**IISC EV Charger intergation with RB-i.MX6UL**

Version: V1.0

Customer: IISC

**Project Co-ordination Team**

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | Name | Email-ID | Company |
| Prepared By | G V SAIKRISHNA | Sai.k@phytec.in | PHYTEC |
| Reviewed By |  |  |  |

**Table of Contents**

[1. Overview 3](#__RefHeading___Toc676_2794178309)

[2. Scope of Work 3](#__RefHeading___Toc678_2794178309)

[3. Hardware Block diagram 3](#__RefHeading___Toc680_2794178309)

[4. Software Architecture 4](#__RefHeading___Toc682_2794178309)

[4](#__RefHeading___Toc684_2794178309)

[4.1 Parameter list on Display 4](#__RefHeading___Toc450_1563508107)

[4.2 CAN frame format and structure 5](#__RefHeading___Toc452_1563508107)

[4.2.1 CAN frame from RB-i.MX6UL to TMS controller: 5](#__RefHeading___Toc454_1563508107)

[4.2.2 CAN frame from TMS controller to RB-i.MX6UL: 5](#__RefHeading___Toc456_1563508107)

[4.3 Cloud connectivity details 6](#__RefHeading___Toc458_1563508107)

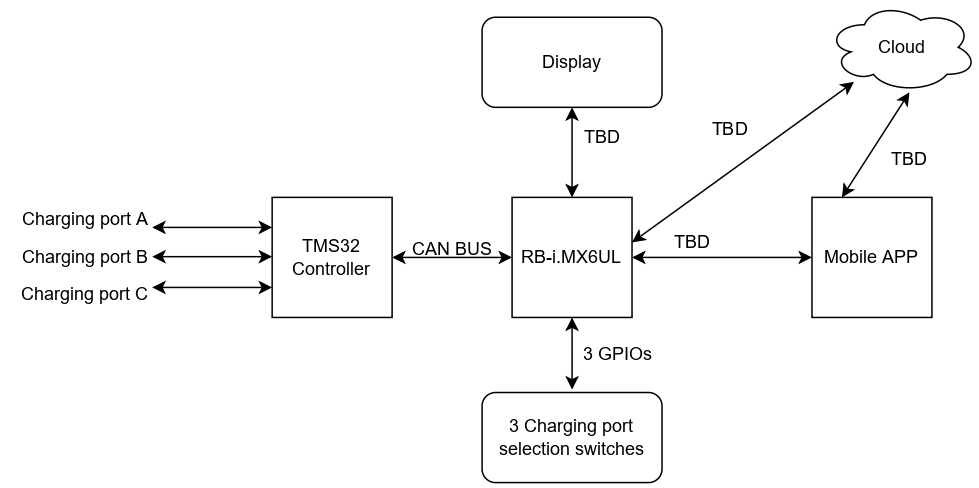
# **1. Overview**

This document defines the integration of RB-i.MX6UL with EV chargins station(TMS controller Board) and the cloud connectivity between RB-i.MX6UL and AWS cloud or simple Mobile APP. The charging station has 3 ports where the vehicles are to be charged

# **2. Scope of Work**

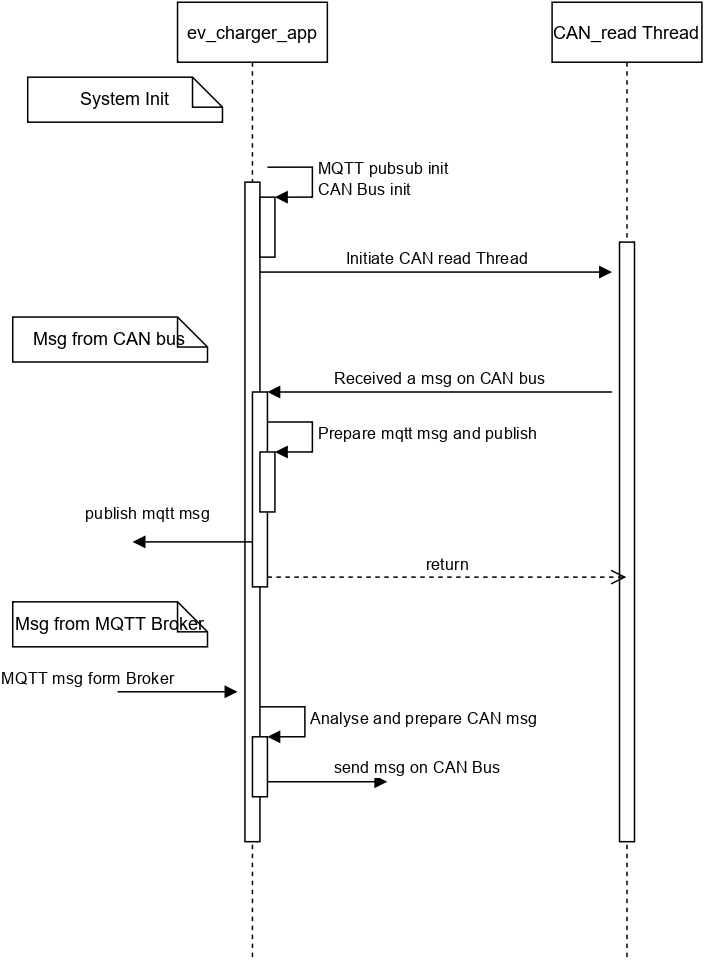
* TMS Controller is the EV charging station consisting of 3 ports for charging the vehicles at different voltage levels(typically 48V, 60V, 72V)
* The same controller system is connected to RB-i.MX6UL over a CAN Bus for exchanging the battery related and power grid related parameters(Voltage, current, charging port ON/OFF etc)
* The same parameters are to be shown on the Display of RB-i.MX6UL and sent to cloud and mobile app also over MQTT(TBD)
* RB-i.MX6UL shall have a provision to select the charging ports from the hardware swithes as well rather than the cloud or Mobile APP.
* A minimal and custom mobile app for sending and receiving the charging port information/commands.

# **3. Hardware Block diagram**



The inter connections among the sub units of the system is as shown above.

# **4. Software Architecture**

****

# 

## 4.1 Parameter list on Display

The below paramters are to be displayed on the LCD display.

1. Battery voltage and current values from the 3 ports
2. Power grid voltage and current values
3. ON/OFF buttons for selecting the charging ports and input box to provide the respective voltage rating of the battery connected to particular port.
4. Incase of any error while inputing the data, a pop-up window should be open with the error details
5. Incase of battery full condition of the respective port, OFF button should be turn to green to indicate battery is full. While in charing mode, the OFF button should be grey which indicates battery is in charging mode.

## 4.2 CAN frame format and structure

The CAN messages format and structure between the RB-i.MX6UL and TMS controller are as follows.

### 4.2.1 CAN frame from RB-i.MX6UL to TMS controller:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |

**B0 –** Will specify whether the given CAN frame is a command or command\_reply or a status frame

Value of the B0 byte is

0x10 – Command

0x60 – Command\_reply (Adding 0x50 to the command value)

0x01 – Status frame

**B1 –** will specity the port numbers and its state (ON/OFF)

Lower Nibble of B1 – Port selection (PA/PB/PC)

0x01 – PA

0x02 – PB

0x03 – PC

Higher Nibble of B1 – Port State (ON/OFF)

0x01 – ON

0x00 – OFF

**B2 –** will specify the integral part of port(PA/PB/PC) voltage

**B3 –** will specify the fractional part of port(PA/PB/PC) voltage

B4, B5, B6 and B7 are dont care conditions.

The below CAN frame is the example for sending a command to turn ON port ‘A’ with 49.36V

CAN frame: 7E0#10113124 where 7E0 is the CAN id

### 4.2.2 CAN frame from TMS controller to RB-i.MX6UL:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |

**B0 –** Will specify whether the given CAN frame is a command or command\_reply or a status frame

Value of the B0 byte is

0x10 – Command

0x60 – Command\_reply (Adding 0x50 to the command value)

0x01 – Status frame

**B1 –** will specity the port numbers and its state (ON/OFF)

Lower Nibble of B1 – Port selection (PA/PB/PC)

0x01 – PA

0x02 – PB

0x03 – PC

Higher Nibble of B1 – Port State (ON/OFF)

0x01 – ON

0x00 – OFF

**B2 –** will specify the integral part of port(PA/PB/PC) voltage

**B3 –** will specify the fractional part of port(PA/PB/PC) voltage

**B4 –** will specify the integral part of port(PA/PB/PC) current

**B5 –** will specify the fractional part of port(PA/PB/PC) current

B6 and B7 are dont care conditions.

The below CAN frame is the example for sending a command reply(with V = 13.19V and I = 8.18A) for turn ON port ‘A’ with 49.36V command

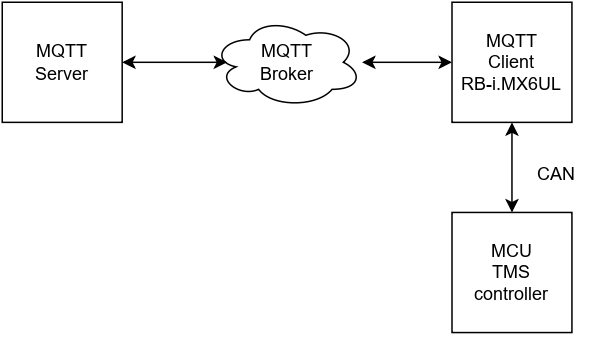
CAN frame: 7E1#60110D130812 where 7E1 is the CAN id of TMS controller

For status messages, the controller will send the port voltage and current details at a regular intervals (‘X’ secs). This status messages and command\_reply messages will be alomost same except the **B0** Byte value.

## 4.3 Cloud connectivity details

As of now, cloud communication is being done with a MQTT online free broker for initial prototype.

The block diagram for the cloud communication is as belows



**MQTT server side:**

subscribing topic: device\_to\_cloud

publishing topic: cloud\_to\_device

MQTT message: <Port> <state> <voltage>

port = PA/PB/PC

state: ON/OFF

voltage: float value

broker ip: mqtt.eclipseprojects.io:1883

**MQTT Client side:**

subscribing topic: cloud\_to\_device

publishing topic: device\_to\_cloud

MQTT message: <Port> <state> <voltage>

port = PA/PB/PC

state: ON/OFF

voltage: float value

broker ip: mqtt.eclipseprojects.io:1883