
</thead>
</table>

---

### Net Function = 3Ch  LUN = 00b

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Request, Response Data</th>
<th>Description</th>
</tr>
</thead>
</table>
| 01h  | Get Frame Information    | Request: Byte [0:2] – IANA ID  
Response: Byte 0 - completion code  
Byte [1:3] – IANA ID  
Byte 1 – Number of Frames |             |
| 02h  | Get update Frame status  | Request: Byte [0:2] – IANA ID  
Response: Byte 0 - Completion Code  
Byte [1:3] – IANA ID  
Byte 4 – |             |
<table>
<thead>
<tr>
<th>Net Function</th>
<th>LUN</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Ch</td>
<td>00b</td>
<td>Get POST code description</td>
</tr>
</tbody>
</table>

**Request:**
- Byte [0:2] – IANA ID
- Byte 3 – POST code index
- Byte 4 – POST code Phase.
  - 01h: Phase 1
  - 02h: Phase 2

**Response:**
- Byte 0 - completion code
- Byte [1:3] – IANA ID
- Byte 4 – current POST code index
- Byte 5 – next POST code index
- Byte 6 – POST code Phase
  - 01h: Phase 1
  - 02h: Phase 2
- Byte 7 – check if it is the last one POST code
  - 00h: this is not the last one of POST code
  - 01h: The last one available POST code
- Byte 8 – length (n)
- Byte 9: (n+1) - human readable string (ASCII format)

**MCU get POST code description from BMC.**
**Post code description definition refer to BIOS spec.**

<table>
<thead>
<tr>
<th>4h</th>
<th>Get GPIO expander IOs description</th>
</tr>
</thead>
</table>

**Request:**
- Byte [0:2] – IANA ID
- Byte 3 – GPIO IO index,
  - 10h: PCA9555 P10
  - 11h: PCA9555 P11
  - 12h: PCA9555 P12
  - 13h: PCA9555 P13
  - 14h: PCA9555 P14
  - 15h: PCA9555 P15
  - 16h: PCA9555 P16
  - 17h: PCA9555 P17
  - FFh: first available pin.

**Response:**
- Byte 0 - completion code
- Byte [1:3] – IANA ID
- Byte 4 – current GPIO IO index
- Byte 5 – next GPIO IO index
- Byte 6 – Pin active level

**MCU to get GPIO expander IO pins definition and description**
### Net Function = 3Ch  LUN = 00b

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>low level active</td>
</tr>
<tr>
<td>1:</td>
<td>high level active</td>
</tr>
<tr>
<td>Byte 7 – function pin define</td>
<td></td>
</tr>
<tr>
<td>00h:</td>
<td>input pin only</td>
</tr>
<tr>
<td>01h:</td>
<td>Power button</td>
</tr>
<tr>
<td>02h:</td>
<td>Reset button</td>
</tr>
<tr>
<td>03h:</td>
<td>UART switch button</td>
</tr>
<tr>
<td>04h~FFh:</td>
<td>reserved</td>
</tr>
<tr>
<td>Byte 8 – length (n)</td>
<td></td>
</tr>
<tr>
<td>Byte 9:(n+ 4) - human readable string (ASCII format)</td>
<td></td>
</tr>
</tbody>
</table>

#### 06h  Control Panel Operation

**Request:**
- Byte [0:2] – IANA ID
- Byte 3 – Control Panel Number
  - 1-based number, 01h is top Control Panel
- Byte 4 – Operation
  - 00h: Get Description
  - 01h: Select Item
  - 02h: Back
- Byte 5 – Item Number
  - 00h: Title of Control Panel
  - Others: Items

**Response:**
- Byte 0 – Completion Code
  - C9h: Parameter out of range
- Byte [1:3] – IANA ID
- Byte 4 – Control Panel Number
  - When Operation is 00h, return requested Panel
  - Otherwise, return new Control Panel Number
- Byte 5 – Item Number
- Byte 6 – Length of description
- Byte 7:N – human readable description string

MCU only get 1 Control Panel information at a time, fetch all items from item 00h(title) in the panel, until Completion Code C9h(no more item). BMC will return new Panel when MCU send “Select Item” or “Back” operation, then MCU can get the items of new panel.

#### 05h  Get frame from BMC

**Request:**
- Byte [0:2] – IANA ID
- Byte 3 – Frame Number
- Byte 4 – Page Number, start from 01h

**Response:**
- Byte 0 - Completion Code
- Byte [1:3] – IANA ID
- Byte 4 – Frame Number
- Byte 5 – Page Number, start from 01h
- Byte 6 – Next page number
Net Function = 3Ch  LUN = 00b

| FFh: no next page need to be updated |
| Byte 7 – length |
| Byte 8:N – frame buffer data (ASCII format), the max data is 128 bytes for 1 page |

Here are some usage scenarios for command 06h:

Get Main panel

- Req: Panel 1 Operation 0 Item 0 Res: Panel 1 item 0 “User Setting”
- Req: Panel 1 Operation 0 Item 1 Res: Panel 1 item 1 “Power Policy”
- Req: Panel 1 Operation 0 Item 2 Res: Panel 1 item 2 “Boot Sequence”
- Req: Panel 1 Operation 0 Item 3 Res: No more items (Completion Code: C9h)

User Press button on item 1, Power Policy

- Req: Panel 1 Operation 1 Item 1 Res: Panel 2 item 0 “Power Policy”
- Req: Panel 2 Operation 0 Item 0 Res: Panel 2 item 0 “Power Policy” (Optional command, previous response already return the title)
- Req: Panel 2 Operation 0 Item 1 Res: Panel 2 item 1 “Always Power On”
- Req: Panel 2 Operation 0 Item 2 Res: Panel 2 item 2 “*Last Power Status”
- Req: Panel 2 Operation 0 Item 3 Res: Panel 2 item 3 “Always Power Off”
- Req: Panel 2 Operation 0 Item 4 Res: No more items (Completion Code: C9h)

User Press button on item 3, Always Power Off

- Req: Panel 2 Operation 1 Item 3 Res: Panel 2 item 0 “Power Policy” (BMC change the internally setting)
- Req: Panel 2 Operation 0 Item 0 Res: Panel 2 item 0 “Power Policy” (Optional command, previous response already return the title)
- Req: Panel 2 Operation 0 Item 1 Res: Panel 2 item 1 “Always Power On”
- Req: Panel 2 Operation 0 Item 2 Res: Panel 2 item 2 “Last Power Status”
- Req: Panel 2 Operation 0 Item 3 Res: Panel 2 item 3 “*Always Power Off”
- Req: Panel 2 Operation 0 Item 4 Res: No more items (Completion Code: C9h)

User Press Left Button, Back to upper control panel

- Req: Panel 2 Operation 2 Item 3 Res: Panel 1 item 0 “User Setting” (BMC don’t care the Item number when Operation is 2h, Back)
Req: Panel 1 Operation 0 Item 0 Res: Panel 1 item 0 “User Setting” (Optional command, previous response already return the title)
Req: Panel 1 Operation 0 Item 1 Res: Panel 1 item 1 “Power Policy”
Req: Panel 1 Operation 0 Item 2 Res: Panel 1 item 2 “Boot Sequence”
Req: Panel 1 Operation 0 Item 3 Res: No more items (Completion Code: C9h)

8. Debug Card ME design

**Dimension**

Back side design: for bluetooth area, there should be a transparent area to let end user to check the MAC for the bluetooth module.
Color proposal

<table>
<thead>
<tr>
<th>Part</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover Front</td>
<td>Green Panton 375C</td>
</tr>
<tr>
<td>Cover Rear</td>
<td>Green Panton 375C</td>
</tr>
<tr>
<td>SWay SW Cap</td>
<td>Green Panton 375C</td>
</tr>
<tr>
<td>BT Knob</td>
<td>Green Panton 375C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Switch</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART.SEL</td>
<td>Brown</td>
</tr>
<tr>
<td>RST</td>
<td>Black</td>
</tr>
<tr>
<td>Power</td>
<td>Red</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Battery</td>
<td>Red when Battery &lt;10%</td>
</tr>
<tr>
<td>BT</td>
<td>Green</td>
</tr>
<tr>
<td>MCU HB</td>
<td>Green</td>
</tr>
</tbody>
</table>

Art work
9. Appendix

- Schematic
- CAD file
- DXF for debug card
- 2D/3D drawing for covers