OCP Debug Card with LCD

Spec v1.0

Author:

Whitney Zhao, Hardware Engineer, Facebook

Revision History

Date	Name	Description
10/17/2016	Whitney Zhao	Version 0.1 draft
1/18/2017		V0.2 - Add HW section for both base board and debug card - Change section sequence - Add section 9 and 10
2/10/2017		- Add pics to section 5 and 9 - Minor contents update
6/12/2017		V0.3 - Update critical sensor frame information
6/20/2017		V0.4 - Add command 06h for user setting frame - Add section for ANSI escape codes - Update system info frame - Type updates
6/22/2017		V0.5 - Add debug card cable info - Update "get frame from BMC" command to 05h
6/23/2017		V0.6 - Update post code frame information to support 5 pages post code history
7/21/2017		V0.6 - Update command 05h
9/8/2017		V0.7 - Update sensor frame info and critical SEL frame info
12/30/2017		V1.0 - Correct the description for section 7 drawing - Add description for "BMC disconnected" in section 6.1.2

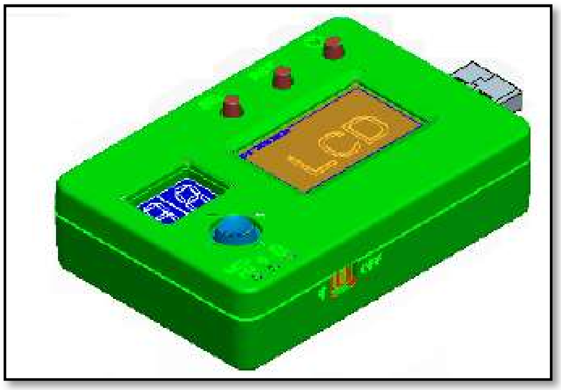
Table of Contents

1. Overview

The specification defines the OCP LCD Debug Card.

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



2. 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.

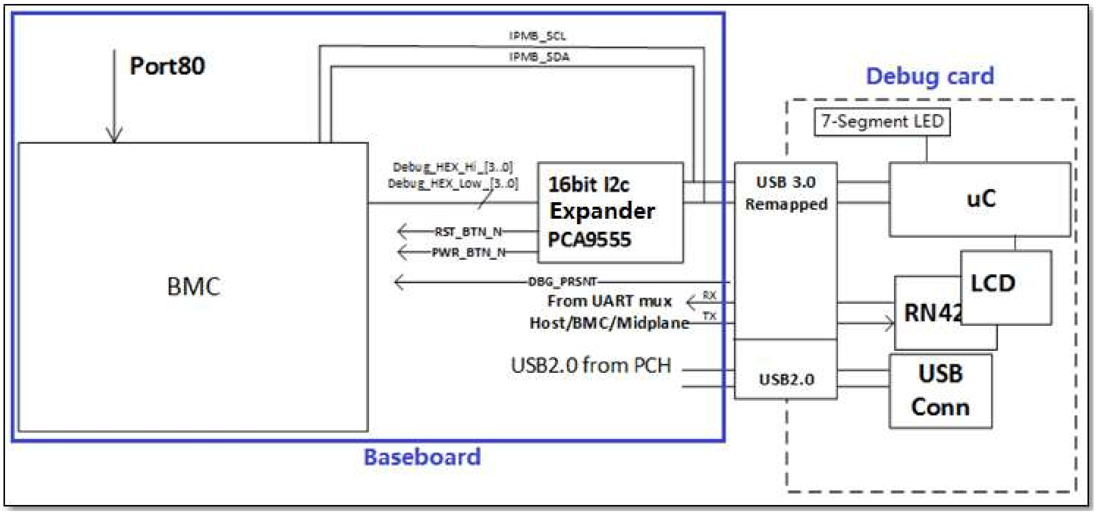


Figure 2-1 Baseboard to debug card diagram

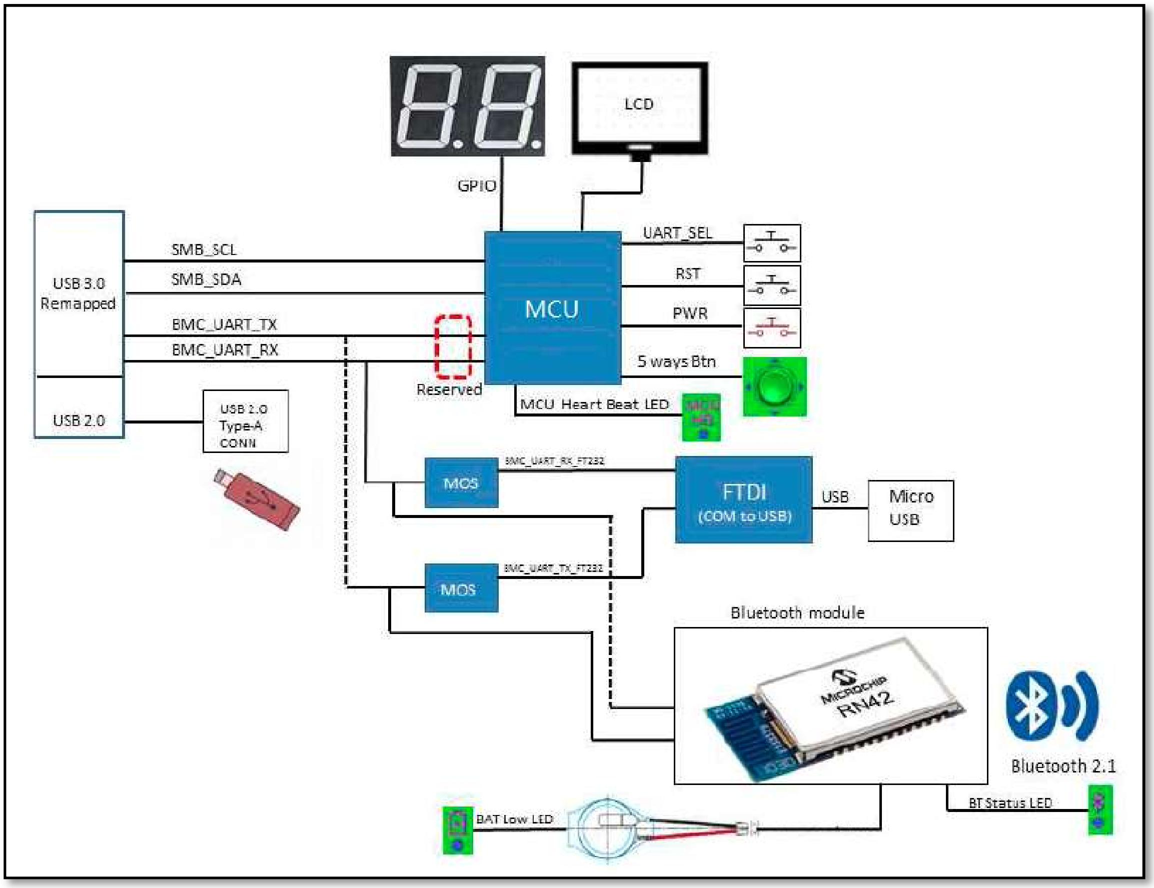
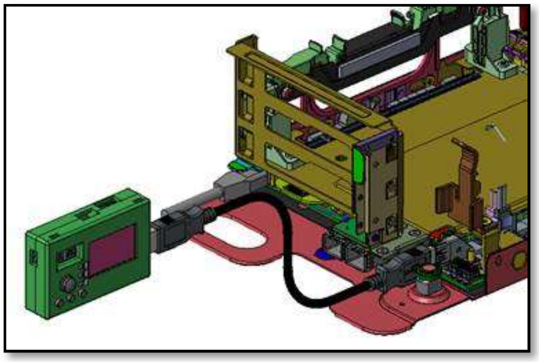


Figure 2-2 LCD Debug Card to system block diagram

3. 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.

3.1. 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:

Pin Number	Signal Name	New Remapped Pin
1	VBUS	VBUS
2	D-	D-
3	D+	D+
4	GND	Ground for power return
5	StdA_SSRX-	SCL
6	StdA_SSRX+	SDA
7	GDN_DRAIN	PRSNT
8	StdA_SSTX-	UART TX
9	StdA_SSTX+	UART RX

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

3.2. Signal Voltage level

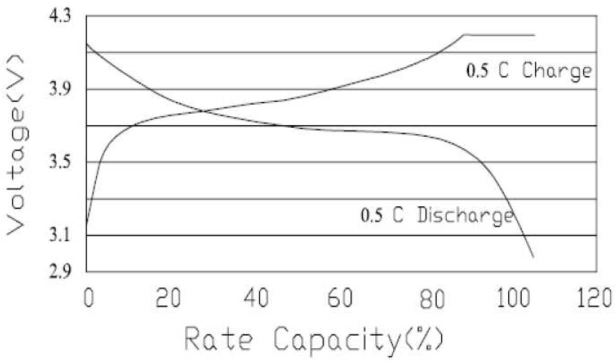
Signal	Voltage level
USB3 RX/TX	0.8-1.2v
UART	3.3v
SCL/SDA	3.3v

4. LCD Debug card HW

4.1. 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, Heartbeat for the micro controller on debug card, blink at 1hz
  - 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

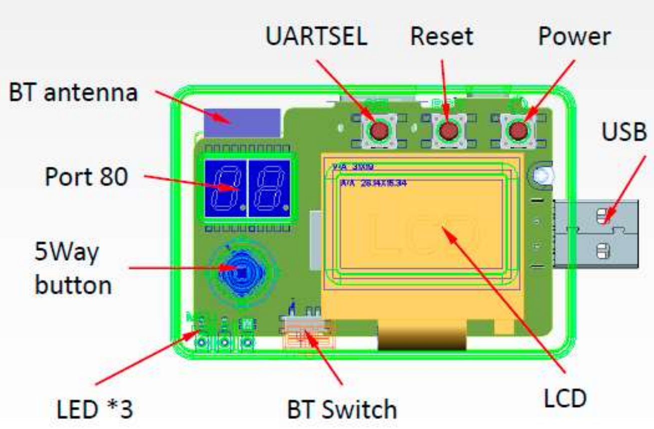
charged through any usb2.0/3.0 port.

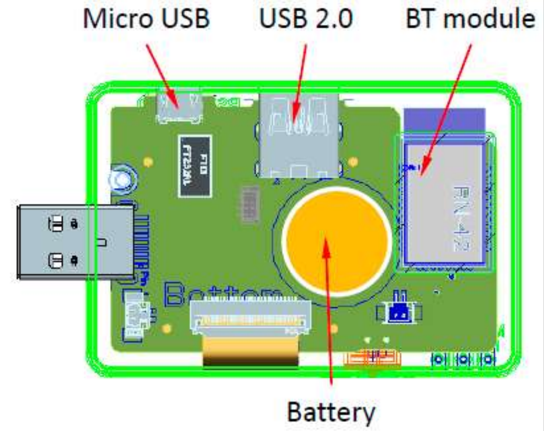


Charge:
(capacity*1.5)/charge current = (70mAh*1.5)/35mA = 3h
So it takes about 3 hours that the battery almost full charge

Discharge:
In our design, it will consume about 25mA and use 90% of battery capacity
(nominal voltage*capacity*90%)/(normal voltage*consume current)
= (3.7V*70mAh*90%)/(3.3V*25mA) = 2.82h
So the battery can sustain about 2.8 hours

4.2. Placement





5. Baseboard HW

5.1. UART

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

5.2. I2C

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

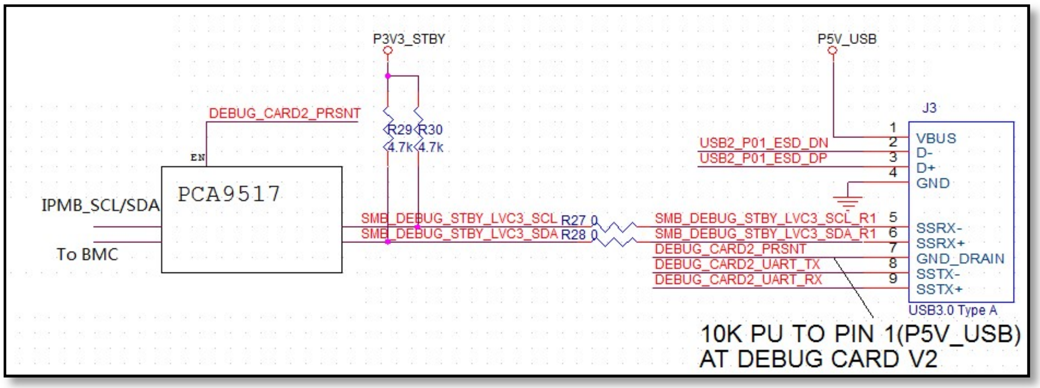
5.3. Present Pin

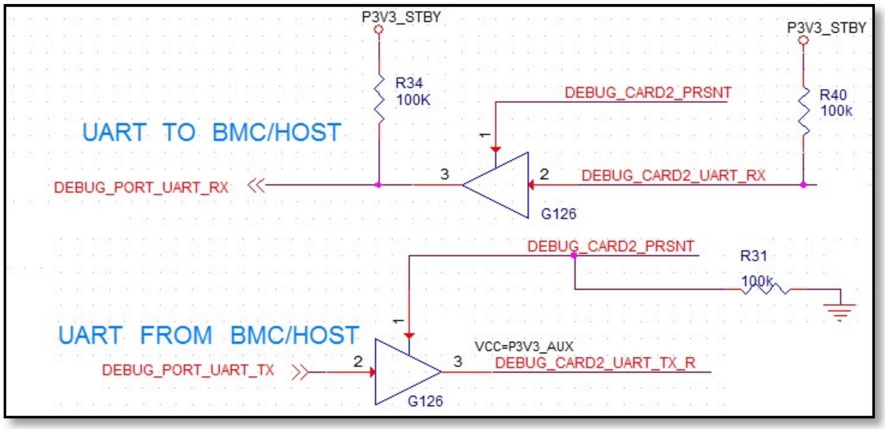
5.3.1. Support LCD Debug Card Only

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.

Present pin Status	Device plugged to baseboard	Implementation on device side
0	USB3	GND
1	LCD Debug Card	PU to 10K
0	USB2 or None	Weak PD on baseboard side

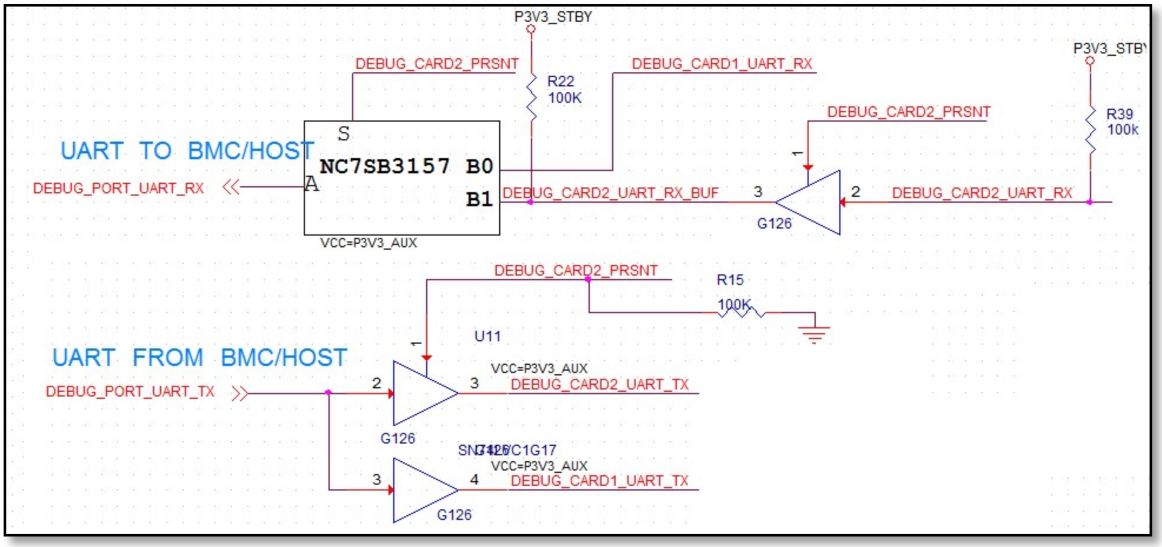
The baseboard design should disconnect of I2C/UART when the USB3.0 device is plugged in so different voltage of UART/I2C will not affect the USB3 device. A reference design is below by using debug card present pin:





5.3.2. Support Both Debug Cards

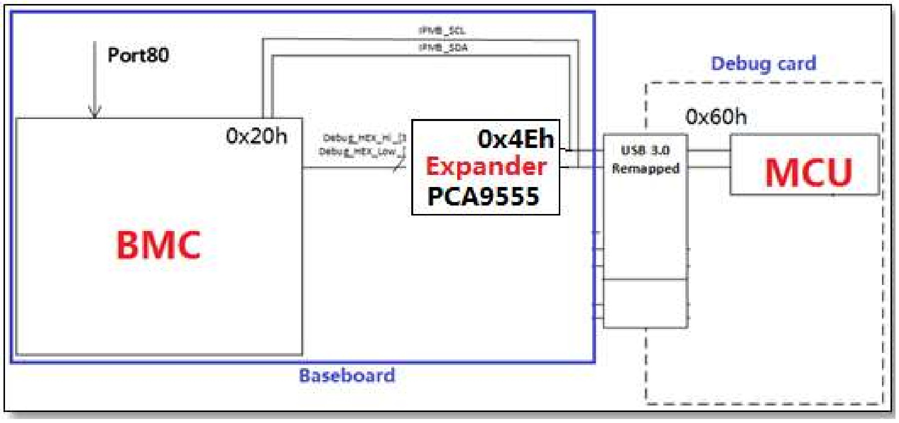
When the baseboard is designed to support both Bluetooth Debug Card V1 and the debug card with LCD, besides the usage of the present pin to disconnect the I2C/UART, the baseboard design should also consider to prioritize the UART paths to/from both debug cards in case of any confliction. A reference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



6. FW design and Interface

6.1. Roles and communication

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



6.1.1. 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.*

6.1.2. 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.
(5) Periodically(every 5s) check with BMC on availability of updated frames and update content.
(6) MCU will retry 5 times if BMC has no response to the query. After 5 times, it will time out and MCU will have System Info/Critical SEL/Critical Sensor Frames to display “BMC disconnected” under the frame title.

6.1.3. GPIO-I2C Expander (0x4Eh)

(1) Get Post Code from BMC (via GPIOs).
(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.

6.2. Communication process

6.2.1. 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.

6.2.2. 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.

6.3. 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 cycle through these pages by using up and down buttons.

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

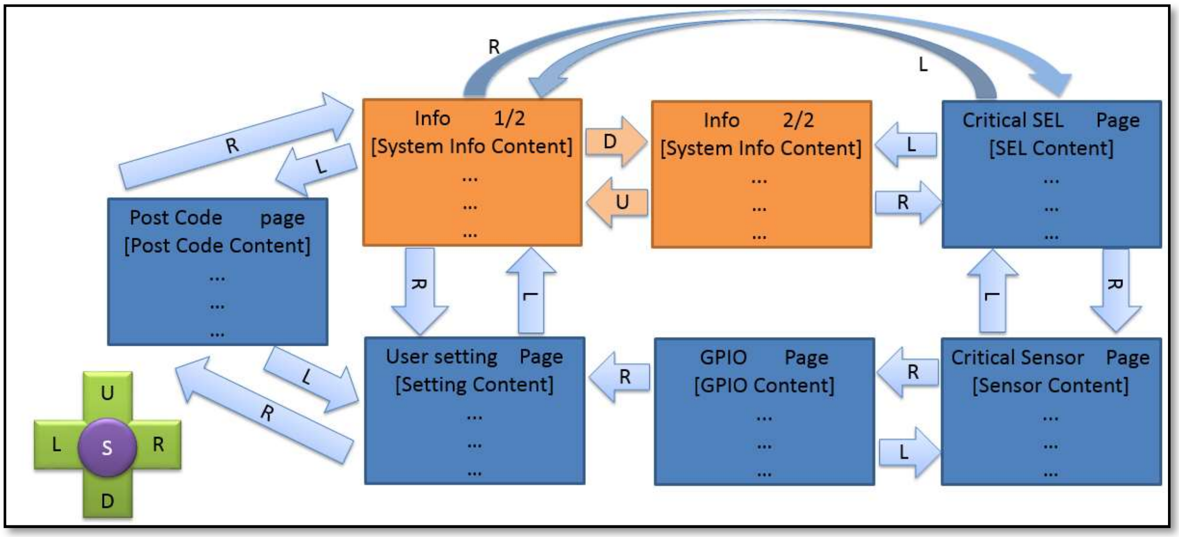


Figure 3 Pre-defined frames

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

6.3.1. Left or Right

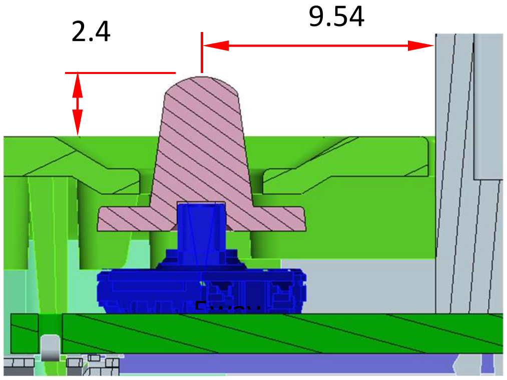
The left or right function uses to rotate different frame information to be shown on LED panel: POST code -> System Info -> BMC Critical SEL -> Critical sensors -> GPIO status -> User Settings.

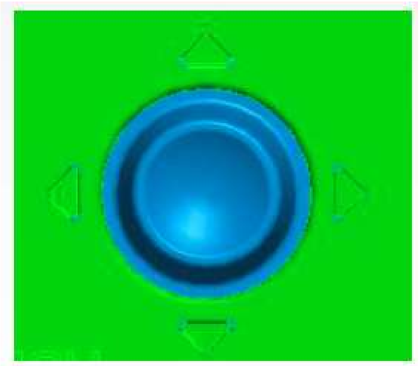
6.3.2. Up or Down

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

6.3.3. 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

6.3.4. 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.

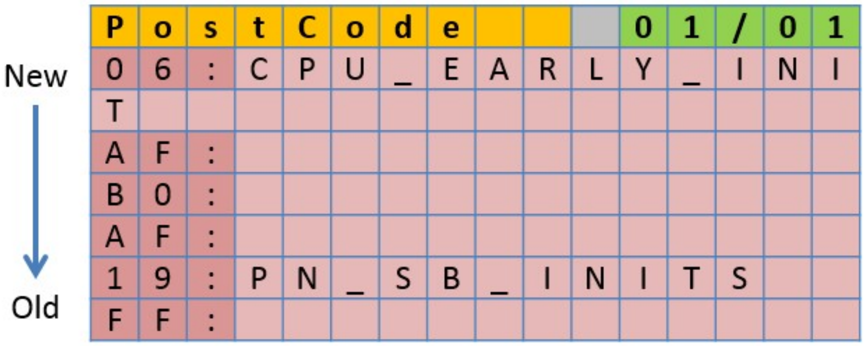
User Setting frame is the only frame that MCU will get the content from BMC to see which user settings that will be open to user to change.

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.

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.

Post Code



P	o	s	t	C	o	d	e				0	1	/	0	1
0	6	:	C	P	U	_	E	A	R	L	Y	_	I	N	I
T															
A	F	:													
B	0	:													
A	F	:													
1	9	:	P	N	_	S	B	_	I	N	I	T	S		
F	F	:													

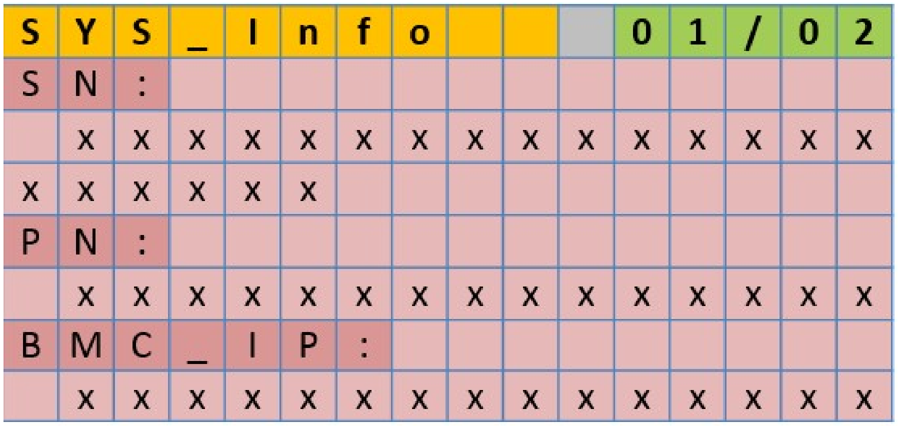
Some post code may have different description during different post phases. When debug card MCU gets post code description from BMC, byte4 will indicate the post code is for which phase. Pls refer to the command sectin for the details. MCU will based on the phase information to decide which description 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.

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 :
    - CPU: Type/Frequency/Cores
    - MEM: Vendor/Frequency/Total memory Capacity
    - HDD: Vendor/Model number
- Battery charging status and percentage
- MCU boot loader version
- MCU FW version



S	Y	S	_	I	n	f	o				0	1	/	0	2
S	N	:													
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x										
P	N	:													
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
B	M	C	_	I	P	:									
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

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:

Name	Messages	
P0 Temp UCR	P0 Temp UCR XXC – Assert/Deassert	
P1 Temp UCR	P1 Temp UCR XXC – Assert/Deassert	
Memory loop code	DIMM XX initial fails	
CATERR/IERR	CPUx IERR	Ex: CPU0/1 IERR/CATERR or MSMI
Machine Check Error	MACHINE_CHK_ERR,Uncorrectable/Correctable bank Number XX	XX: total 60 Machine Check Bank Register Table
Other IIO Error	CPU X, Error ID 0xYY – Source	X:0/1, YY: Refer to Skylake EDS, Source: IRP0, IRP1, IIO-Core, VT-d, Intel Quick Data, Misc, Reserved Ex: IIO_ERR CPU 0, Error ID 0x41 – Reserved
PROCHOT#	CPU FPH by XXX Assert/Deassert	Trigger source: UV, OC, timer exp, pmbus alert
Memory ECC/UECC	DIMMxx ECC/UECC err	
Fan Fail	Fan0/1 fail	
PCIe ECC error	XXXX PCIe err	Indicate which PCIe slot number
Power Fail	XXXX power rail fails	Indicate which power rail

AC lost	AC lost	
Sys Fan	Fan0/1 UCR xxRPM Assert/Deassert	Report the fan speed when SEL logged
CPU0/1 thermtrip	CPU0/1 thermtrip Assert/Deassert	
Voltage low/high critical	PXXV lower/upper critical	Report the voltage when SEL logged, Ex: P3V_BAT UCR XXV – Assert/Deassert

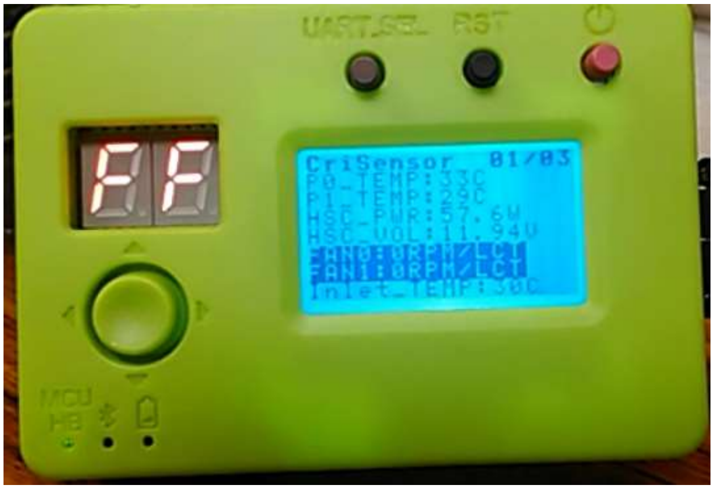
Critical Sensor Frame

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

Sensor Name	Messages	
CPU0 temp	P0_TEMP:XXC	show “P0_temp XXC/UCT” If out of range and invert the font
CPU1 temp	P1_TEMP:XXC	show “P1_temp XXC/UCT” If out of range and invert the font
HSC Power	HSC_PWR:XXX.XW	Show “HSC Pwr XXX.Xw/UCT” If out of range and invert the font
HSC voltage	HSC_VOL:XX.XXV	show “XX.XXV/LCT” or “XX.XXV/UCT” If out of range and invert the font
Fan0/1 speed	Fan0/1:XXXXRPM	Show ”XXXXRPM/UCT” or “XXXXRPM/LCT” if out of range and invert the font
Inlet Temp	Intel_TEMP:XXC	Show ”XXC/UCT” if out of range and invert the font
CPU0/1 VR temp	P0/P1_VR_TEMP:XXC	Show “XXC/UCT” if out of range and invert the font
CPU0/1 VR PIN	P0/P1_VR_Pwr:XX.XW	Show “XXW/UCT” if out of range and invert the font
DIMM Temp	DIMMXX_TEMP:XXC	Show ”XXC/UCT” if out of range and invert the font Base on platform sensors it will show a group of

		DIMM sensors, eg: DIMMA_Temp: XXC DIMMB_Temp: XXC

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:



```
CriSensor    01/03
P0_TEMP:33C
P1_TEMP:29C
HSC_PWR:57.6W
HSC_VOL:11.94V
FAN0:0RPM/LCT
FAN1:0RPM/LCT
Inlet_TEMP:30C
```

GPIO Status Frame

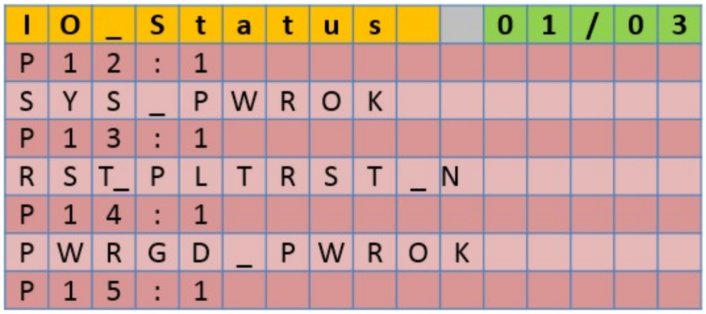
The content of this frame contains all the GPIO signal information with user readable GPIO signal name and its status i.e. 'CPU_CARERR_MSMI_LVT3_N'. Since GPIO signals are specific to a given platform, MCU sends IPMB request to BMC to get the look up table that shows the association of GPIO signal and 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:

Bit	Usage	Messages	Direction (In the perspective of PCA9555)
IO1_0	RST_BTN_N	IO1_0: FM_DBG_RST_BTN	Output
IO1_1	PWR_BTN_N	IO1_1: FM_PWR_BTN	Output
IO1_2	PWRGD_SYS_PWROK	IO1_2: SYS_PWROK	Input
IO1_3	RST_PLTRST_N	IO1_3: RST_PLTRST	Input
IO1_4	PWRGD_DSW_PWROK	IO1_4: DSW_PWROK	Input
IO1_5	FM_CPU_CATERR_MSMI_LVT3_N	IO1_5: FM_CATERR_MSMI	Input

IO1_6	FM_SLPS3_N	IO1_6: FM_SLPS3_N	Input
IO1_7	FM_SOL_UART_CH_SEL	IO1_7: FM_UART_SWITC	Output

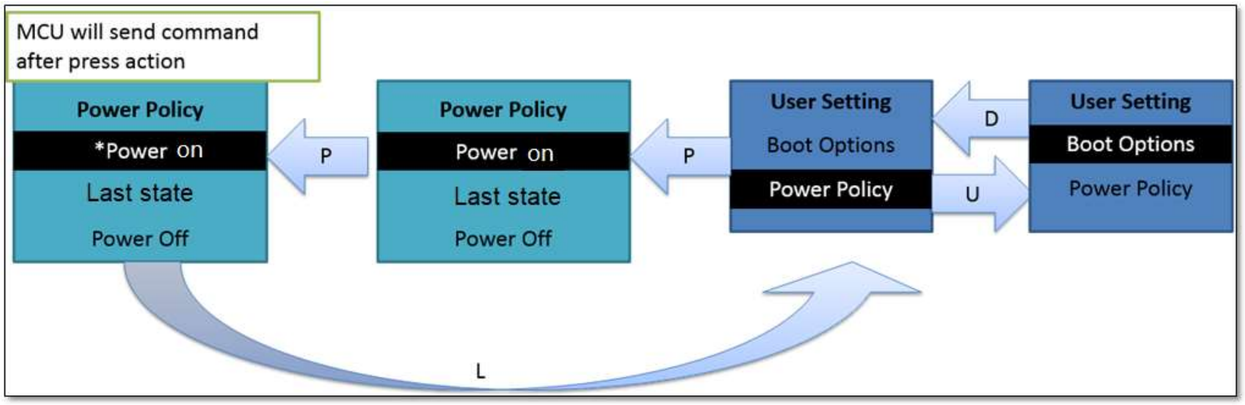
GPIO



I	O	_	S	t	a	t	u	s			0	1	/	0	3
P	1	2	:	1											
S	Y	S	_	P	W	R	O	K							
P	1	3	:	1											
R	S	T_	P	L	T	R	S	T	_	N					
P	1	4	:	1											
P	W	R	G	D	_	P	W	R	O	K					
P	1	5	:	1											

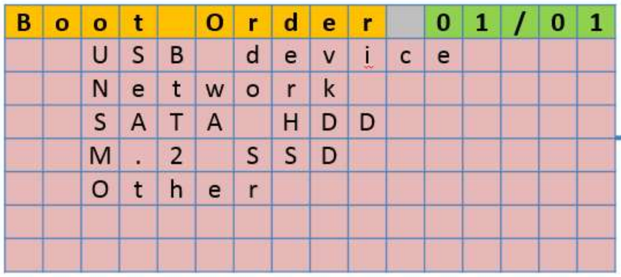
User Setting Frame

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



MCU will send command after press action

Boot Order:



B	o	o	t		O	r	d	e	r		0	1	/	0	1
		U	S	B		d	e	v	i	c	e				
		N	e	t	w	o	r	k							
		S	A	T	A		H	D	D						
		M	.	2		S	S	D							
		O	t	h	e	r									

MCU work step for boot order

1. Get currently boot order.
2. After selecting the boot option, user can use “Up””Down”option to fine tune the sequence.MCU will send the set boot order command to BMC

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.

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.

6.3.5. 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:

Escape sequence	description	example
LCD FW Non-CSI codes Start with the characters ESC the final byte is technically any character in the range 64–95 (hex 0x40–0x5F, ASCII @ to ~)		
ESC 'B'	Battery power percentage; overwrite the following data	
ESC 'U'	MCU Bootloader version; overwrite the following data	
ESC 'R'	MCU Runtime firmware version; overwrite the following data	
CSI codes - reference: https://en.wikipedia.org/wiki/ANSI_escape_code 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 ~)		
ESC '[' **n** 'm'	**n** can be zero or more SGR parameters separated with ';'. With no parameters, **n** is treated as 0 (Reset)	**ESC [5 ; 7 m** represent the following data are Blink and Reversed; **ESC [m** represent the following data are Reset to normal

SGR (Select Graphic Rendition) parameters	
0	Reset / Normal

5	Blink
7	Reverse; swap foreground and background

6.4. 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:

6.4.1. Request

rsAddr	NetFn / rsLun	Header checksum	rqAddr	rqSeq	cmd	Request data bytes (0 or more)	Data checksum

6.4.2. Response

rqAddr	NetFn / rqLun	Header checksum	rsAddr	rqSeq	cmd	Completion code	Response data bytes (0 or more)	Data checksum

Net Function = 3Ch LUN = 00b			
Code	**Command**	**Request, Response Data**	**Description**
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 – 00h : no update	

Net Function = 3Ch LUN = 00b			
		01h: the total number of updated frames Byte 5:N – updated frame number(s)	
03h	Get POST code description	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.
04h	Get GPIO expander IOs description	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 0: low level active	MCU to get GPIO expander IO pins definition and description

Net Function = 3Ch LUN = 00b			
		1: high level active Byte 7 – function pin define 00h: input pin only 01h: Power button 02h: Reset button 03h: UART switch button 04h~FFh: reserved Byte 8 – length (n) Byte 9:(n+ 4) - human readable string (ASCII format)	
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 Contorl 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 FFh: no next page need to be updated	

Net Function = 3Ch LUN = 00b			
		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 o6h:

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)

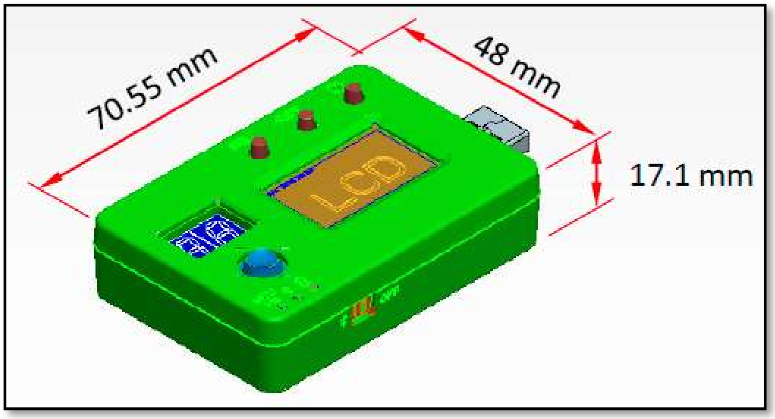
Req: Panel 1 Operation 0 Item 1 Res: Panel 1 item 1 “Power Policy”

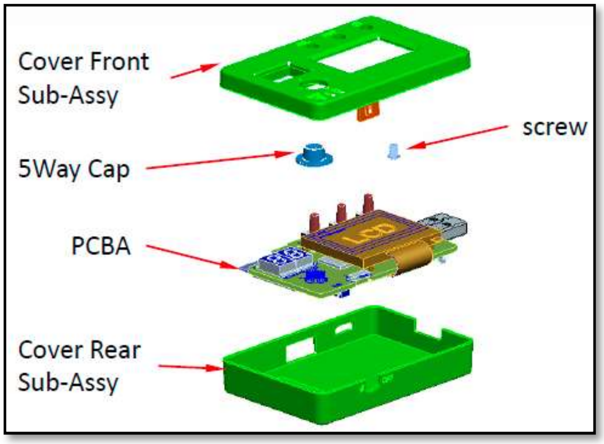
Req: Panel 1 Operation 0 Item 2 Res: Panel 1 item 2 “Boot Sequence”

Req: Panel 1 Operation 0 Item 3 Res: No more items (Completion Code: C9h)

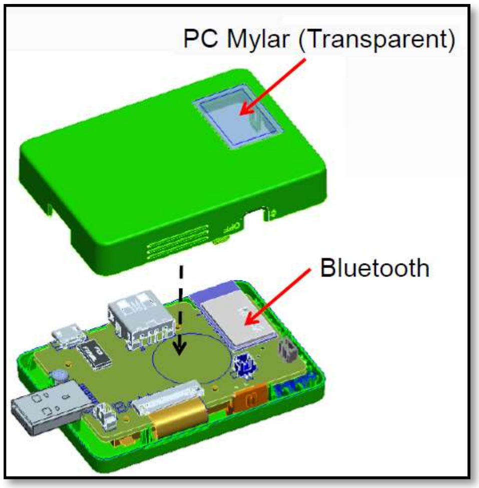
7. Debug Card ME design

Dimension

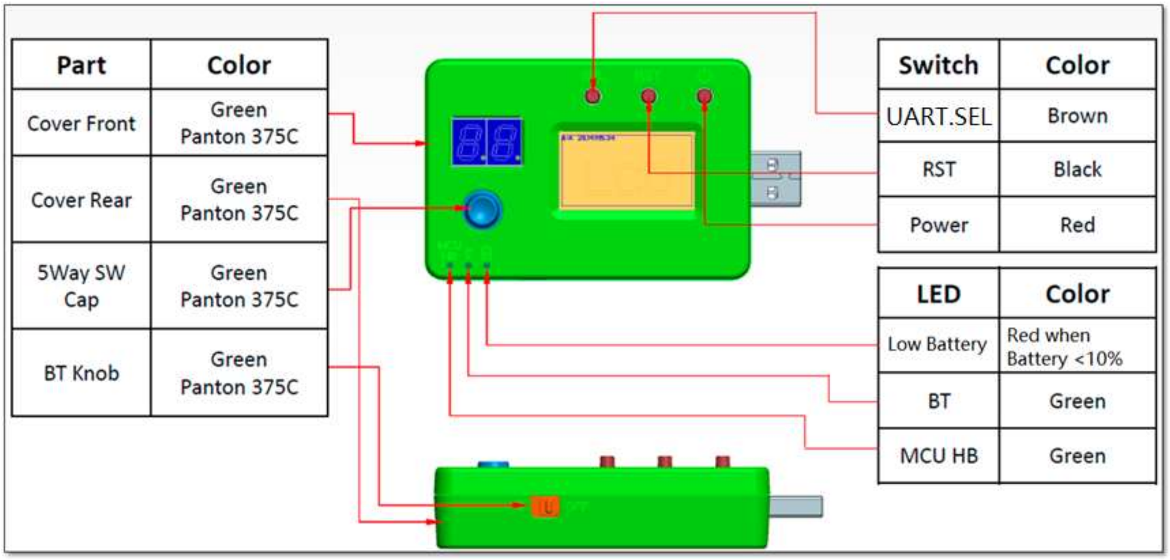




Back side design: for bluetooth area, there should be a transparent area to let end user to check the MAC for the bluetooth module.



Color proposal

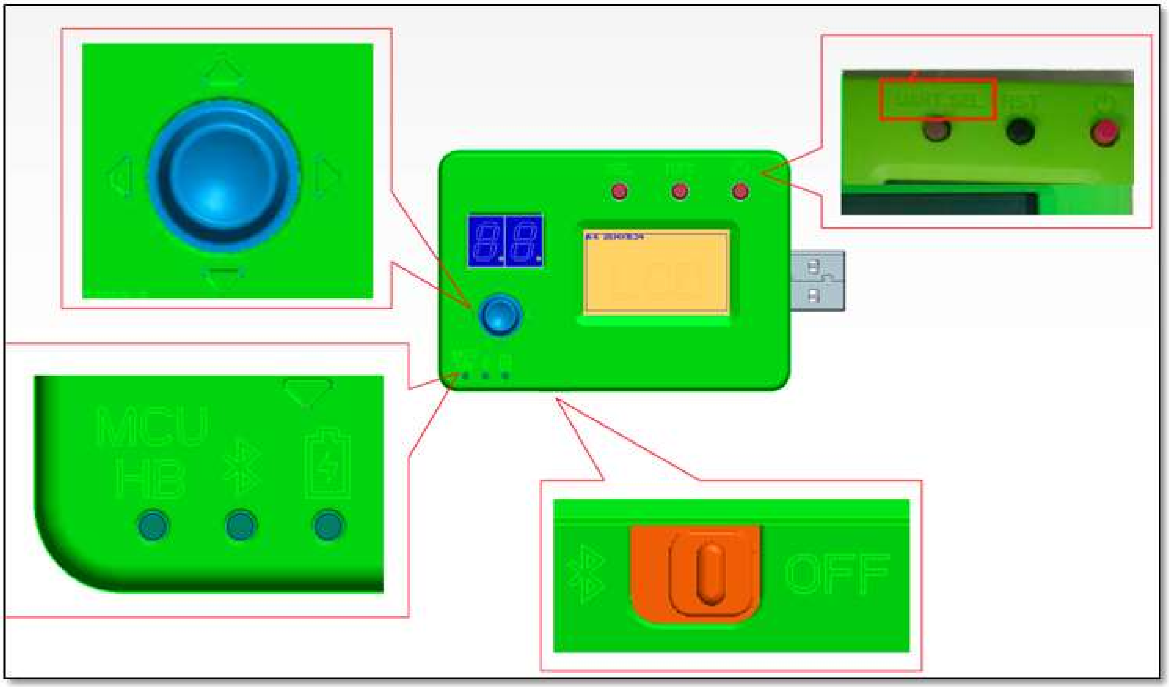


Part	Color
Cover Front	Green Panton 375C
Cover Rear	Green Panton 375C
5Way SW Cap	Green Panton 375C
BT Knob	Green Panton 375C

Switch	Color
UART.SEL	Brown
RST	Black
Power	Red

LED	Color
Low Battery	Red when Battery <10%
BT	Green
MCU HB	Green

Art work


UART SEL
RST
LCD
MCU HB
OFF

8. Appendix

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