maxAI 280 Design Studio Software Guide

Revision History

Ver	Change Description	Date	Author	Approver
1.0	Initial release	Sep 28, 2022	Zaid N	
2.0	Changes made in LED Section 7.4, 7.4.6, 7.4.7, 7.4.9, 7.4.10 Digital IO 7.2.2, 7.2.3, 8.2.4.1 Keypad 7.1.6, 7.1.3.3 In each section of the platform service regarding the time out configuration and task priority, added the text "Recommended value" Common sections included from AI430 user manual into the AI280 user manual.	Oct 17, 2022	Zaid N	
3.0	Overall corrections. See attached excel file for details: maxAI280 SDK Manual Corrections	Oct 21st, 2022	Luis Figueroa	Victor Rios

References

No	Document Name	Ver.	Lotion
01	touchgfx-documentation-4.16.pdf	1.0	https://support.touchgfx.com/4.20/docs/introduction/welcome
02	STM 32 document	1.0	http://www.st.com/en/microcontrollers-microprecessors/stm32-32-bit-arm-cortex-mcus.html
03	Free RTOS	1.0	https://controllerstech.com/free-rtos-tutorial-2-0-with-stm32/
04	J1939	1.0	https://copperhilltech.com/a-brief-introduction-to-the-sae-j1939-protocol/

Table of Contents

Purpose of document

The purpose of this document is to enable an application developer to write TouchGFX applications for the maxAI280 SDK using the features, modules, interfaces, and possible configurations that is available with the maxAI280 SDK hardware platform.

Scope of document

The scope of the document is to list all the features and functionalities of the maxAI280 SDK which are of relevance to the Touch GFX Application Developer using the SDK.

Acronyms & Abbreviations

No	Acronyms	Definition
01	SDK	Software development kit
02	IDE	Integrated development Environment
03	DL	Data Layer
05	RTC	Real Time Clock
06	BLE	Bluetooth low energy
08	DB	Data Base
09	API	Application Programming Interface
10	LCD	Liquid crystal Diode
11	CAN	Control area network

1 Maximatecc Software Overview

1.1.1 maxAI™ Configurator

For quick and easy setup, use the Configurator Tool to automatically populate your engine monitoring data with preset options and layouts. No need for complex coding or additional resources.

1.1.2 maxAI™ Design Studio

The Design Studio is a Software Development Kit (SDK) that provides a higher level of flexibility and control. You choose the advanced engine monitoring parameters to equip your display with all the information you need to know.

1.1.3 maxAI™ Specialized

The Specialized provides you with access to the maximatecc engineering team, who develop a custom interface that meets your specific application needs. The team supports all elements of the engineering and setup process for ease and flexibility.

2 Introduction

The AI280 SDK platform is an embedded software solution for custom applications based on the AI280 hardware only. This platform provides a set of software components to reduce the development effort to create a complete embedded application compliant with all the customer requirements. The SDK solution potentiates the scope of the AI280 platform.

The user can explore all the possibilities to cover the requirements and needs by using the AI280 peripherals and by creating their own custom graphical applications.

The SDK platform has the following benefits:

➢ Short development time.
➢ Pre-established low level driver administration.
➢ Portable software components.
➢ Low technical development skills required.
➢ Pre-configured and stable SW architecture.
➢ Secure custom and private algorithms implementation.

3 SDK Setup and installation

To get started with the AI280 SDK, you will need to setup the right environment. Please follow the procedure described in this section to install the necessary tools required to use the SDK and create a Touch GFX application.

3.1 SDK Development Environment

The AI280 SDK allows TouchGFX Applications to be custom built on the AI280 platform. Please ensure the below hardware and software setup is available.

3.1.1 Hardware Requirements

Host PC	WINDOWS (64-bit OS)
RAM Size	4 GB RAM required minimum
Disk Space	2 GB disk space required minimum
Board with power supply	MAXAI280 kit
Debugger	ST Link V2 in-Circuit debugger with USB cable

3.1.2 Software Requirements

Development IDE	STM32 Cube (1.8.0)
Development IDE	TOUCH GFX (4.18.1)
SW Package	S/W package released with the MAXAI280 kit

3.2 IDE Installations

To get started with the AI280 SDK, please follow the below links to install the STM32 Cube IDE and the Touch GFX IDE.

1) Install the STM32 Cube IDE following the instructions in the document https://www.st.com/resource/en/user_manual/um2563-stm32cubeide-installation-guide-stmicroelectronics.pdf

2) Install the Touch GFX IDE following the instructions listed in the document https://support.touchgfx.com/4.18/docs/introduction/installation

3) Once the STM32 Cube IDE is installed to open one of the SDK projects (integration test project, demo project or blank template project) open the Cube IDE and from the File menu select *"open project from file system…"* option

For reliability within TouchGFX, please ensure the project files are located in a file path directory with no spaces. i.e.,
"C:\Users\91822\Desktop\AI280\Project_Build\AI280_Project_Integration"

To avoid conflicts only open one SDK project at the time.

File Edit Source Refactor Navigate Search Project Ru

New	Alt+Shift+N >
Open File...	
Open Projects from File System...	
Recent Files	>
Close Editor	Ctrl+W
Close All Editors	Ctrl+Shift+W
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
Revert	
Move...	
Rename...	F2
Refresh	F5
Convert Line Delimiters To	>
Print...	Ctrl+P
Import...	
Export...	
Properties	Alt+Enter
Switch Workspace	>
Restart	
Exit	
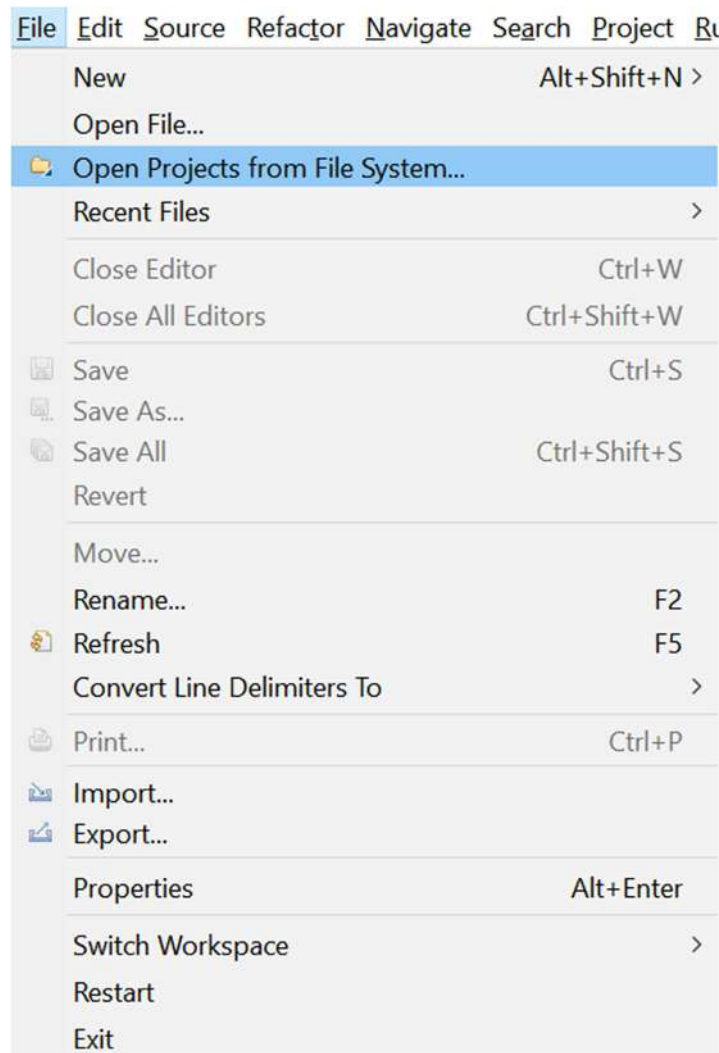
Select the option "Directory…" to look for the folder where the SDK project is located.

After some seconds the option "Finish" will be enable and need to be clicked to finish the process.
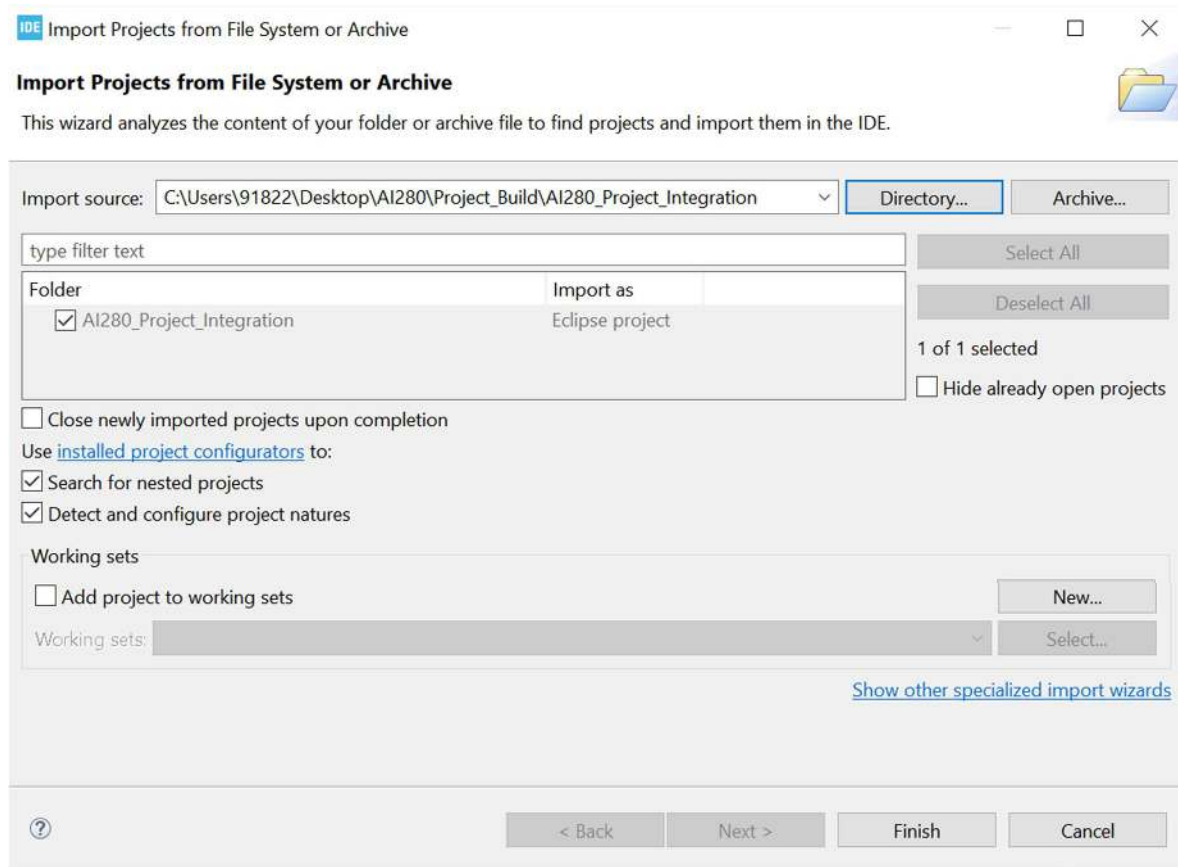
Once the process to import the project is done the files will be available in the left section of the IDE.
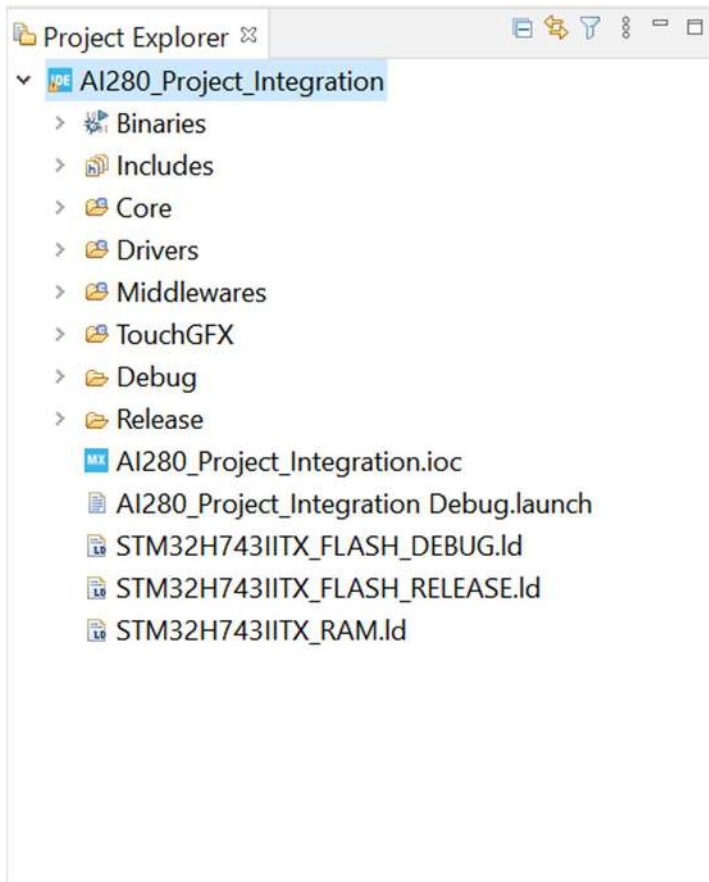
4) Once the SDK project is imported an external loader need to be copied into the installation folder.
   Inside the AI280 project in Tools folder the external loader is located.
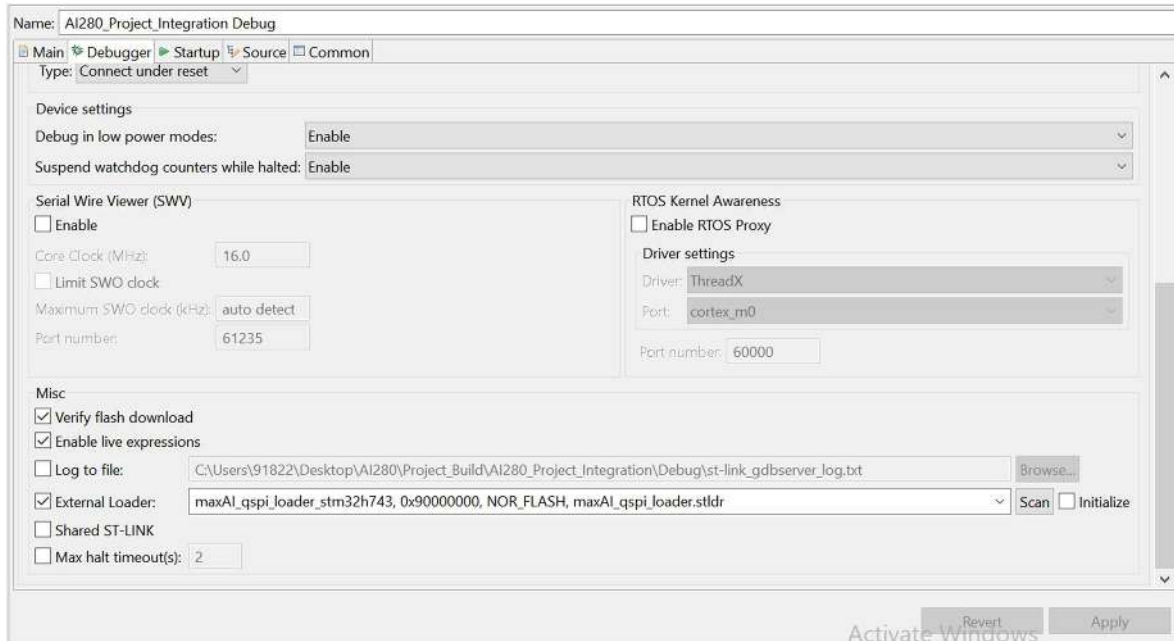
The maxAI_qspi_loader. stldr need to be copied in the following path:

C:\ST\STM32CubeIDE_1.8.0\STM32CubeIDE\plugins\com.st.stm32cube.ide.mcu.externaltools.cubeprogrammer.win32_2.0.100.202110141430\tools\bin\ExternalLoader

3.3 S/W Release Package Details

The MAXAI280 SDK kit comes with the below S/W release package. It has 3 project files released.

1) Demo Project File
2) Application Project File
3) Blank Project File

3.3.1 AI280 Project Structure

The MAXAI280 project files are integrated source code which include the TouchGFX application integrated with the AI280 SDK. These applications leverage the hardware capabilities of the AI280 platform via the SDK interface.

In this section we will describe to you the variations in the three project files released with the MAXAI280 SDK which will enable you to write full-fledged applications using the AI280 SDK.

3.3.2 Demo Project File

The Demo Project File is a fully graphical pre-built project file which leverages all the functionality of the AI280 SDK. This application is an integrated example which communicates with different modules in the SDK in a single UI screen. This project can be used as a reference for all users who are working on creating integrated applications for their specific needs. This project has 5 UI screen and the details of how-to setup and test the same are described in section 7.2. The below image shows the folder structure of the AI280 demo project file. The Main folders of interest are

    a. Core
    b. Drivers
    c. Touch GFX
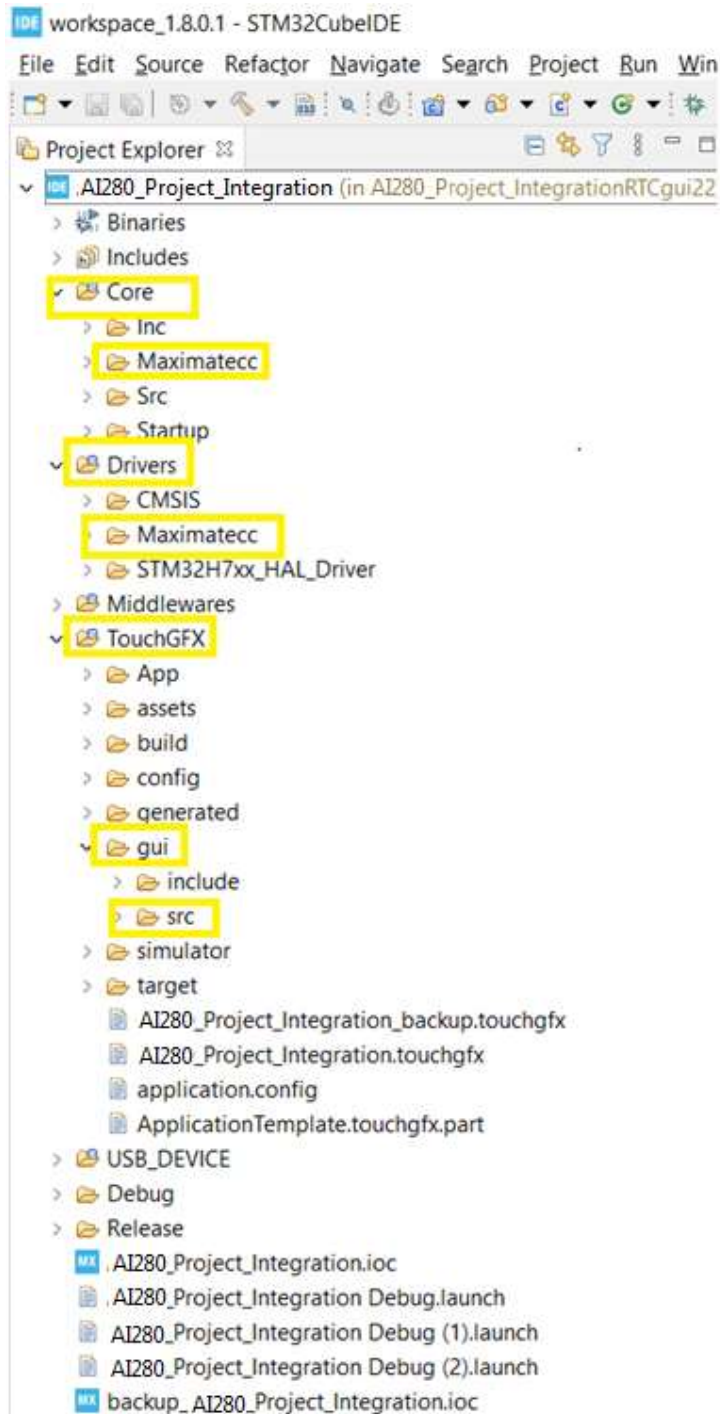
3.3.2.1 Core Directory

The core directory includes the files which form the core of the SDK architecture which include the platform service files for all the modules. (Platformservice.h and Platformservice.c) They are located under the Core\Maximatecc\Inc and Core\Maximatecc\Src directories.
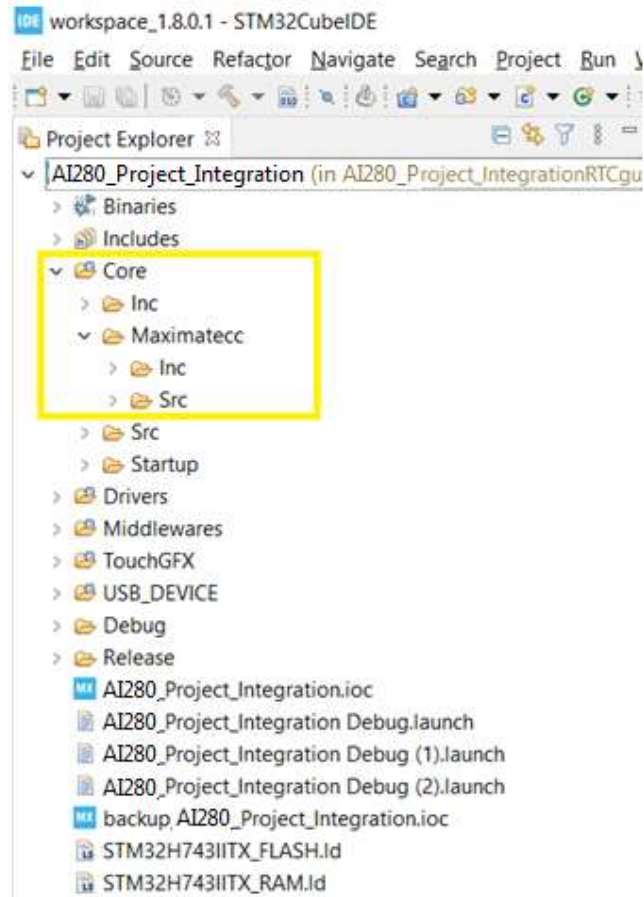
3.3.2.2 Driver Directory

The Driver files for all the modules in the MAXAI280 are located under the folder structure Drivers\Maximatecc\inc and Drivers\Maximatecc\src.
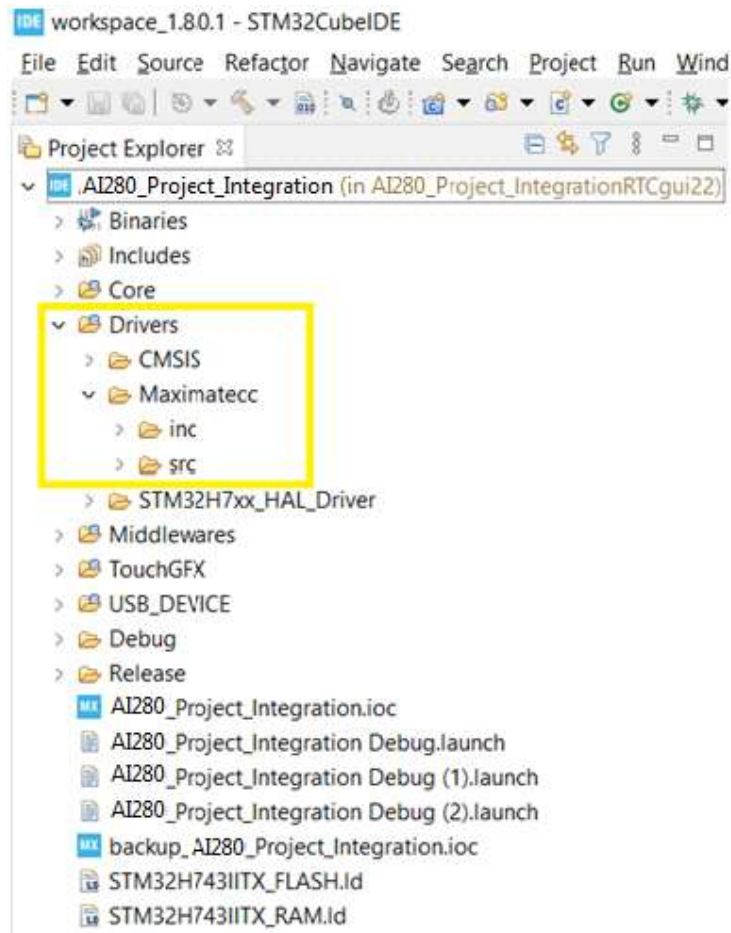
3.3.2.3   TouchGFX Directory

The GUI files for Demo purpose and understanding are available under the folder structure TouchGFX\gui\src. The screens that are available as demo are scr_io_screen,scr_main_screen and scr_timer_screen.

As a TouchGFX developer any GUI code that is developed by you would go into this directory.
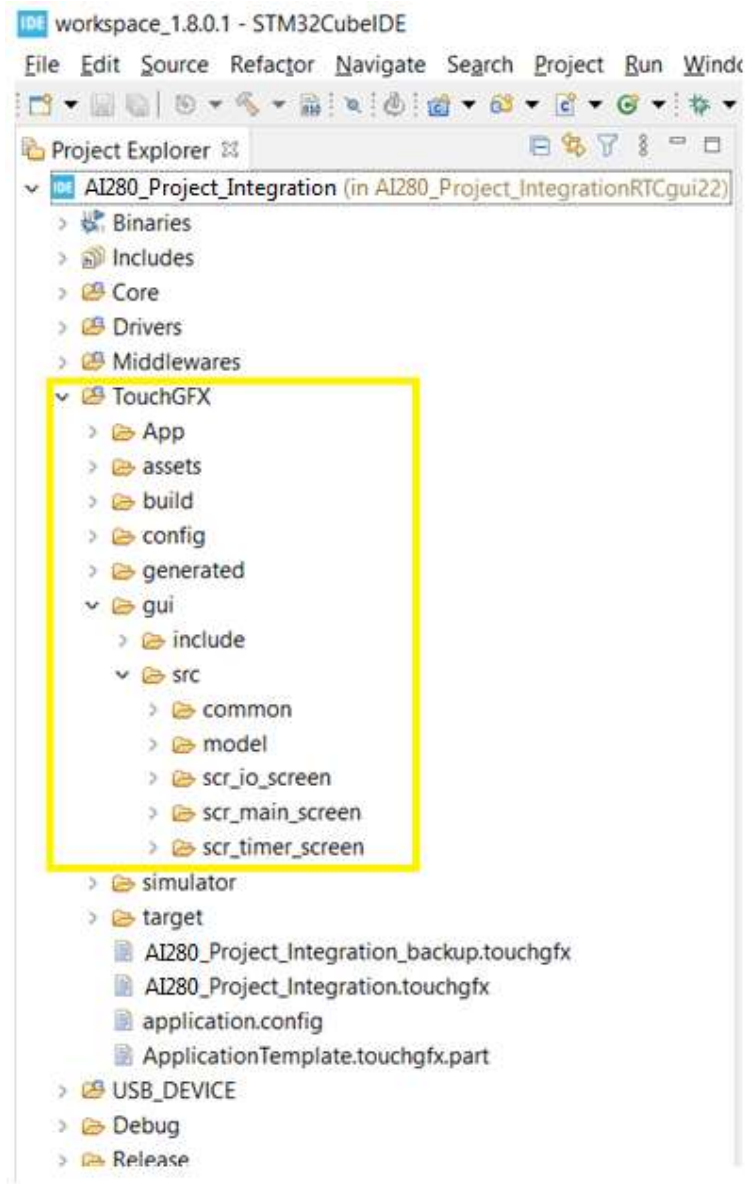
Dept: Software	User Manual Document	Rev No.	Rev 3.0
		Date:	Oct 17, 2022

3.3.3 Sample Application Project File

The Application Project File is a semi graphical pre-built project file which details each module which is available in the AI280 hardware. References for all the functionality of the AI280 SDK can be found in the application project file. This sample application has standalone screens for each and every module in the SDK and elaborates in detail the possible ways you can interact with each module in the SDK. This project can be used as a reference for all users to understand in detail the individual modules of the SDK and get sample reference of how to use the various functionalities in the individual modules. The below image shows the folder structure of the AI280 sample application project file. The Main folders of interest are

  a. Core: Similar to the Demo project
  b. Drivers: Similar to the demo project.
  c. Touch GFX
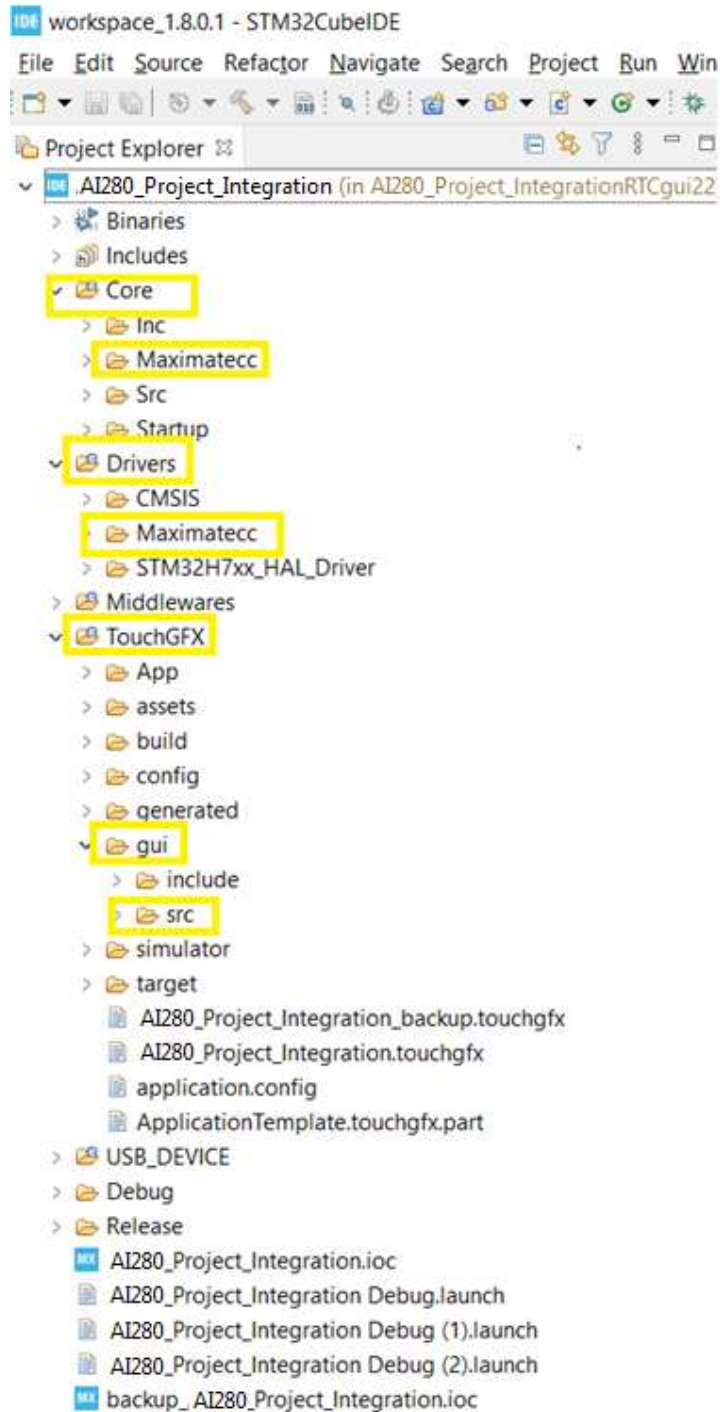
3.3.3.1 TouchGFX Directory

The GUI files for application purpose and understanding are available under the folder structure TouchGFX\gui\src. The screens for all the possible user applications are available under mentioned folder as shown in the below diagram.
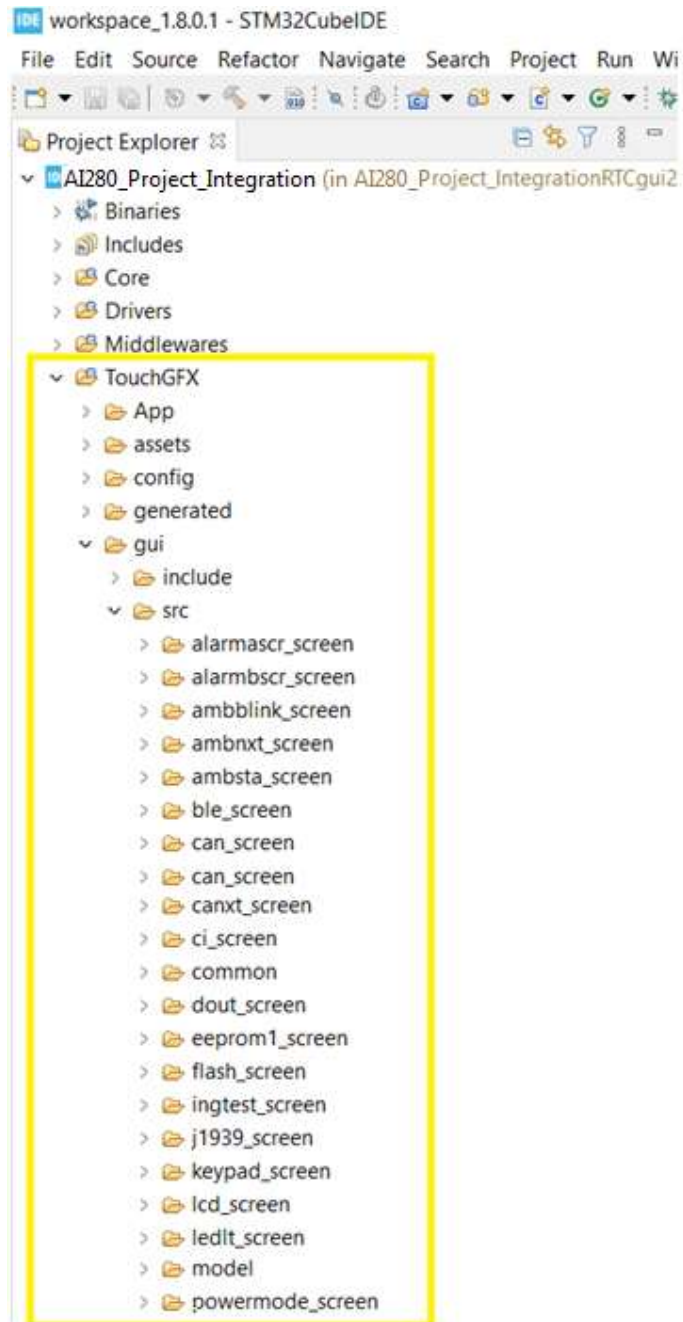
3.3.4   Blank Project File

The Blank Project File template is provided as a convenience for the end user to begin the firmware development. The Blank Project file can be unzipped to the desired location (folder) and renamed to a name as per users' choice.

3.3.4.1   TouchGFX Directory

The GUI files for creating new applications should be added under the folder structure TouchGFX\gui\src. Section 3 describes in detail how a sample application can be written and integrated with the SDK and tested on a MAX AI280 board.
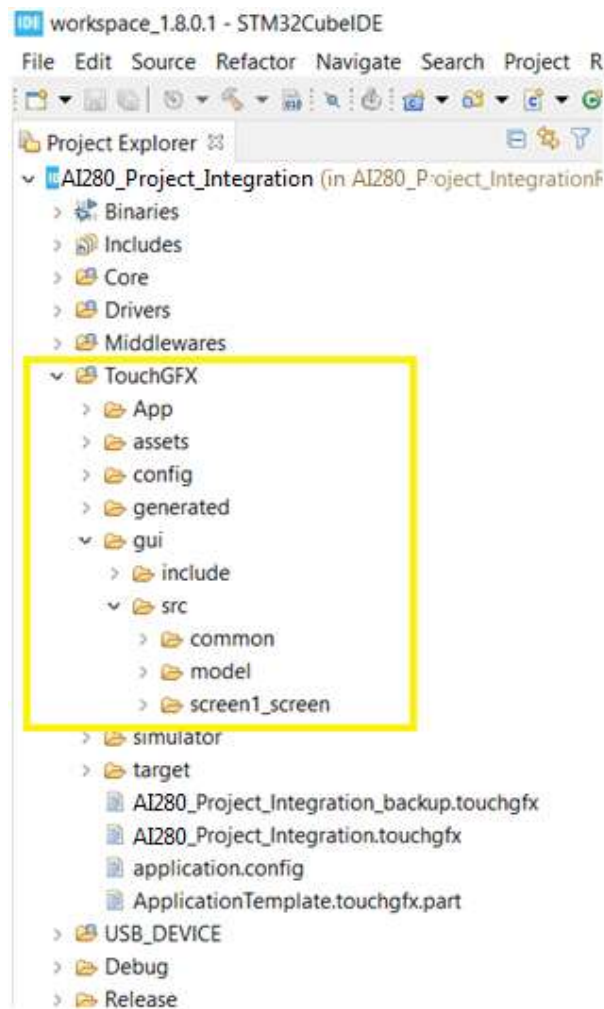
3.3.4.2 Blank User Task Files

The SDK has included some blank user tasks that can be used by the application developers if they would like to create some tasks that run in the platform independent of the Touch GFX.As shown in the image below, you can find the user tasks in the path core/src/user_task.c as shown below.
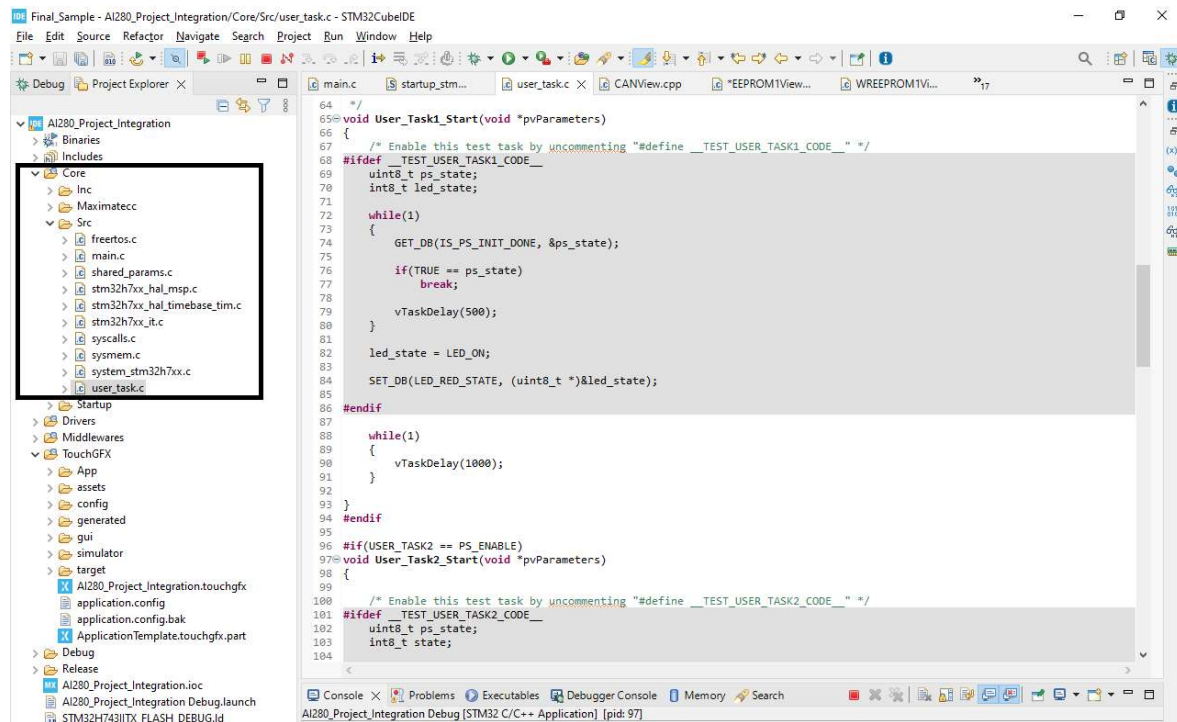
Please refer to section 3.5 to understand how to edit these user tasks and integrate with the SDK and tested on a MAX AI280 board.

4   SDK Application Development Procedure

In this section we will walk you through the procedure to write a simple Touch GFX application using the blank project file provided in the S/W release package and compile the same with the STM Cube IDE and flash it on the MAXAI280 hardware and test it. We will also provide you the details on how you can debug the application using the ST link debugger.

4.1   Blank Project Installation and loading

The MaxAI280 SDK release comes with 3 project files as described in the Section 2.3. Please go to the folder with the zip file (AI280_GettingStart.zip) containing the blank project file is available and unzip the same. You will get the directory, AI280_Project_IntegrationRTCgui22 after unzip the file. The below image shows the contents and path of the blank project after it has been unzipped.
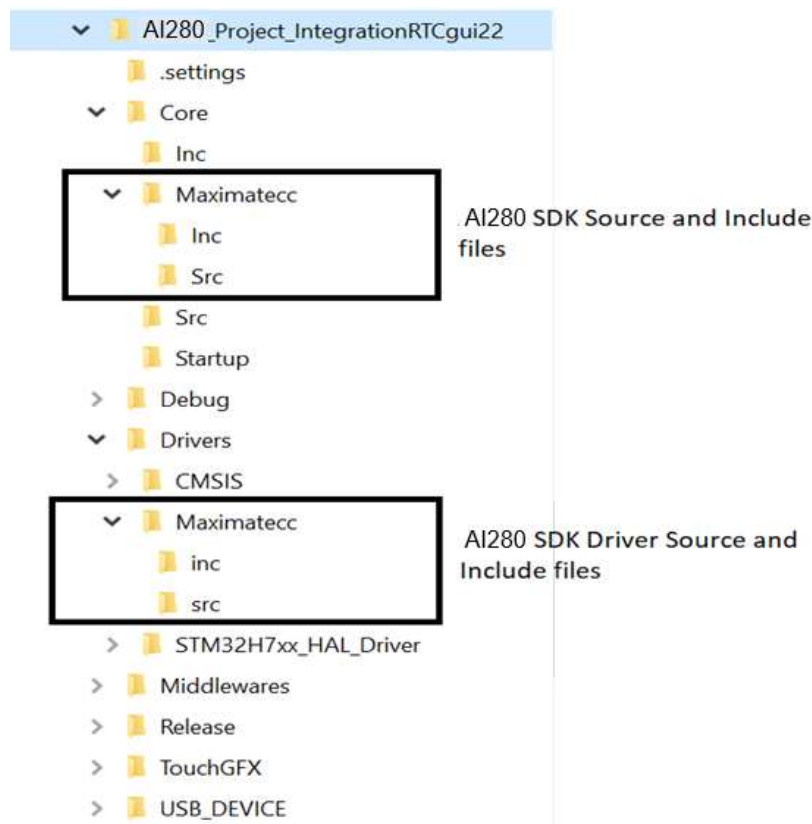
4.1.1 STM32 Cube IDE Setup.

Click on **STM32CubeIDE Launcher** and provide the Workspace name on the pop-up window given below and then click on **Launch** option given at the bottom right corner of the Pop-up notification to open the IDE with any desired workspace. We will then be adding our project into this workspace.
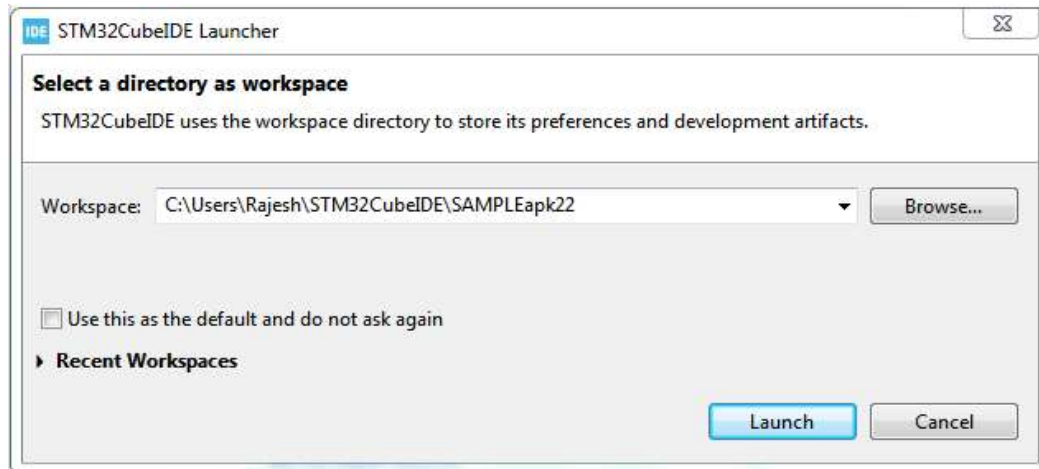
Once the user enters the Workspace, he can see the below screen.
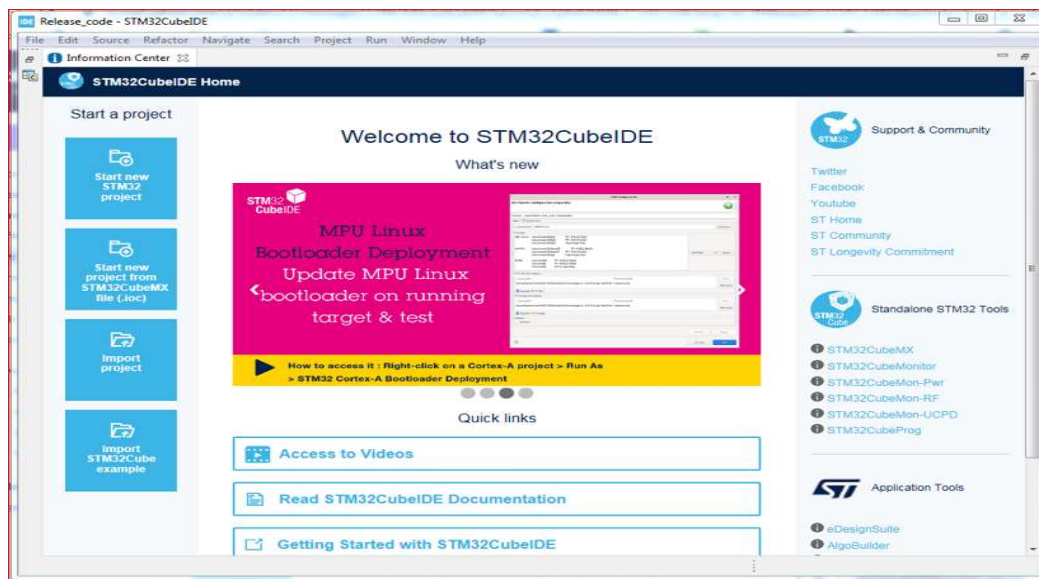
The user must now import the blank project into this workspace. Click file menu and select the option "Open Projects from File System". Refer the image given below.
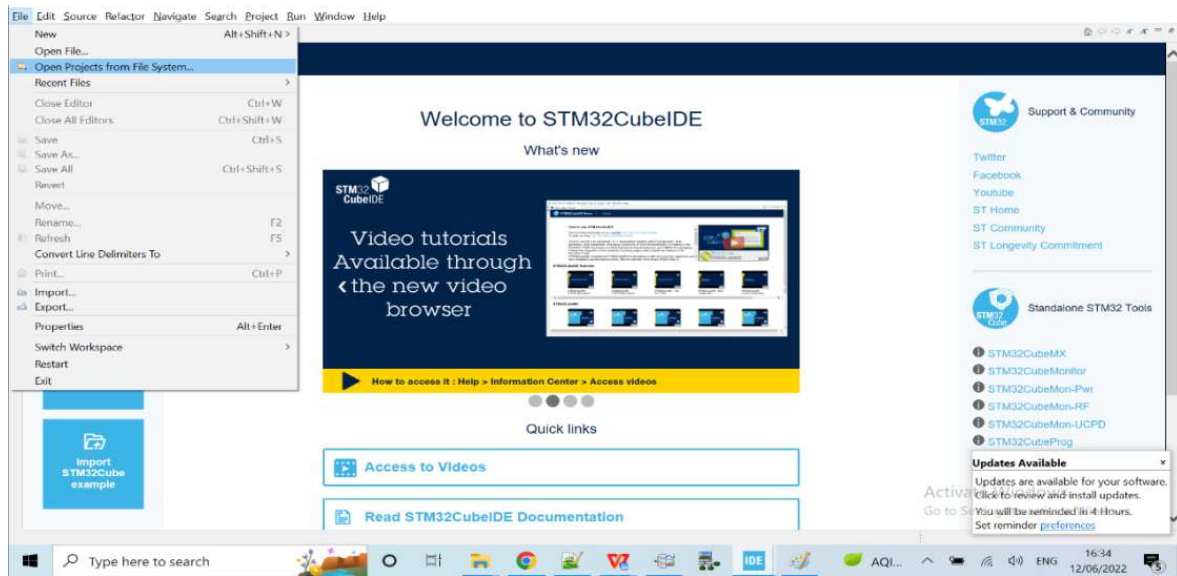
Clicking on the "Open Projects from File System" will bring up the below window. Please provide the path of the unzipped blank folder in the import source option.

Dept: Software	User Manual Document	Rev No.	Rev 3.0
		Date:	Oct 17, 2022
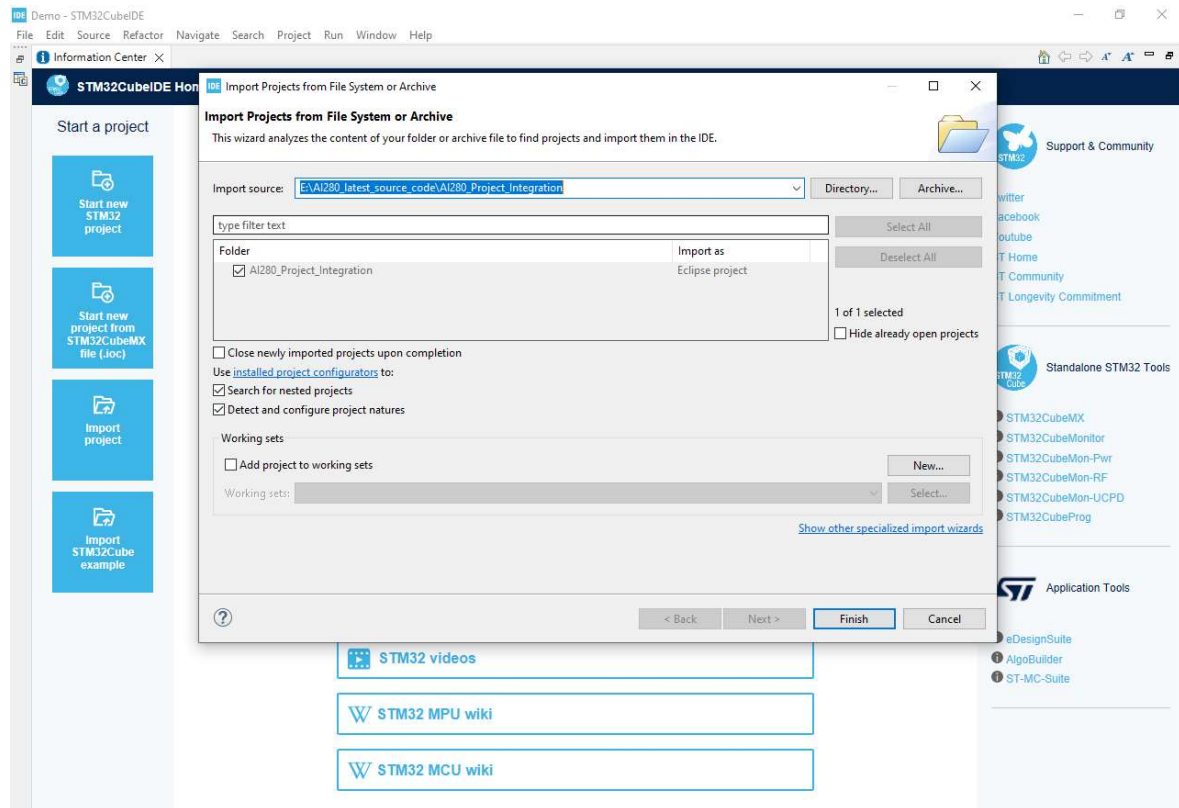
Now click on Finish once it recognizes the project file. You will get the below screen once you click on Finish.
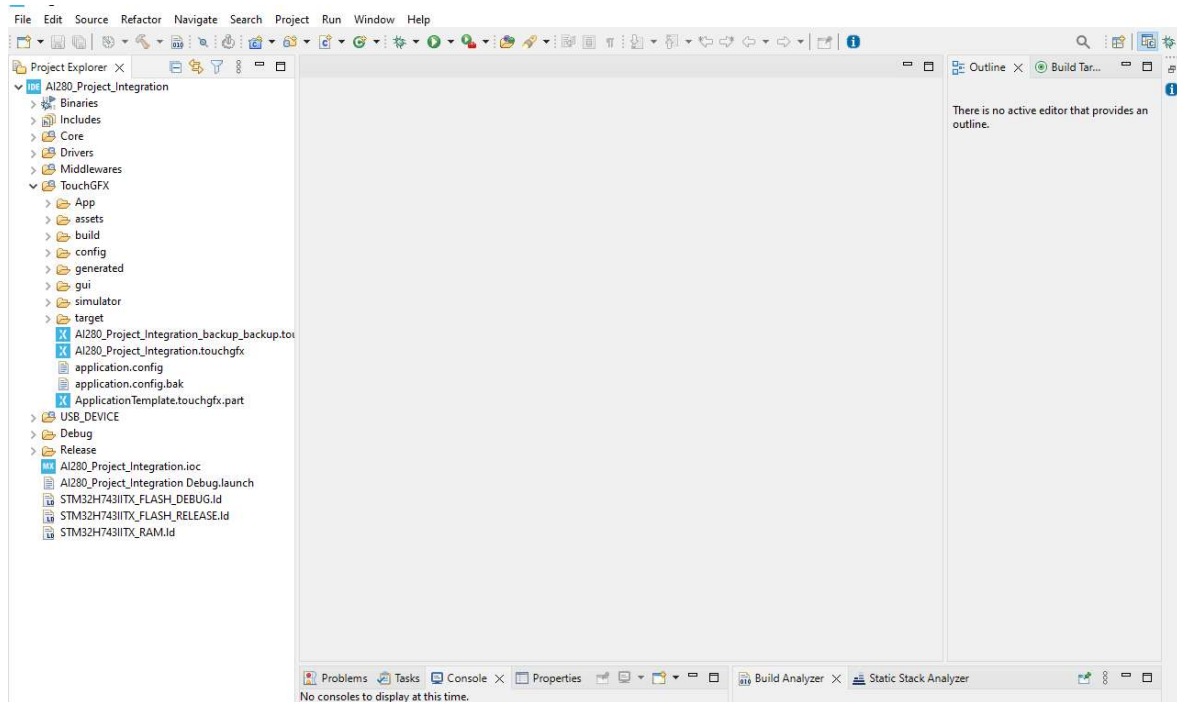
4.2 Build and Flash Procedure

Now we will create our own TouchGFX UI and compile and test it on the board. Expand the TouchGFX folder from the STM32CubeIDE and double click on the AI280_Project_Integration.touchgfx file. Refer the image given below.
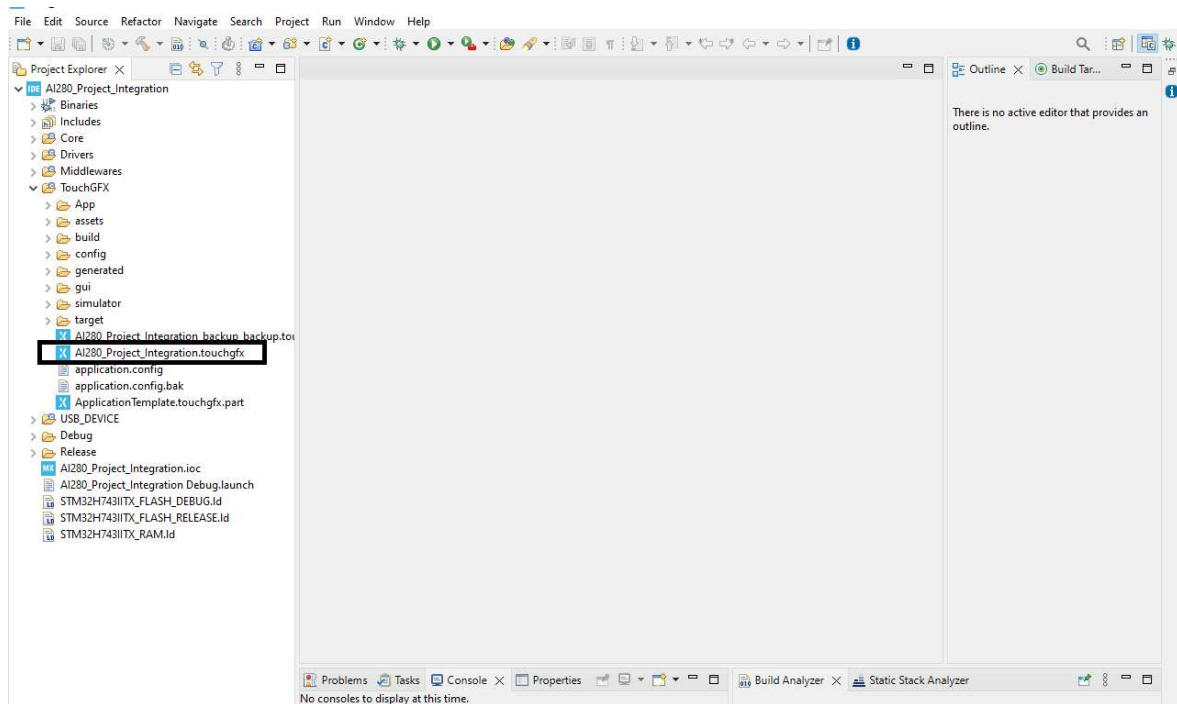
Once you double click the AI280_Project_Integration.touchgfx file, TouchGFX IDE will be opened with our designed UI as shown below.
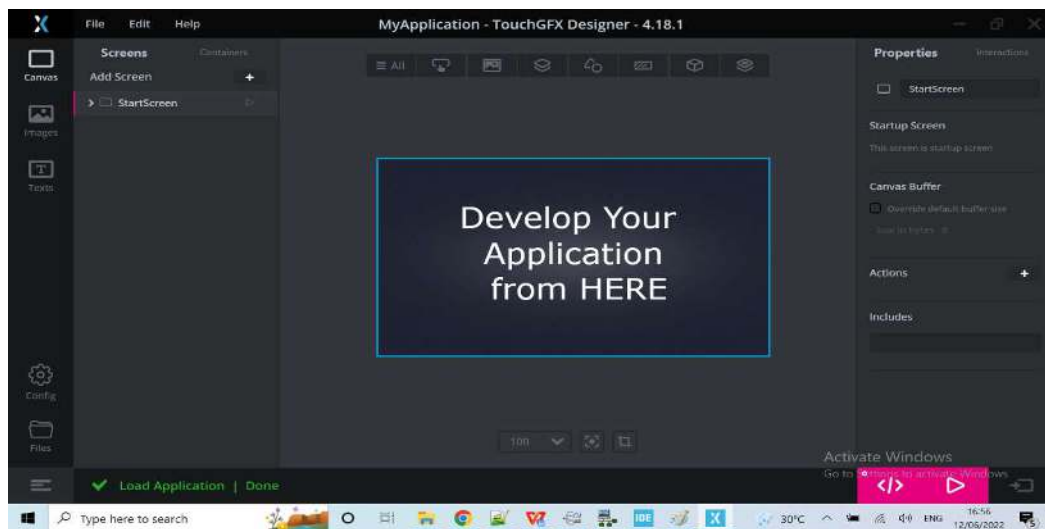
Now we will modify some text and test it on the board. Changed the text with "Modified this Text". Click "</>" button for generating the TouchGFX code. Refer the below image.
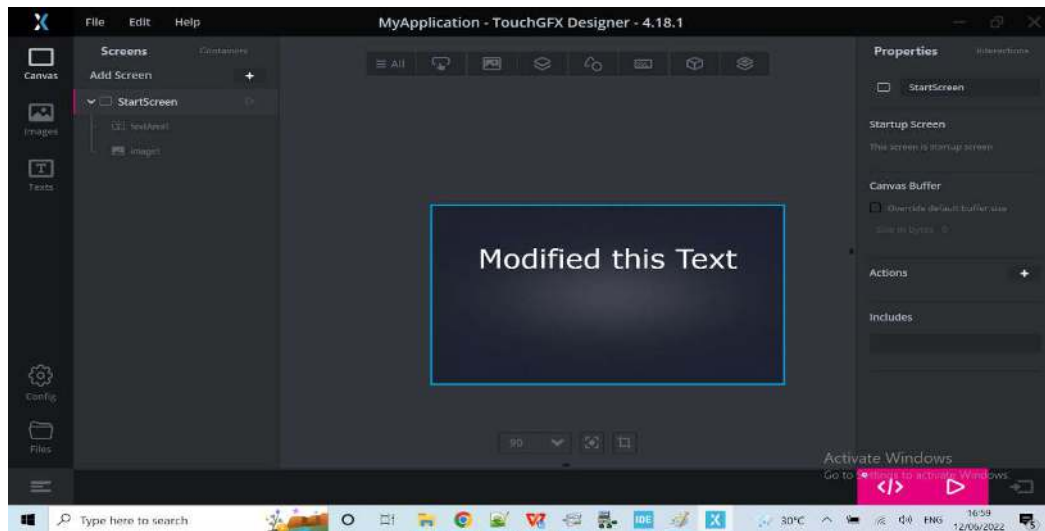
Go to STM32CubeIDE Screen. And click Project ===> Build All. Refer the below image.
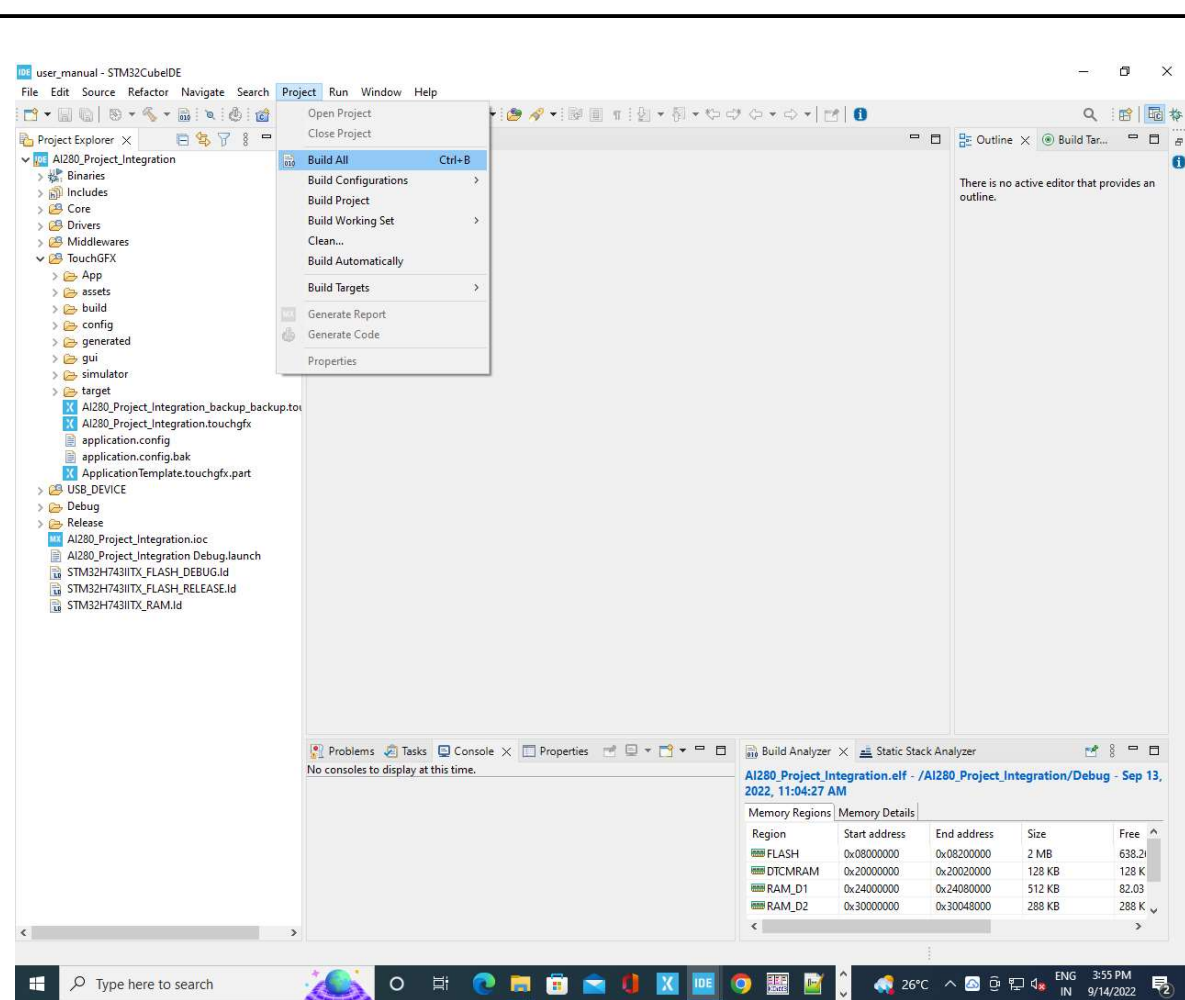
Once the build is success, you will get the console windows with "0 errors, 0 warnings".
As shown below

Now we will flash the binary using the ST JTAG, click debug ⚷ icon to flash the code. Refer the highlighted (▢) part in the below screen.

While flashing the code, you will get the below screen. Please click "Switch" button to continue.

Once the flashing is completed you will get the below screen.

Click Resume ▶ button (Highlighted ▢) in the above image) to run the application. You will get the below Screen on the AI280 board.

We were successfully able to compile and flash the blank project with minimal changes to the MAX AI280 board. In the next section we will elaborate with an example on how you can leverage the features of the SDK.

4.3 CAN Module Demo

This section elaborates the procedure on how the Touch GFX application interacts with the SDK. We have used the CAN module demo to explain this procedure.

4.3.1 Adding new GUI elements in the TouchGFX Screen

In this section, we will elaborate how we can add new elements in the touch GFX screen and then link them with the SDK.

Select the "Text Area" under "All". Now you can type the text in the text box. You need to select wildcard1 option for Gauge value. Gauge will fetch the data from CAN analyser and display it on the screen. Refer the below screen.
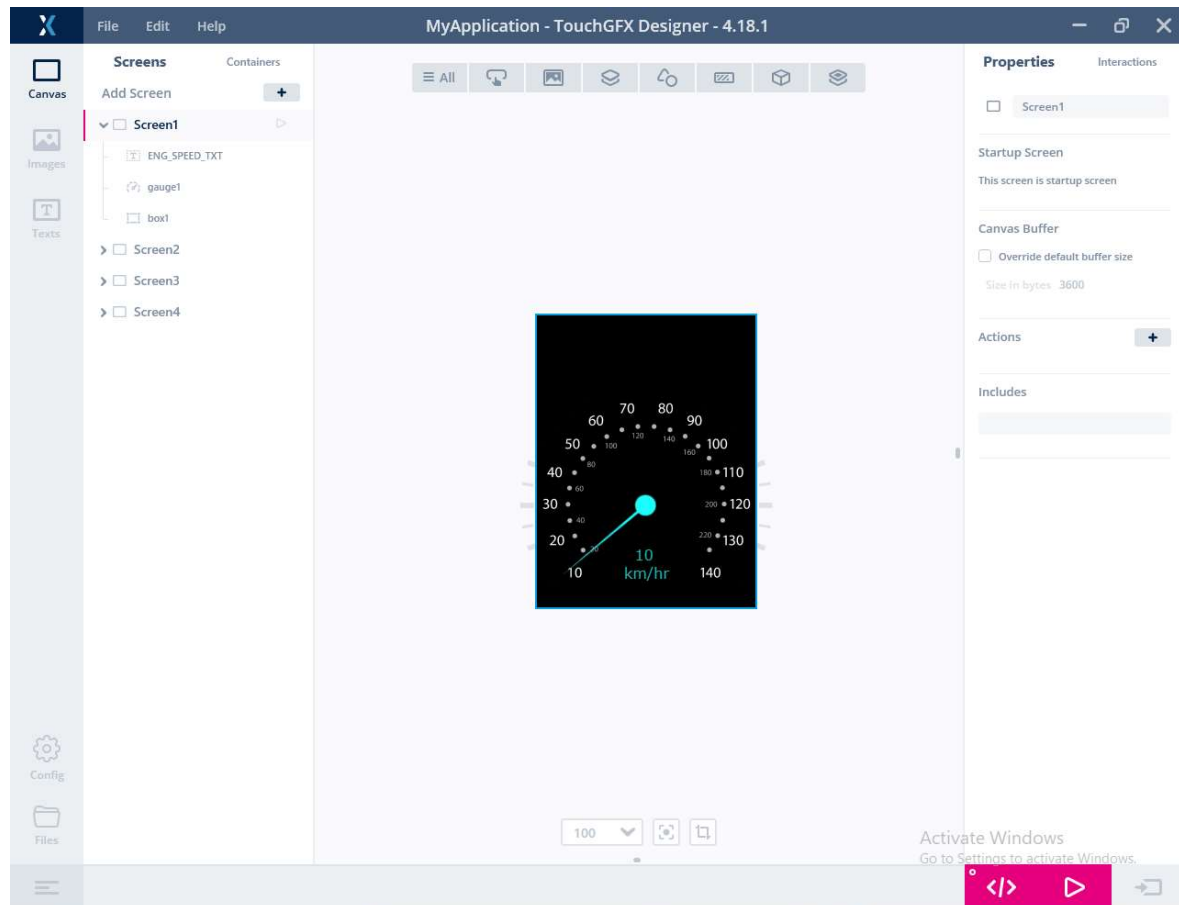
To create the gauge, select gauge under the section "All". The initial value can be from 0(Min) to 140(Max). You can assign the initial value under Properties section. Refer the below screen.
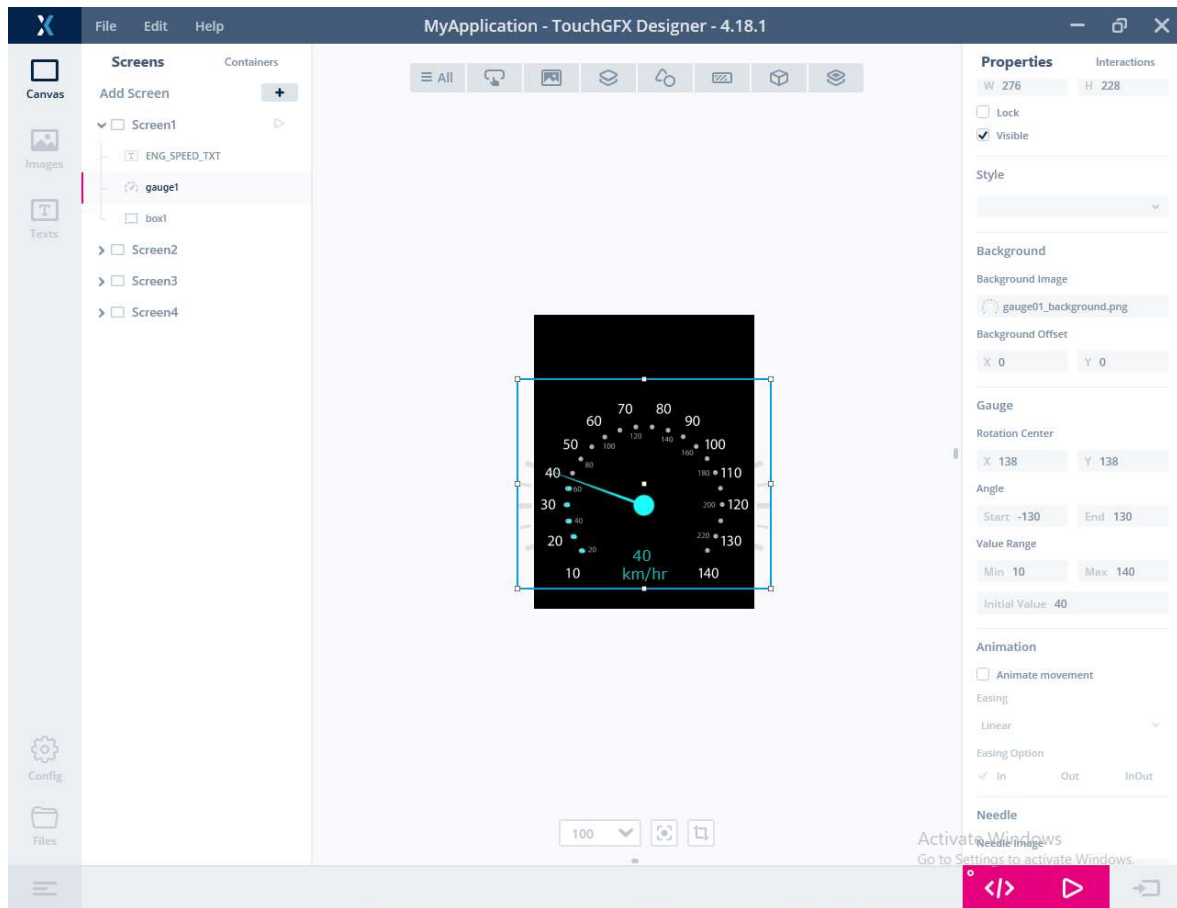
Select the "Text Area" under "All". Now drag the Text Area and place the text box over the circle. You need to select wildcard1 option for gauge value. Gauge will fetch the data from CAN and display it on the screen. Refer the below screen.
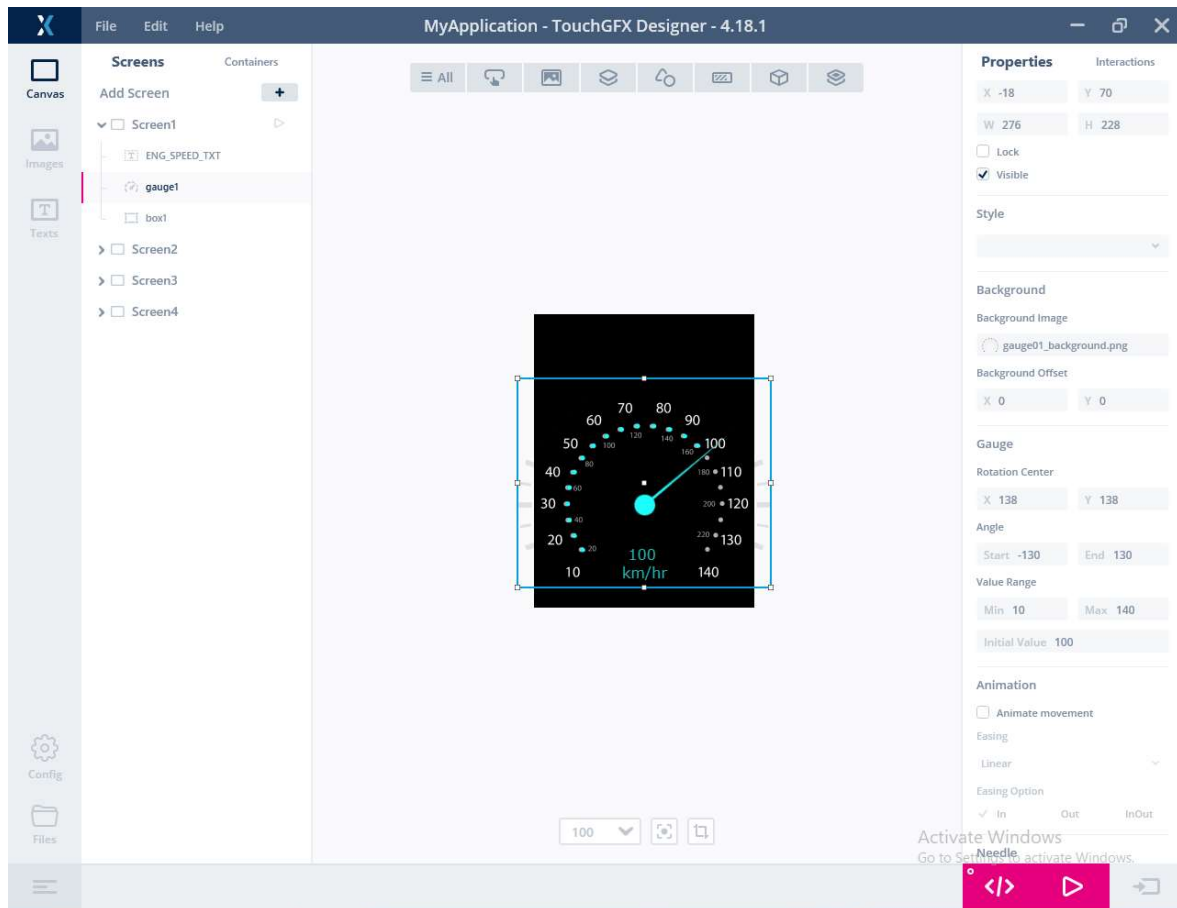
Click "</>" button for generating the TouchGFX code. Refer the highlighted part (▢)   in the below screen.

After the code is generated, please navigate to the STM 32 Cube IDE to edit the code and link the graphical elements to the SDK.

4.3.2   Edit the DB Variables

Now we are going to enable the graphical elements added in the last section to configure the CAN module. To do the same we will need to link it with the SDK DB.

In this section, we will describe how to link the DB Variables to the TouchGFX elements.

Go to STM32CubeIDE Screen. Add the following function declarations in

"Screen1Presenter.hpp" file under the "TouchGFX/gui/include/gui/Screen1_screen" folder structure.

" virtual void keyUpdate(uint8_t val); "
" virtual void handleTickEvent();"
" virtual void trigger();"
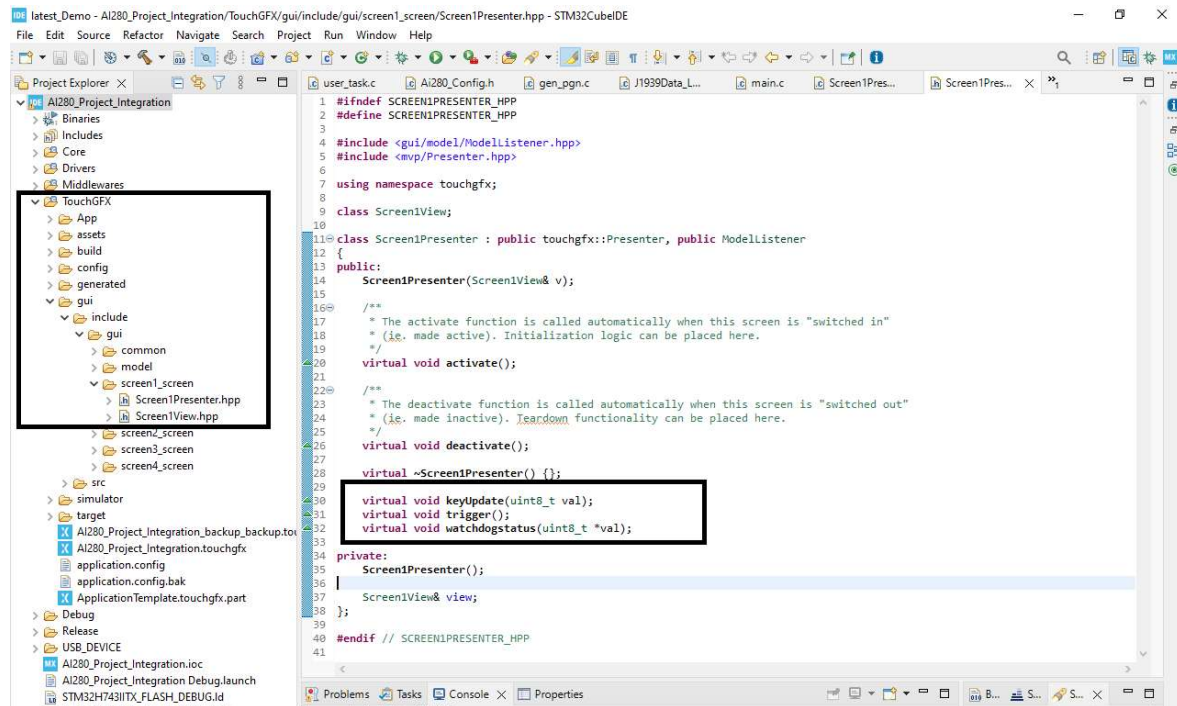
Add the following functions which will be used for the CAN Module in

"Screen1View.hpp" file under the

"TouchGFX/gui/include/gui/Screen1_screen" folder structure.

"virtual void keyUpdate(uint8_t val);"
"void trigger();"
"void moveNext();"
"void movePrevious();"
Add the following functions for screen navigation purposes.

"void moveNextScreen();"
"void moveBackScreen();"
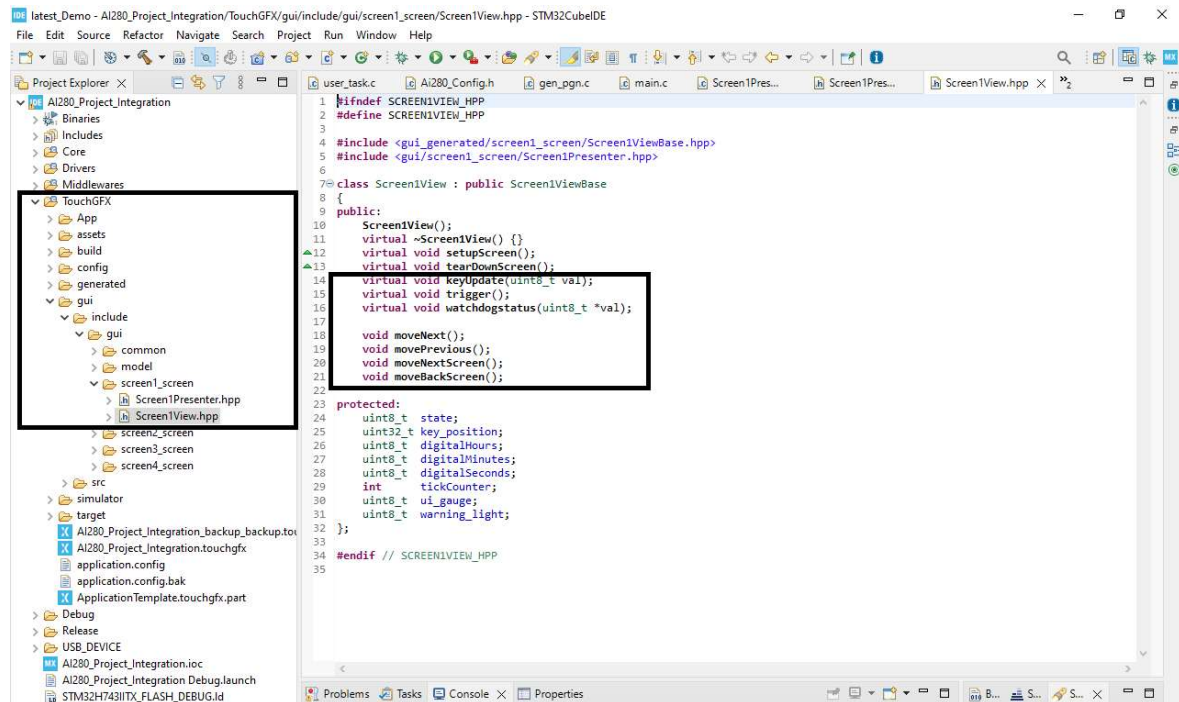
Add the following keypad function to access the functionality in "Screen1Presenter.hpp" file under the

"TouchGFX/gui/include/gui/Screen1_screen" folder structure.

```
void Screen1Presenter::keyUpdate(uint8_t val);
{
    view.keyUpdate(val);
}
```
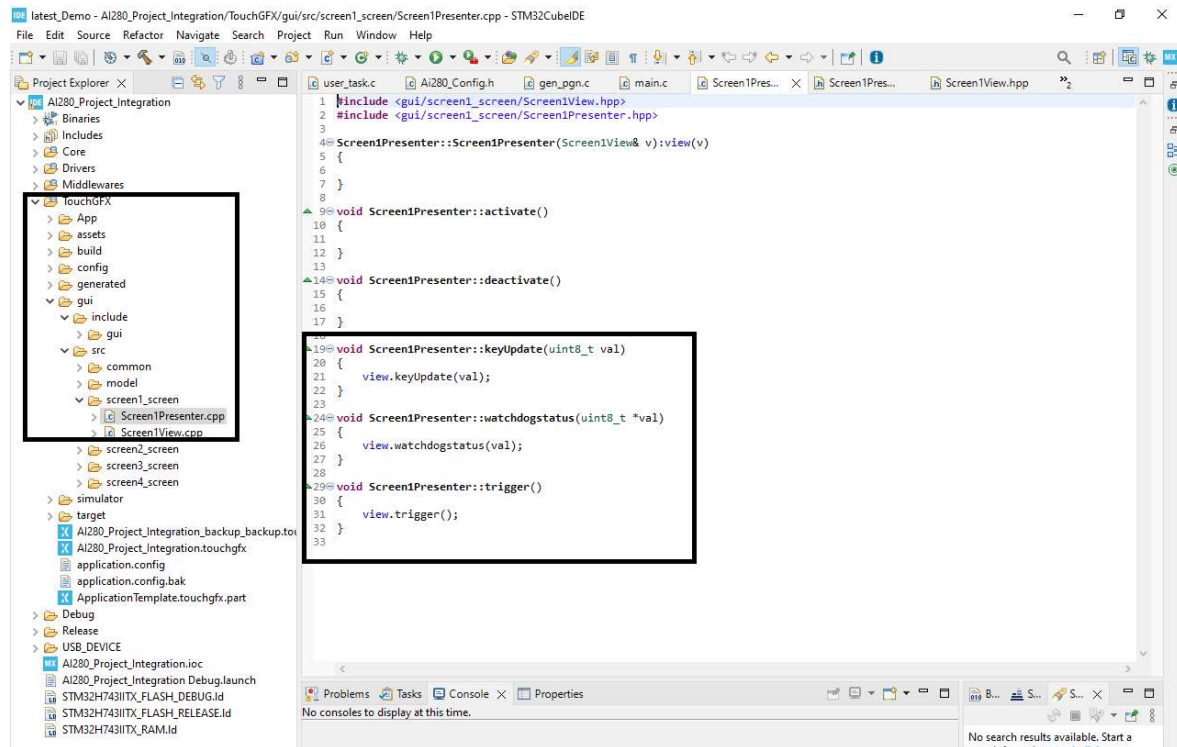
To access the DB variables, the following header need to be added.

#include "../../../../Core/Maximatecc/Inc/User_Define.h"

To highlight the selected text, include the following line.

#include <touchgfx/Color.hpp>

Refer the below screen to see the code snippet and file path.
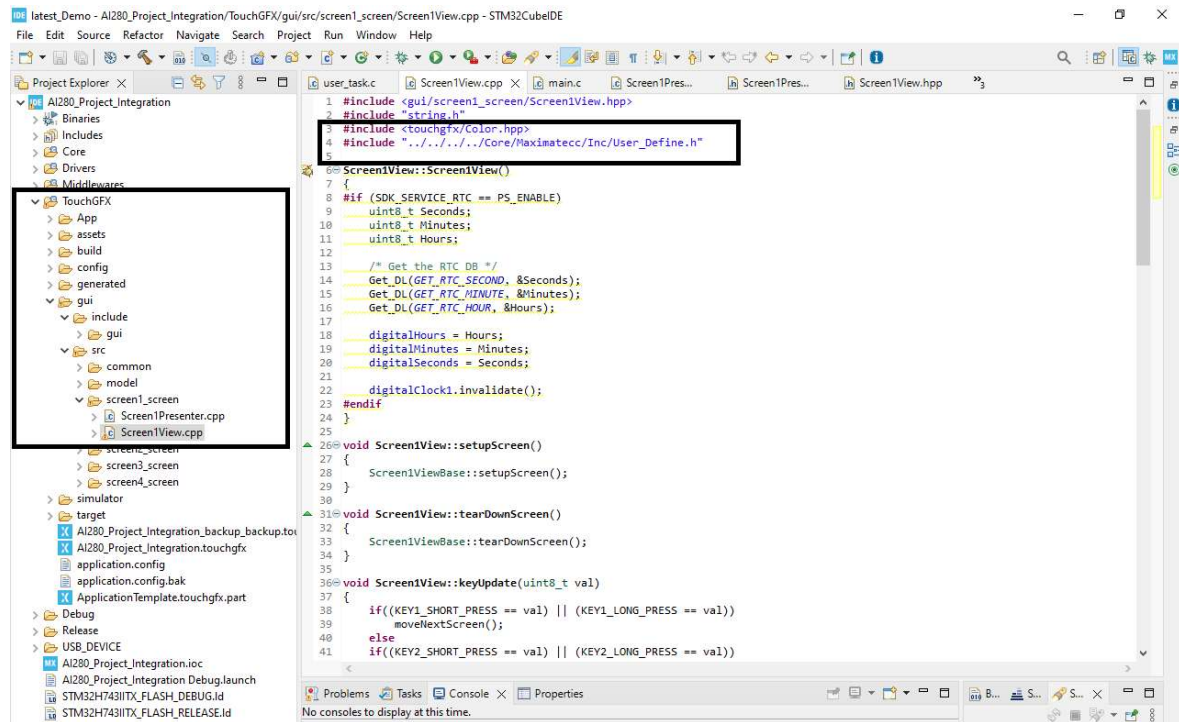
Based on the Gauge data, it will adjust Needle movement. Get_DL function is used to read the DB variable of CAN data. Set_DL function is used to modify the DB variable of CAN.
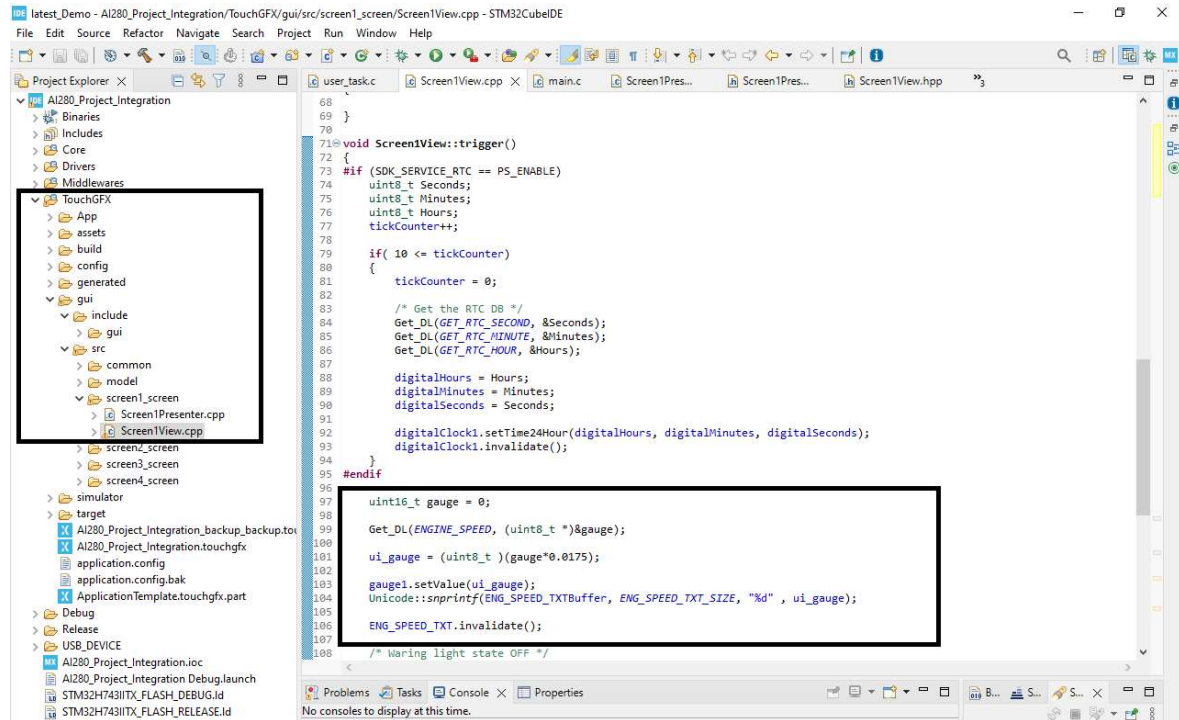
4.3.3 Configurations

This section describes the configurations required for CAN module. This default configuration can be done in the AI280_config.h file under the section core/Maximatecc/Inc.

The below are the configurable parameters available for CAN module

To configure these, we are using

FDCAN1_BAUDRATE	BAUDRATE_50K
FDCAN1_IDENTIFIER	0x19FEFCFE //0x111
FDCAN1_IDTYPE	FDCAN_EXTENDED_ID
FDCAN1_TXFRAMETYPE	FDCAN_DATA_FRAME
FDCAN1_DATALENGTH	FDCAN_DLC_BYTES_8
FDCAN1_ERRORSTATEIND	FDCAN_ESI_ACTIVE
FDCAN1_BITRATESWITCH	FDCAN_BRS_ON
FDCAN1_FDFORMATE	FDCAN_FD_CAN
FDCAN1_TXEVENTFIFOCONTROL	FDCAN_STORE_TX_EVENTS
FDCAN1_MESSAGEMARKER	0

FDCAN1_ID	FDCAN_STANDARD_ID
FDCAN1_FILTERINDEX	0
FDCAN1_FILTERTYPE	FDCAN_FILTER_DUAL
FDCAN1_FILTERCONFIG	FDCAN_FILTER_TO_RXBUFFER
FDCAN1_FILTERID1	FDCAN_DEAFULT_FILTERID1
FDCAN1_FILTERID2	FDCAN_DEAFULT_FILTERID2

Kindly refer the below screen for code snippet.
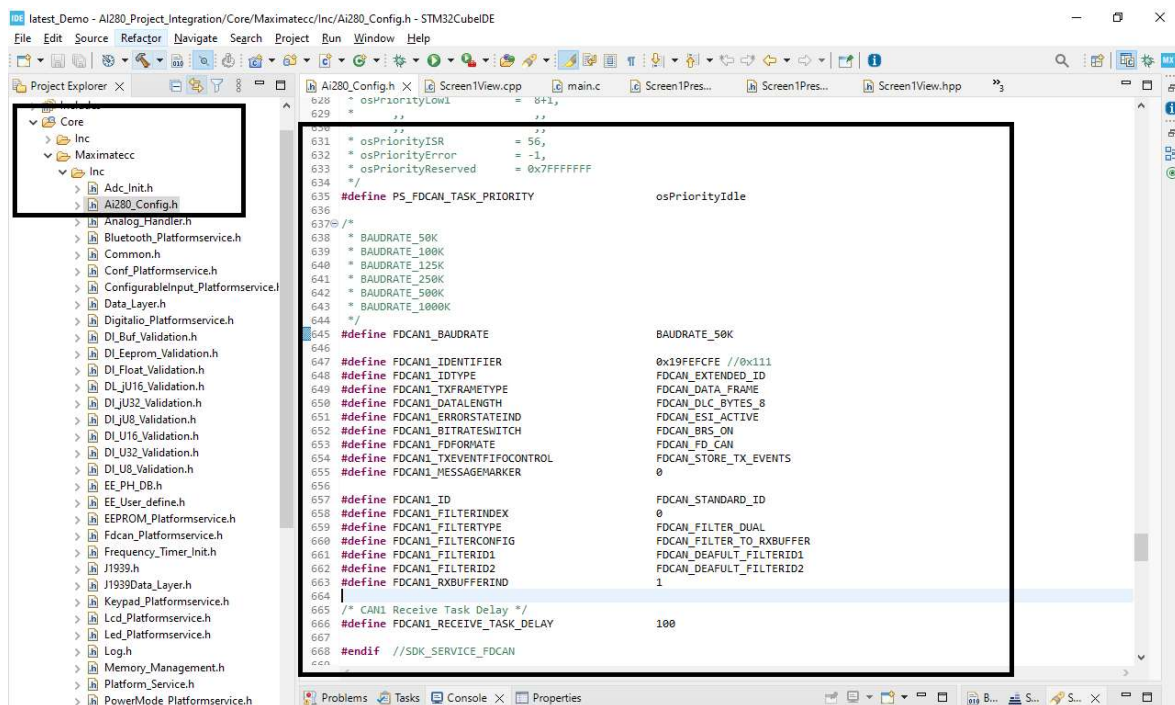
Now we need to compile and test it on the board. For compilation, follow the instructions provided under the section 3.2

4.3.4 Output

Once we flashed the updated binary on MAX AI280 board we will observe that the application we designed is launched on the GUI.

Refer the below screens captured to show different Gauge values being updated in the UI.

Dept: Software	User Manual Document	Rev No.	Rev 3.0
		Date:	Oct 17, 2022

4.4 LED Demo

The demo application is written for user who would like to write application without using the TouchGFX. They can use the same blank project released in the SDK, but disable the touch GFX and then write code which will link with the SDK and use the functionalities but will have a blank UI.

To Disable TouchGFX we need to click on AI280_Project_Integration.ioc.

Go to STM32CubeIDE Screen. Now double click on AI280_Project_Integration.ioc. The below screen will come up.
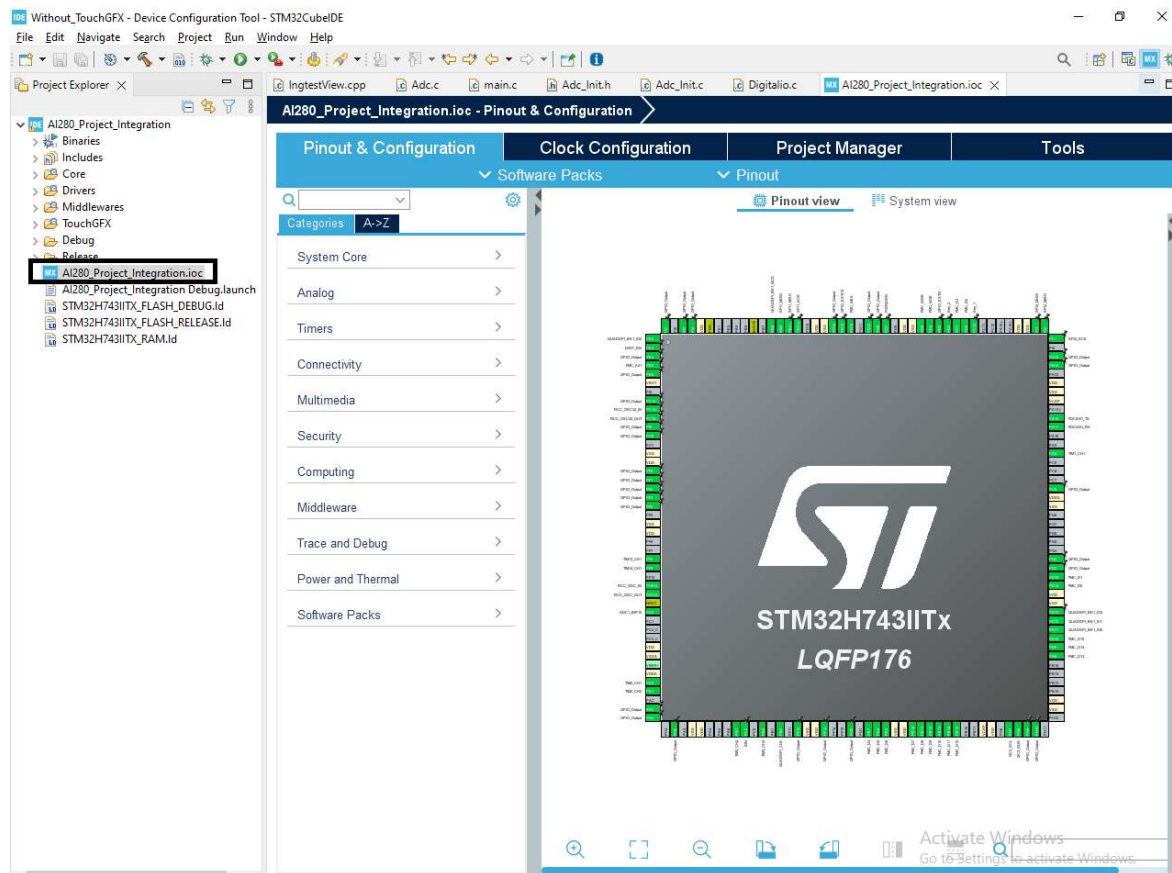
You will get the below screen once you click on AI280_Project_Integration.ioc. Now click Pinout & Configuration ➔ Categories ➔ Software Packs ➔ STMicroelectronics.X-CUBE-TOUCHGFX. Now you can untick Graphics Application box. This will disable the TouchGFX configurations. Refer the highlighted areas in the below screen.
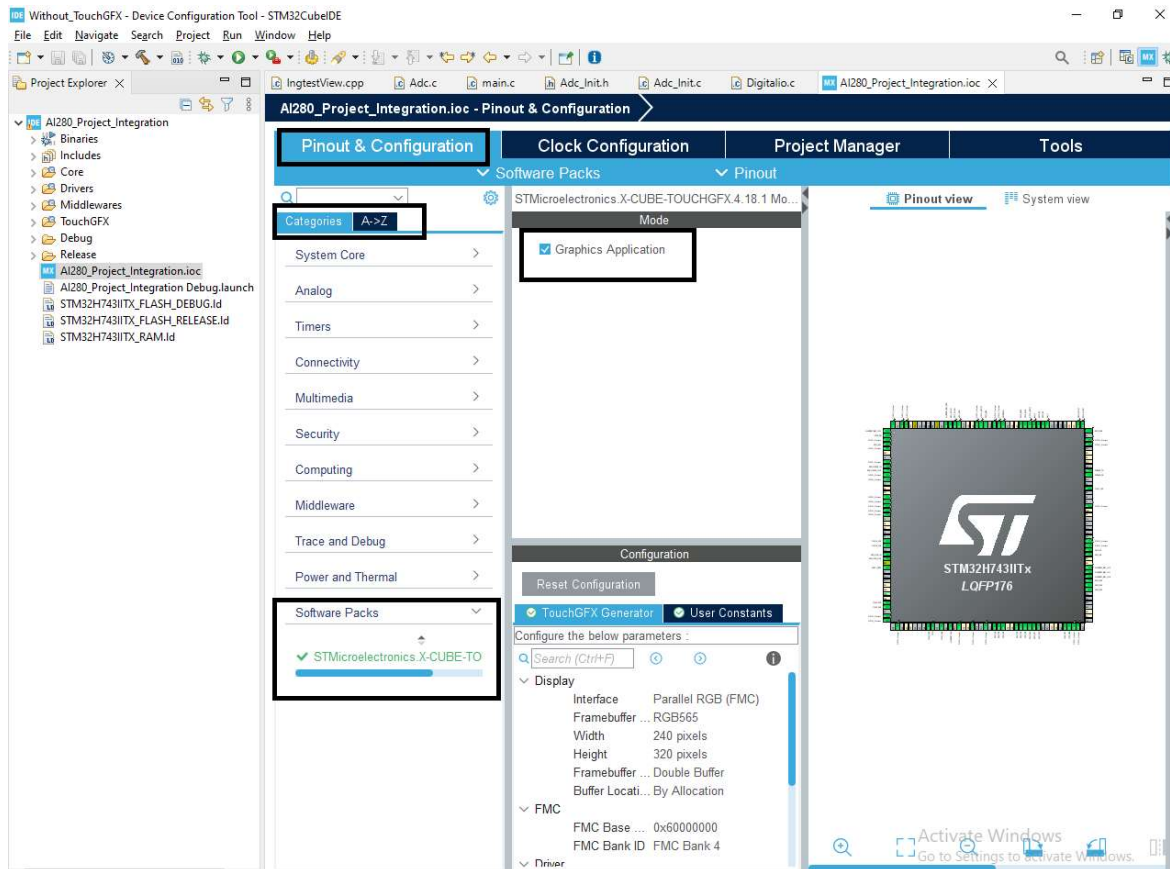
Now click Pinout & Configuration ➔ Software Packs ➔ Select Components ➔ STMicroelectronics.X-CUBE-TOUCHGFX ➔ Graphics Application ➔ Application. Then select "Not selected" from the list. Then click on OK.

Dept: Software	User Manual Document	Rev No.	Rev 3.0
		Date:	Oct 17, 2022
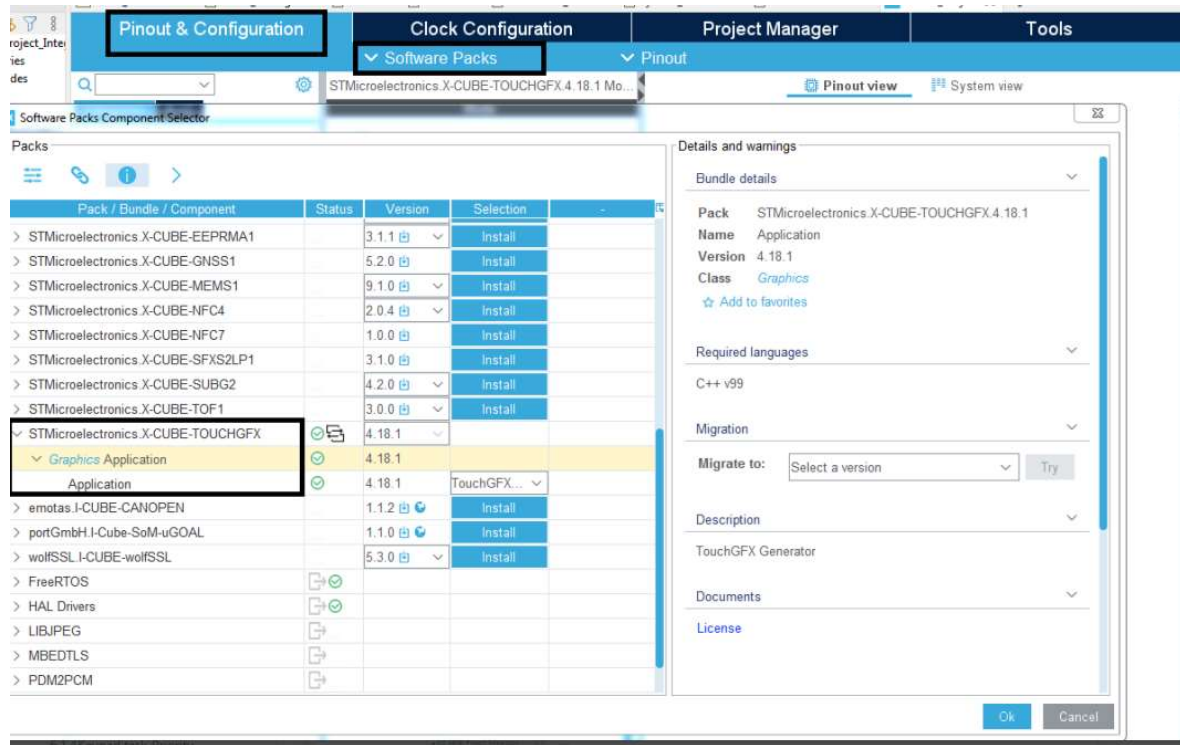
Now TouchGFX is disabled.

The next step would be for the user to generate the code so that he can add his features. Click Device Configuration Tool Code Generation [icon] button.
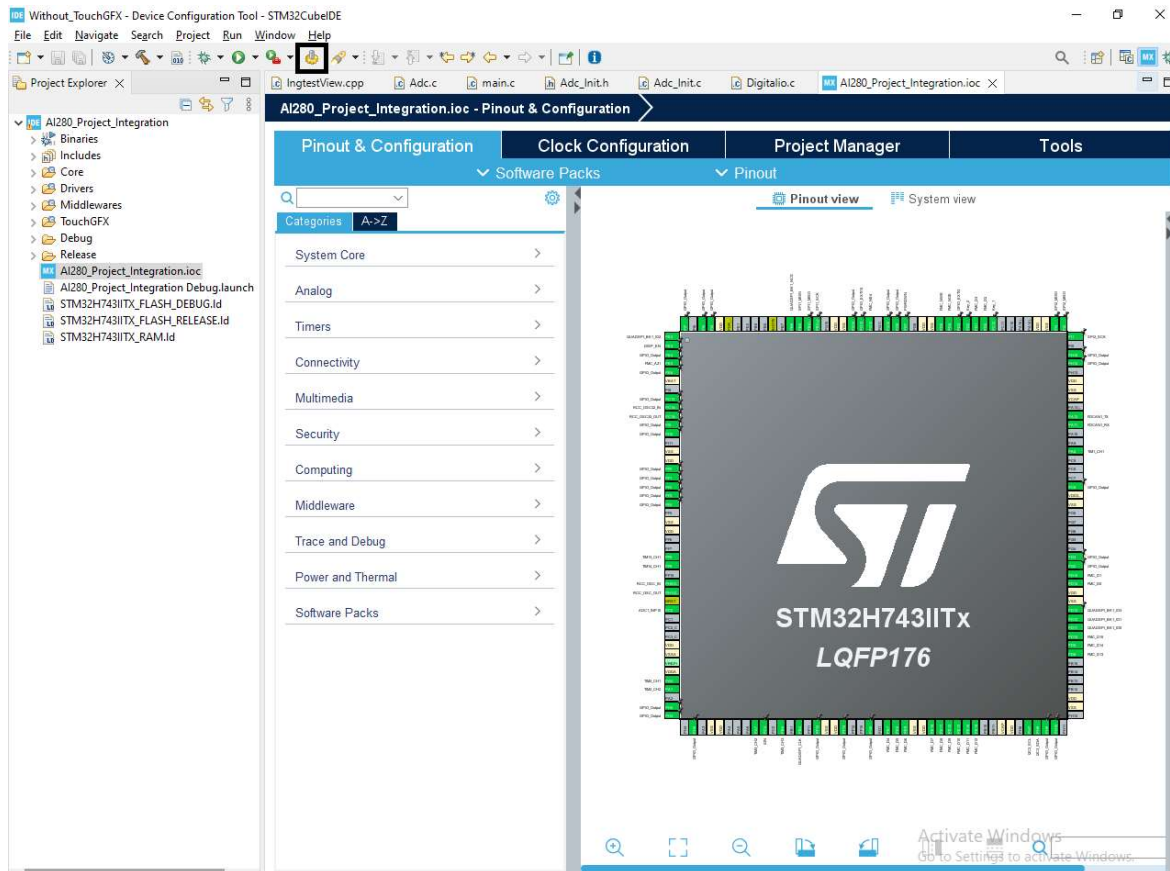
To add the Test task function, go to main.c file under the section core/src. Refer the below screen.
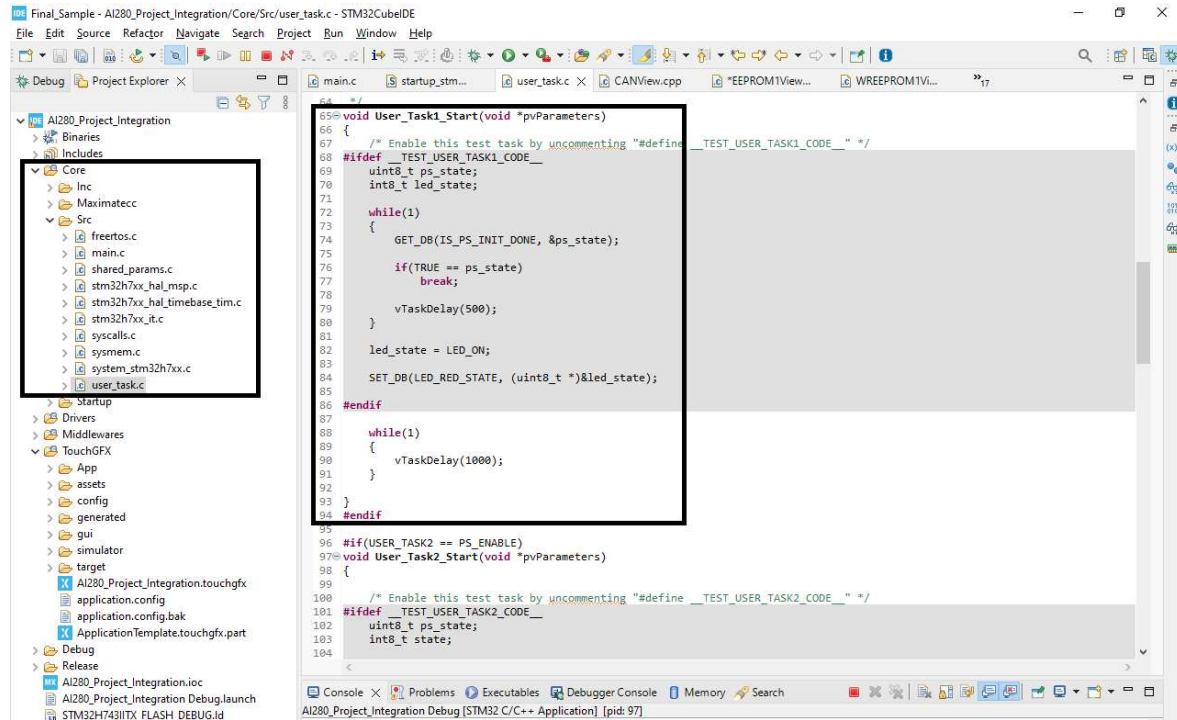
Now create the test task function. Refer the screen below for code snippet.
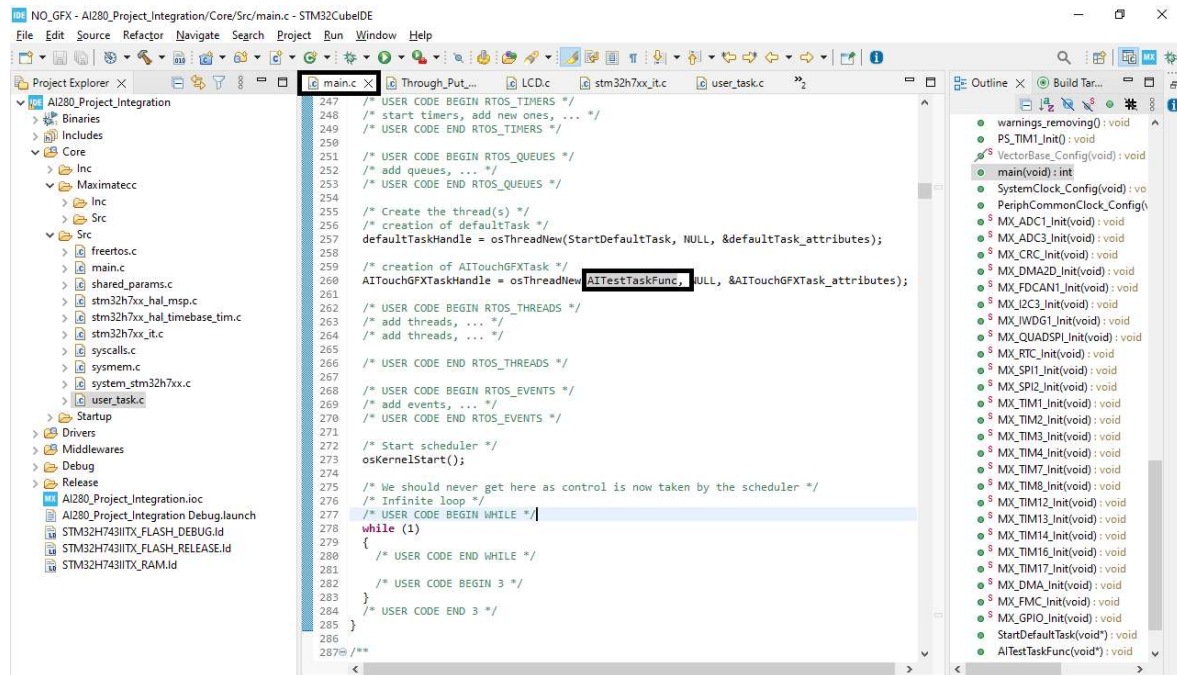
The user can add his code inside this test task.

```
#endif

#if(USER_TASK3 == PS_ENABLE)
void User_Task3_Start(void *pvParameters)
{
        /* Enable this test task by uncommenting "#define __TEST_USER_TASK3_CODE__" */
#ifdef __TEST_USER_TASK3_CODE__
    uint8_t ps_state;
    int8_t led_state;

    while(1)
    {
            GET_DB(IS_PS_INIT_DONE, &ps_state);

            if(TRUE == ps_state)
                    break;

            vTaskDelay(500);

    }

    led_state = LED_ON;

    SET_DB(LED_AMB_STATE, (uint8_t *)&led_state);
#endif

    while(1)
    {
            vTaskDelay(1000);
    }

}
#endif
```

For example, we have added the below lines for LED functionality. User can call Get_DL and Set_DL functions inside AITestTaskFunc. Refer the below screen for code snippet.

Now click Project ===> Build All. Refer the below image.

Once the build is successful, you will get the console windows with "0 errors, 0 warnings." As shown below.

Now we will flash the binary using the ST JTAG, click debug icon to flash the code. Refer the highlighted (☐) part in the below screen.

During the flashing, the code you will get the below screen. Please click "Switch" button to continue.

Once the flashing is completed you will get the below screen.

Click Resume ▶ button (Highlighted(▫) in the above image) to run the application. You will get the below Screen on the AI280 board.

In the AITestTaskFunc, we enabled LED. You can see the below screen shot where the AMBER LED is enabled.

4.5 User Task Edit Details

This section elaborates how user can initialize their user task and call the APIs, Set_DL and Get_DL without using the TouchGFX. They can also perform any non-UI related tasks here. User can create the tasks as shown below. Below images shows that the user is creating three tasks. Refer to the below two images for the sample code snippet to create the tasks.

These tasks are created in their user_task.c file as shown in the below image,
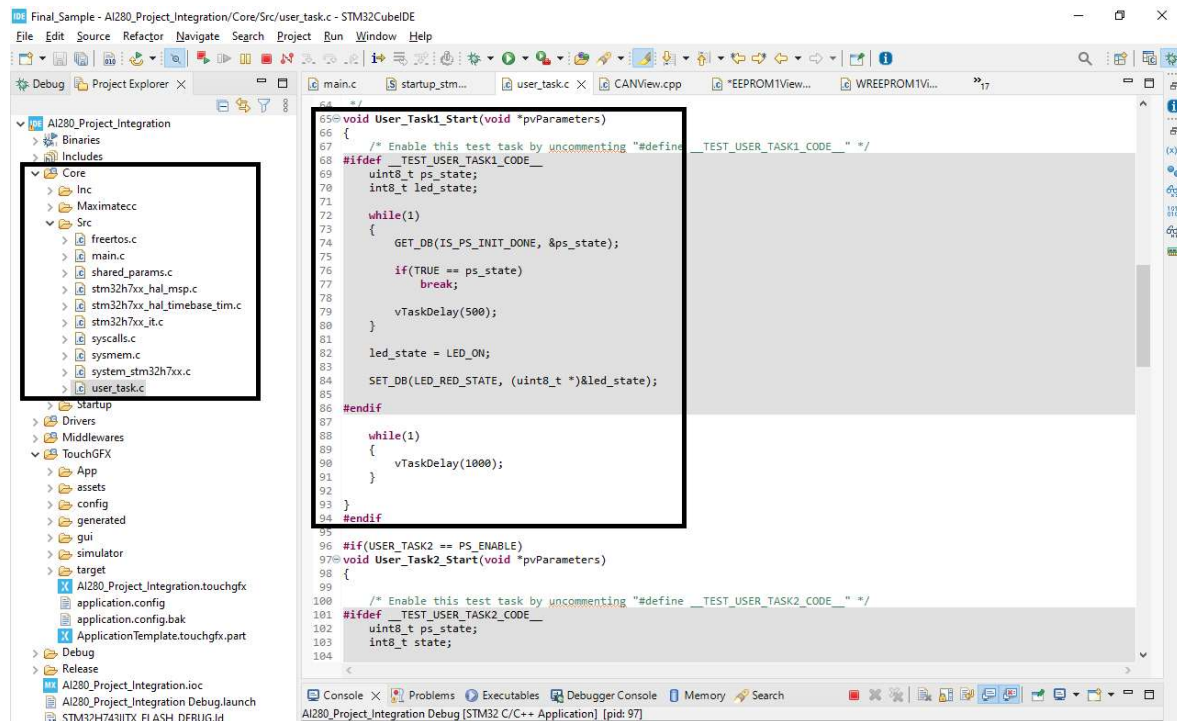
```
*/
int8_t User_Task_Start(void)
{
#if((USER_TASK1 == PS_ENABLE) || (USER_TASK2 == PS_ENABLE) || (USER_TASK3 == PS_ENABLE))
        BaseType_t xReturned;
#endif

#if(USER_TASK1 == PS_ENABLE)
        /* Start the User Task1 Start service thread */
        xReturned = xTaskCreate(User_Task1_Start, "User_Task1",
                        512, NULL, osPriorityIdle,   &user_task1);

        if (xReturned != pdPASS)
        {
                LOGERROR("Unable to initialize the User_Task1 \r\n");
                return PS_FAILURE;
        }
#endif

#if(USER_TASK2 == PS_ENABLE)
        /* Start the User Task2 Start service thread */
        xReturned = xTaskCreate(User_Task2_Start, "User_Task2",
                        512, NULL, osPriorityIdle,   &user_task2);

        if (xReturned != pdPASS)
        {
                LOGERROR("Unable to initialize the User_Task2 \r\n");
                return PS_FAILURE;
        }
#endif




#if(USER_TASK3 == PS_ENABLE)
        /* Start the User Task3 Start service thread */
        xReturned = xTaskCreate(User_Task3_Start, "User_Task3",
                        512, NULL, osPriorityIdle,   &user_task3);

        if (xReturned != pdPASS)
        {
                LOGERROR("Unable to initialize the User_Task3 \r\n");
                return PS_FAILURE;
        }
#endif

        return PS_SUCCESS;
}
#endif
```

User can call SET_DB and GET_DB APIs inside the created user tasks to access the required functionalities. Please see the below image for the code snippets.

```
*/
void User_Task1_Start(void *pvParameters)
{
        /* Enable this test task by uncommenting "#define __TEST_USER_TASK1_CODE__" */
#ifdef __TEST_USER_TASK1_CODE__
        uint8_t ps_state;
        int8_t led_state;

        while(1)
        {
                GET_DB(IS_PS_INIT_DONE, &ps_state);

                if(TRUE == ps_state)
                        break;

                vTaskDelay(500);
        }

        led_state = LED_ON;

        SET_DB(LED_RED_STATE, (uint8_t *)&led_state);

#endif

        while(1)
        {
                vTaskDelay(1000);
        }

}
#endif
```

```c
#if(USER_TASK2 == PS_ENABLE)
void User_Task2_Start(void *pvParameters)
{

        /* Enable this test task by uncommenting "#define __TEST_USER_TASK2_CODE__" */
#ifdef __TEST_USER_TASK2_CODE__
        uint8_t ps_state;
        int8_t state;

        while(1)
        {
                GET_DB(IS_PS_INIT_DONE, &ps_state);
                if(TRUE == ps_state)
                        break;

                vTaskDelay(500);
        }

        state = KEY_BACKLIGHT_ON;

        Set_DL(KEYPAD_BACKLIGHT , (uint8_t *)&state);
#endif

        while(1)
        {
                vTaskDelay(1000);
        }

}
#endif


#if(USER_TASK3 == PS_ENABLE)
void User_Task3_Start(void *pvParameters)
{
        /* Enable this test task by uncommenting "#define __TEST_USER_TASK3_CODE__" */
#ifdef __TEST_USER_TASK3_CODE__
        uint8_t ps_state;
        int8_t led_state;

        while(1)
        {
                GET_DB(IS_PS_INIT_DONE, &ps_state);

                if(TRUE == ps_state)
                        break;

                vTaskDelay(500);
        }

        led_state = LED_ON;

        SET_DB(LED_AMB_STATE, (uint8_t *)&led_state);
#endif

        while(1)
        {
                vTaskDelay(1000);
        }

}
#endif
```

If user enables the USER_TASK_STATE in AI280_Config.h file, User_Task_Start function gets called from Platform_Service.c. Refer the below screen for code snippet.
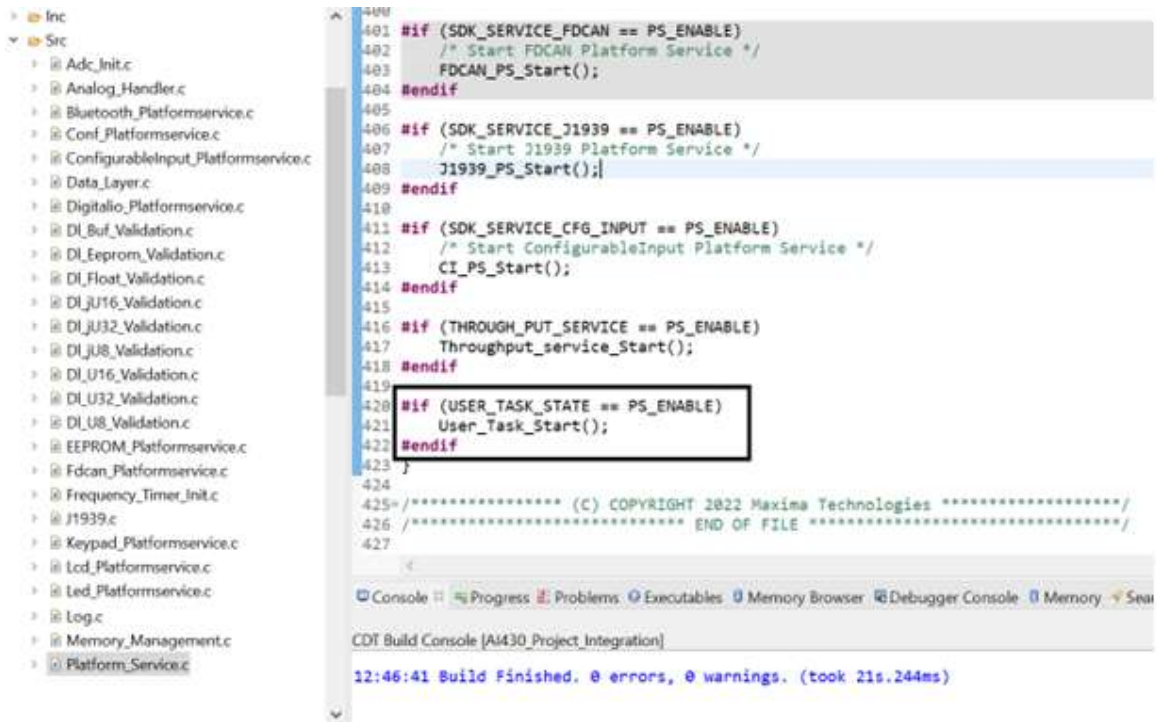
Before using the tasks, the user will need to initialize the tasks from the main.c. User can call there USERTASK1_PS_Start under platform service init as shown in the below image.

User need to create and enable the User Tasks in the configuration file. Refer the below screen for code snippet.

The user can then compile and flash the same following the below procedure.

Now click Project ===> Build All. Refer the below image

Once the build is successful, you will get the console windows with "0 errors, 0 warnings." As shown below.

Now we will flash the binary using the ST JTAG, click debug ⚙ icon to flash the code. Refer the highlighted (☐) part in the below screen.

During the flashing, the code you will get the below screen. Please click "Switch" button to continue.

Once the flashing is completed you will get the below screen.

Click Resume ▶ button (Highlighted ☐ in the above image) to run the application. You will get the below Screen on the AI280 board.

Once the device powers up with this build user can see the output on the board. By default, the MAXAI280 LED state is OFF but, in the user, task3 that we created and flashed we have changed the LED state as ON.

Hence if we power up the board with this build, the user can see that LED is ON as shown below which confirms that the user task 3 has executed successfully.

Hence the user can add any non-UI based functionalities in these tasks and execute them in the background.

5   SDK Overview

5.1 SDK Architecture

The AI280 SDK is built with the below four layers which are well organized to give the application the flexibility to be written independently with minimum knowledge regarding in the internal functionality of the underlying hardware.
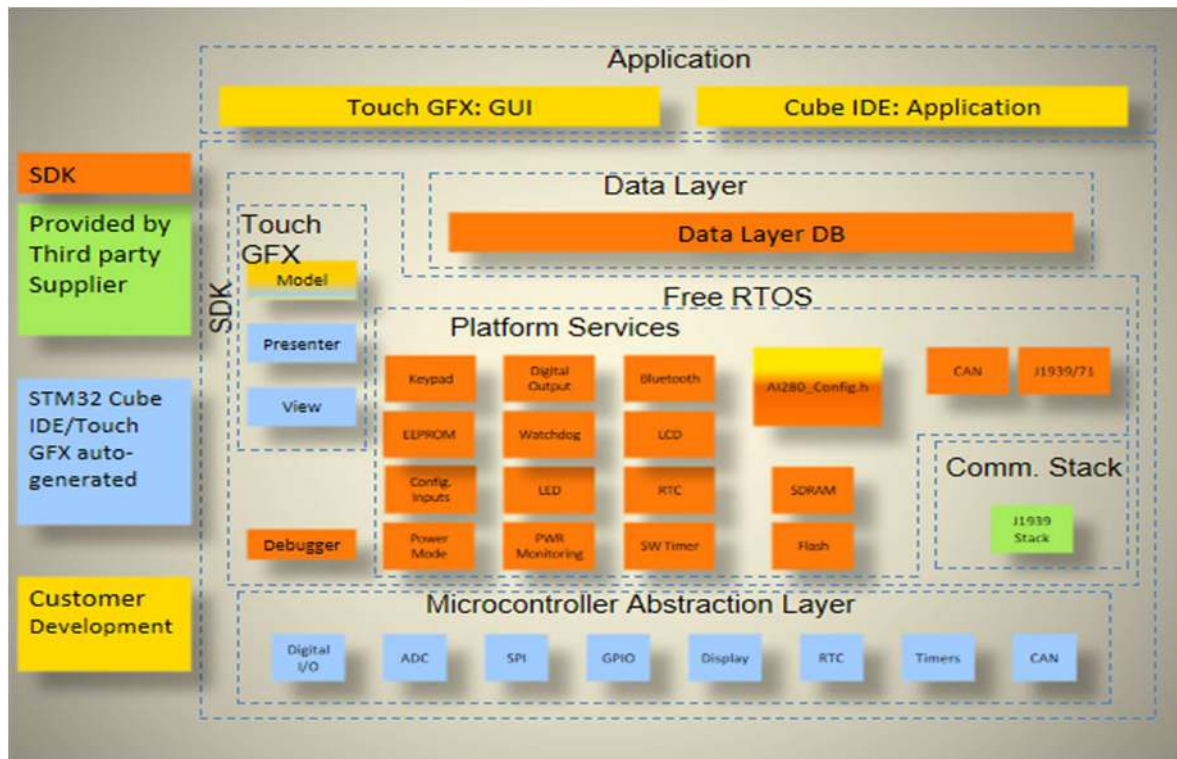
A quick overview of the layers is described below,

Application   :

Blank Touch GFX will be provided where USER will be able to create the GUI layout using Touch GFX or USER developed widgets, Hardware configuration of the Touch GFX project would be predefined in the SDK. The user can create the graphical elements and link then with the SDK modules to achieve the desired results. The user modified code resides here.

Data Layer DB:

Data Layer DB is the interface between USER application and the SDK Platform services. It acts as an intermediate layer and used for communication between user application and platform services.

Data Layer DB consist of a collection of RAM variables containing the data of the platform services, this data shall be updated with latest data from each platform service iteration/event. The Data Layer DB will also work as a channel to input data from the application to the Platform services.

Platform Services   :

Platform Services will work as an interface between Data Layer DB and platform driver. It will create and manage tasks for hardware peripherals based on user configuration/application requirements. These created tasks will run in RTOS. Data requested from user application will be obtained by Platform services from platform driver. After receiving data, Platform services will push that information to Data Layer DB. Then, user application can fetch requested data from the Data Layer DB.

Platform Drivers   :

Platform Drivers will be used for accessing and controlling the hardware. Platform drivers will configure the hardware based on user configuration/application requirements. It will receive the relevant data needed by user application. Data received by platform driver will be sent to platform services.

5.2 SDK Interfaces

The SDK adaptation software provides an interface between the USER tasks and platform driver layer on AI280 hardware platform. This design provides the easy to include / exclude design for the SDK modules/drivers using the configuration file (.h) in the final firmware application. And the USER can easily integrate the TouchGFX UI into the SDK and use it on the AI280 hardware platform. Using this design, the USER can easily focus on the design of the end application. The below diagram depicts the overall design architecture of the final firmware application.
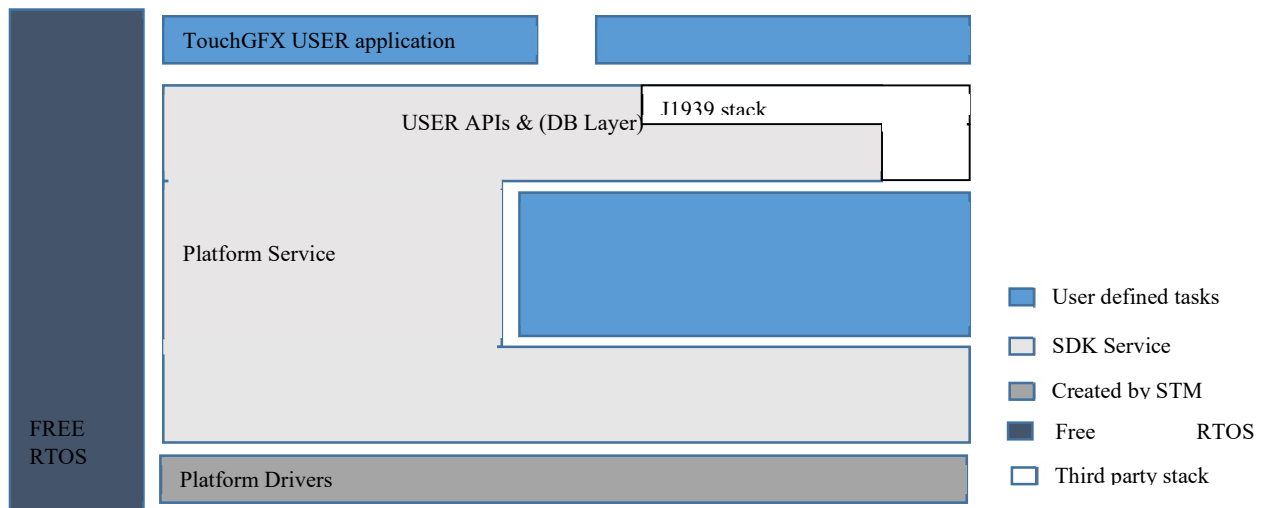
The user interacts with the SDK for the below functionalities,
1) Enable and disable the modules via the configuration file.
2) Provide default configuration for the properties of the modules as per their requirements.
3) Access Data Layer Data Base to get/set individual properties of the modules.

The user is free to enable any of the services or modules if needed to improve memory constraints. So, the first step is to enable the desired services in the configuration file and then configure the properties for each manager. The managers are directly connected to the OS and works automatically on the background, so the user does not need to worry about the usage or the error management. In the below sections a detailed description is provided on how each module of the SDK can be accessed by the user for the full filling their requirements.

5.3 SDK Boot flow

The diagram below depicts the SDK boot flow for the MAXAI280 platform.

```
                    ( 1. System Reset )
                            |
                            v
        +-------------------------------------------+
        |  2. System Init (Clock, SysTick, Start WDG) |
        +-------------------------------------------+
                            |
                            v
        +-------------------------------------------+
        |  3. Configure the Peripherals one by one based |
        |  on the .h configuration                  |
        +-------------------------------------------+
                            |
                            v
             +--------------------------------+
             |  4. Start the DB Layer         |
             +--------------------------------+
                            |
                            v
        +-------------------------------------------+
        |  5. Start the Platform Service Manager layer |
        |  one by one based on the .h configuration |
        +-------------------------------------------+
                            |
                            v
             +--------------------------------+
             |  6. Start the TouchGFX task    |
             +--------------------------------+
                            |
                            v
             +--------------------------------+
             |  7. Start the USER task(s)     |
             +--------------------------------+
                            |
                            v
             +--------------------------------+
             |  8. Start the FreeRTOS Scheduler |
             +--------------------------------+
```
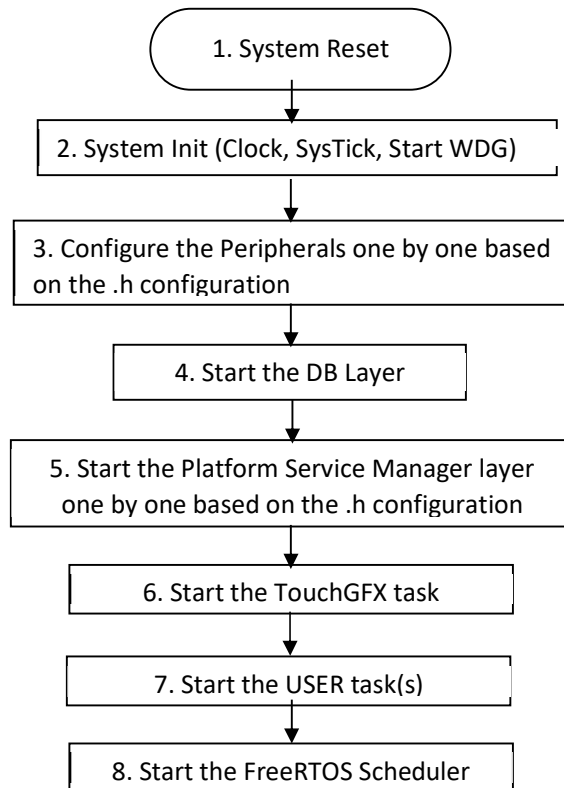
6   Application and SDK Interaction

The user can interact with the SDK to configure the individual modules of the SDK. This can happen either during power up configuration or during the run time configuration.

6.1   SDK Module default configuration

The user can configure the SDK by enabling and disabling individual modules. If a user enables or disables a particular module the entire stack is disabled for the particular module as shown in the below diagram.

For Example, If the USER disables the RTC driver, the RTC related (SDK USER APIs, RTC DB Layer, RTC Platform service and RTC Driver) codes will not be included in the final firmware image. That is, each device driver related SDK APIs, DB layer service, platform service and driver will be blocked with preprocessor MACROs (e.g., HAL_RTC_MODULE_ENABLED). This implementation will reduce the final image size.
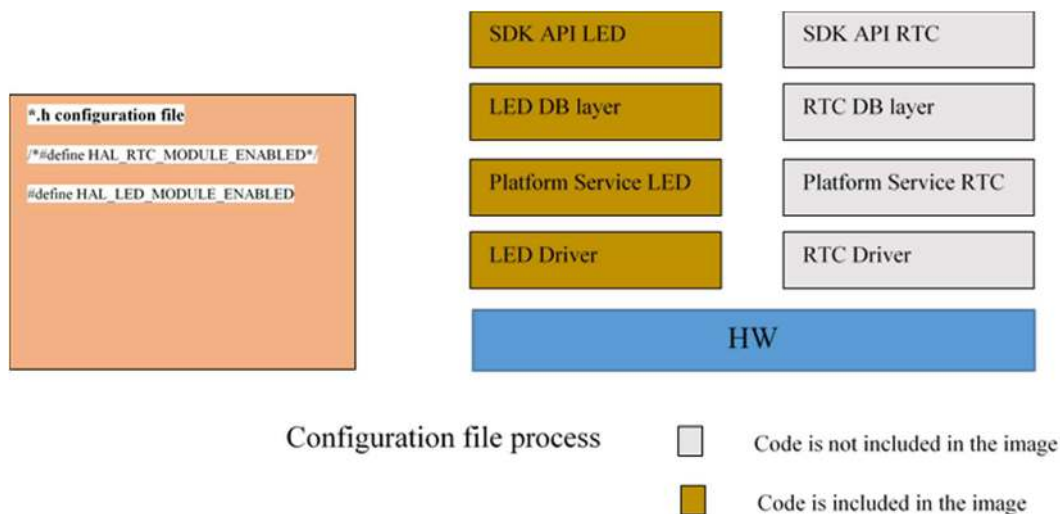
For example: - If user wants to enable the keypad module in the current build, user has to configure the below mentioned variable as PS_ENABLE.

 #define SDK_SERVICE_KEYPAD   PS_ENABLE

The user can find the configuration file in the source code in the below-mentioned path. Users can configure variables for any modules based on the requirements in the AI280_Config.h.

 AI280_Project\Core\Maximatecc\Inc\AI280_Config.h

Each module in the configuration file is differentiated with Headers/comments and users can easily find the SDK modules they are looking for.

For Example:- Keypad Module configuration

As shown in the above pic, the keypad module configurations are listed in the header file after the above comments. You can find similar comments for each module section in the configuration file.

The user can configure certain parameters per module which will impact the default configuration of the individual modules. This can be done by modifying the configuration file which is used by the SDK to configure the individual modules during the power up sequence. Once configured the modules continue the same configuration until it is changed by the user.

For Example: - The below parameter configures the keypad backlight. It can be configured as ON / OFF and when the device powers up the SDK Reads this configuration file and updates the keypad backlight accordingly. In the MAXAI280 the default configuration for this parameter is true and hence the keypad backlight is always ON after device powers up unless the application turns it of during runtime.

```
/*!
 * MACRO Supported
 *
 * Keypad backlight configuration state
 * 1: KEY_BACKLIGHT_ON /
 * 0: KEY_BACKLIGHT_OFF
 */
#define KEYPAD_BACKLIGHT_CFG_STATE              KEY_BACKLIGHT_ON
```

When the user modifies any configuration in the config.h file, the user will have to re - compile the source code and flash the updated binaries to the device and verify the changes.

6.2   Run Time Configuration:

The user can modify certain parameters per module during the run time to interact with the module and to execute their desired functionality. This can be achieved by using the Datalayer database API's to read/write into the DB entries for each module.

Data Layer DB will collect the data from the platform service / platform layers and update the data into the proper variable.

Data Layer DB will be accessed by using GET/SET APIs from the application. If the application needs any platform related data, it uses the GET/SET API of the DB layer with the proper platform field name/id.

The below platform service / platform layers are linked with the DB layer.

- Keypad
- Digital Output
- Configurable Inputs
- Power Mode
- LED
- Power Monitoring
- Bluetooth
- RTC
- SW Timer
- EEPROM
- Watchdog
- LCD
- Flash
- CAN
- J1939

The Data Layer DB will interact with the Platform service through platform service SDK GET/SET APIs.

6.3 DBLayer USER APIs

The USER shall access the DB layer field through the below set and get functions.

6.4 Function Name: GET_DL

S no	API Syntax	Parameter	Return value
1.	**GET_DL()** The user can use this API to retrieve the value of the data from the database. The field id is defined in the database.h file which identifies the data field we are interested in.	uint16_t dl_index , uint8_t *buf Value : We have to pass the Data ID for the data field we are looking to retrieve the data and then pass a buffer where the data to be written will be stored when the function call returns to the application.	0: Success 1: Failure

Below is the snippet of the description of the function,

```
/** @brief Set_DL
 *
```

```
 *   This function will Set the data to the platform service
 *
 *   @param  dl_index[IN]    : DB index value
 *           buf[IN]              : input buffer
 *
 *   @return  0 : FAILURE
 *            1 : SUCCESS
 *           -3 : NULL_POINTER
 *
 */
int8_t Set_DL(uint16_t dl_index, uint8_t *buf)
```

6.5 Function Name: SET_DL

S no	API Syntax	Parameter	Return value
1.	**SET_DL()** The user can use this API to store the value of the data from the database. The field id is defined in the database.h file which identifies the data field we are interested in.	uint16_t dl_index , uint8_t *buf Value: We have to pass the index ID for the data field we are looking to retrieve the data and then pass a buffer where the data to be written will be stored when the function call returns to the application.	0: Success 1: Failure

Below is the snippet of the description of the function,

```
/** @brief Get_DL
 *
 *   This function will get the data from the platform service
 *
 *   @param  dl_index[IN]    : DB index value
 *           buf[IN]      : input buffer
 *
 *   @return  0 : FAILURE
 *            1 : SUCCESS
 *           -3 : NULL_POINTER
 *
 */

int8_t Get_DL(uint16_t dl_index, uint8_t *buf)
```

For Example: -

If we want to get the current status of the keypad backlight and then toggle it, we can do so by using the below code snippet,

/* Get the current backlight status */

```
If (Get_DL(KEYPAD_BACKLIGHT , &state) == SUCCESS)
{
      /* If current state is ON, set it OFF */
      If ( state == KEY_BACKLIGHT_ON )
      {
            State = KEY_BACKLIGHT_OFF;
            Set_DL(KEYPAD_BACKLIGHT , &state);
      }
      /* If current state is OFF, set it ON */
      Else
      {
            State = KEY_BACKLIGHT_ON;
            Set_DL(KEYPAD_BACKLIGHT , &state);
      }

}
```

7   SDK Modules

Described below are the functionalities supported by each SDK module which can be used by the application developer to full fill their requirements.

7.1   Keypad module

The User would be able to use the below functionalities of the keypad module via the DB variables and configuration file.

7.1.1   Keypad module Enable/Disable

The SDK provides the user the ability to enable/disable the Keypad functionality by modifying the default configuration file. Please see section 7.1.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_ KEYPAD	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the Keypad module in the SDK **PS_DISABLE: -** Disables the Keypad module in the SDK

7.1.2   Keypad BackLight ON/OFF

The AI280 SDK supports default configuration of the Keypad backlight and this can be done by modifying the below parameter in the configuration file. Please see section 7.1.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	KEYPAD_BACK LIGHT_CFG_STA TE	1: KEY_BACKLIG HT_ON / 0: KEY_BACKLIG HT_OFF	KEY_BACKLIG HT_ON	User can configure the default state of the keypad Backlight to ON/OFF using this variable.

During runtime the user can read and modify the keypad backlight by reading and writing to the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
KEYPAD_BACKLIGHT	DBu8	READ/WRITE	1	ON/OFF	This field is used to turn ON/OFF the Keypad Backlight.

The sample code for set/get the Backlight using Key#1 is as below:

```
uint8_t state;

if (KEY1_SHORT_PRESS == val)
{
        state = KEY_BACKLIGHT_ON;
        Set_DL(KEYPAD_BACKLIGHT, &state);

}
/* Get the backlight state */
Get_DL(KEYPAD_BACKLIGHT, &state);
```

7.1.3 Keypad Time Out Configuration

The AI280 SDK user can configure the timeout value of keypress to differentiate between short press and long press. Short Press can be configured in the range (> 10ms && < 255ms). If the key is pressed longer than the short press timeout it would be considered as long press. This default configuration can be done in the AI280_config.h. Please see section 7.1.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SHORT_PRESS_TIMEOUT	Short Press (> 10 && < 255) Long press (> short press time)	10ms (Recommended Value)	The user can configure the timeout value of keypress to differentiate between short press and long press.

7.1.4 Keypad task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the keypad module in the configuration file. Please see section 7.1.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_KEYPAD_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.1.5 Keypad Keys Enable/Disable

The AI280 SDK supports four keys and the user can enable/disable each of the keys using the default configuration file.

Sr. No	Variables	Options	Default State	Description
1	CONF_KEYPAD_01_STATE	PS_ENABLE PS DISABLE	PS_ENABLE	**PS_ENABLE:** Enable the HW Key01 in the Keypad SDK platform service **PS_DISABLE:** Disable the HW Key01 in the Keypad SDK platform service
2	CONF_KEYPAD_02_STATE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE:** Enable the HW Key02 in the Keypad SDK platform service **PS_DISABLE:** Disable the HW Key02 in the Keypad SDK platform service
3	CONF_KEYPAD_03_STATE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE:** Enable the HW Key03 in the Keypad SDK platform service **PS_DISABLE:** Disable the HW Key03 in the Keypad SDK platform service
4	CONF_KEYPAD_04_STATE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE:** Enable the HW Key04 in the Keypad SDK platform service **PS_DISABLE:** Disable the HW Key04 in the Keypad SDK platform

				service

7.1.6 Keypad Keys read status

The AI280 SDK user can then read the Key status variables to know if the keys are active or inactive. This DB entry has to be read first for receiving the keypress event. If the KEY_STATUS_01 is KP_ACTIVE then the USER has to read the KEY_PRESS_01 DB variable to check the state of the key press.

Field ID	Data Type	Permission	Size	Description	Comments
KEY_STATUS_01	DBu8	READ	1	KP_ACTIVE/KP_IN ACTIVE	This field is used to Read the status of the Key#1.
KEY_STATUS_02	DBu8	READ	1	KP_ACTIVE/KP_IN ACTIVE	This field is used to Read the status of the Key#2
KEY_STATUS_03	DBu8	READ	1	KP_ACTIVE/KP_IN ACTIVE	This field is used to Read the status of the Key3
KEY_STATUS_04	DBu8	READ	1	KP_ACTIVE/KP_IN ACTIVE	This field is used to Read the status of the Key#4
KEY_PRESS_01	DBu8	READ/WRI TE	1	INACTIVE/SHORT_ PRESS/LONG_PRE SS/CONTINUOUS_ PRESSED	This field detects the type of KeyPress for Key#1.
KEY_PRESS_02	DBu8	READ/WRI TE	1	INACTIVE/SHORT_ PRESS/LONG_PRE SS/CONTINUOUS_ PRESSED	This field detects the type of KeyPress for Key#2.
KEY_PRESS_03	DBu8	READ/WRI TE	1	INACTIVE/SHORT_ PRESS/LONG_PRE SS/CONTINUOUS_ PRESSED	This field detects the type of KeyPress for Key#3.
KEY_PRESS_04	DBu8	READ/WRI TE	1	INACTIVE/SHORT_ PRESS/LONG_PRE SS/CONTINUOUS_ PRESSED	This field detects the type of KeyPress for Key#4.

The user can read the key state (short press/ long press/ continuous / inactive) by continuously monitoring the below DB variables. Once the USER gets any one of the Keypress events (SHORT_PRESS/LONG_PRESS), the USER has to ACK the key press (KEY_PRESS_01) with the value 1. For CONTINUOUS state is not necessary to acknowledge the state.

Below is a code snippet for the keypress read and acknowledge usecase.

```c
uint8_t key_status = 0;
#if(CONF_KEYPAD_01_STATE == PS_ENABLE)
    /* Read the Key Status from the DB */
    res = Get_DL(KEY_STATUS_01, &key_status);
    /* Proceed only if the Key # 1 is active */
    if (KEY_INACTIVE != key_status)
    {
        res = Get_DL(KEY_PRESS_01, &key_status);
        if (SHORT_PRESSED == key_status)
        {
            key_status = 1;
            /* ACK the Keypress */
            res = Set_DL(KEY_PRESS_01, &key_status);
        }
        else
        if (LONG_PRESSED == key_status)
        {
            key_status = 1;
            /* ACK the Keypress */
            res = Set_DL(KEY_PRESS_01, &key_status);
            modelListener->keyUpdate(KEY1_LONG_PRESS);
        }
        else
        if (CONTINUOUS_PRESSED == key_status)
        {
            modelListener->keyUpdate(KEY1_LONG_PRESS);
        }
    }
#endif      //KEYPAD_01
```

7.1.7 Keypad sample Configuration

```c
* Keypad Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_KEYPAD                          PS_ENABLE
#if (SDK_SERVICE_KEYPAD == PS_ENABLE)
/*!
 * Keypad Task Priority
 * osPriorityNone         =  0,
```

```
 * osPriorityIdle          =  1,
 * osPriorityLow           =  8,
 * osPriorityLow1          =  8+1,
 *           ,,                            ,,
 *           ,,                            ,,
 * osPriorityISR           = 56,
 * osPriorityError         = -1,
 * osPriorityReserved      = 0x7FFFFFFF
 */
#define PS_KPD_TASK_PRIORITY                            osPriorityIdle
/*!
 * MACRO Supported
 *
 * Keypad backlight configuration state
 * 1: KEY_BACKLIGHT_ON /
 * 0: KEY_BACKLIGHT_OFF
 */
#define KEYPAD_BACKLIGHT_CFG_STATE              KEY_BACKLIGHT_ON
/*!
 * SHORT Press timeout (Millisecond)
 */
#define SHORT_PRESS_TIMEOUT        10
/*!
 * MACOR Supported
 *
 * Keypad configuration state
 * 1: PS_ENABLE /
 * 0: PS_DISABLE
 */
#define CONF_KEYPAD_01_STATE                  PS_ENABLE
#define CONF_KEYPAD_02_STATE                  PS_ENABLE
#define CONF_KEYPAD_03_STATE                  PS_ENABLE
#define CONF_KEYPAD_04_STATE                  PS_ENABLE
#endif      //SDK_SERVICE_KEYPAD
```

7.2 Digital Output Module

The User would be able to use the below functionalities of the digital output module via the DB variables and configuration file.

7.2.1 Digital Output module Enable/Disable

The SDK provides the user the ability to enable/disable the Digital Output functionality by modifying the default configuration file. Please see section 7.2.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_DIGITAL_OUTPUT	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the Digital Output module in the SDK **PS_DISABLE: -** Disables the Digital Output module in the SDK

7.2.2 Digital Output Configuration

The AI280 SDK supports default configuration of the digital output status and this can be done by modifying the below parameter in the configuration file. Please see section 7.2.6 for sample Configuration.

Sr. No	Variables	Options	Default State	Description
1	DIGITAL_OUTPUT_CFG_01	CONF_OPEN_DRIVE_DRIVER/ CONF_LOW_SIDE_DRIVER/ CONF_HIGH_SIDE_DRIVER	CONF_OPEN_DRIVE_DRIVER	User can select the Digital output 01 configuration as open drive driver when the digital output status is OFF User can select the Digital output configuration as low side driver. User can select the Digital output configuration as high side driver.
2	DIGITAL_OUTPUT_CFG_02	CONF_OPEN_DRIVE_DRIVER/ CONF_LOW_SIDE_DRIVER/ CONF_HIGH_SIDE_DRIVER	CONF_OPEN_DRIVE_DRIVER	User can select the Digital output 02 configuration as open drive driver when the digital output status is OFF User can select the Digital output configuration as low side driver. User can select the Digital output configuration as high side driver.

3	DIGITAL_OUTPUT_CFG_03	CONF_OPEN_DRIVE_DRIVER/ CONF_LOW_SIDE_DRIVER/ CONF_HIGH_SIDE_DRIVER	CONF_OPEN_DRIVE_DRIVER	User can select the Digital output 03 configuration as open drive driver when the digital output status is OFF User can select the Digital output configuration as low side driver. User can select the Digital output configuration as high side driver.
4	DIGITAL_OUTPUT_CFG_04	CONF_OPEN_DRIVE_DRIVER/ CONF_LOW_SIDE_DRIVER/ CONF_HIGH_SIDE_DRIVER	CONF_OPEN_DRIVE_DRIVER	User can select the Digital output 04 configuration as open drive driver when the digital output status is OFF User can select the Digital output configuration as low side driver. User can select the Digital output configuration as high side driver.

The user can do the same configuration during the runtime via the DB variables and configuration file as shown below,

Field ID	Data Type	Permission	Size	Description	Comments
CF_DIGITAL_OUTPUT_01_CFG	DBu8	READ/WRITE	1	OPEN_DRIVE / HIGH_SIDE_DRIVER / LOW_SIDE_DRIVER	This field is used to configure the digital output 01 as high side, low side or open drive. The field is also used to enable/ disable the Digital Output. The status of the field can also be read back once enabled.
CF_DIGITAL_OUTPUT_02_CFG	DBu8	READ/WRITE	1	OPEN_DRIVE / HIGH_SIDE_DRIVER / LOW_SIDE_DRIVER	This field is used to configure the digital output 02 as high side, low side or open drive. The field is also used to enable/ disable the Digital Output. The status of the field can also be read back once enabled.

CF_DIGITAL_ OUTPUT_03_C FG	DBu8	READ/WRI TE	1	OPEN_DRIVE / HIGH_SIDE_DRIVE R / LOW_SIDE_DRIVE R	This field is used to configure the digital output 03 as high side, low side or open drive. The field is also used to enable/ disable the Digital Output. The status of the field can also be read back once enabled.
CF_DIGITAL_ OUTPUT_04_C FG	DBu8	READ/WRI TE	1	OPEN_DRIVE / HIGH_SIDE_DRIVE R / LOW_SIDE_DRIVE R	This field is used to configure the digital output 04 as high side, low side or open drive. The field is also used to enable/ disable the Digital Output. The status of the field can also be read back once enabled.

Below is the sample code for accessing the Digital output configuration DB variables.

```
uint8_t state;
/* Read the Digital Output configuration */
Get_DL(DIGITAL_OUTPUT_01_CFG, &state);
if(state == CONF_LOW_SIDE_DRIVER)
{
      state = CONF_HIGH_SIDE_DRIVER;
      /* Set the High side Digital Output */
      Set_DL(DIGITAL_OUTPUT_01_CFG, &state);
}
Get_DL(DIGITAL_OUTPUT_01_CFG, &state);
if(state == CONF_HIGH_SIDE_DRIVER)
{
      state = CONF_LOW_SIDE_DRIVER;
      /* Set the Low side Digital Output */
      Set_DL(DIGITAL_OUTPUT_01_CFG, &state);
}
```

7.2.3   Digital Output ON/OFF

The AI280 SDK user can turn ON / OFF the digital output during runtime. To do so he can use the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
DIGITAL_OUTPUT_01_STATE	DBu8	READ/WRITE	1	CONF_DIGITAL_OUTPUT_ON/CONF_DIGITAL_OUTPUT_OFF	This field is used to turn ON or OFF the digital output 01. The status of the field can also be read back once to get the current status of the pins.
DIGITAL_OUTPUT_02_STATE	DBu8	READ/WRITE	1	CONF_DIGITAL_OUTPUT_ON/CONF_DIGITAL_OUTPUT_OFF	This field is used to turn ON or OFF the digital output 02. The status of the field can also be read back once to get the current status of the pins.
DIGITAL_OUTPUT_03_STATE	DBu8	READ/WRITE	1	CONF_DIGITAL_OUTPUT_ON/CONF_DIGITAL_OUTPUT_OFF	This field is used to turn ON or OFF the digital output 03. The status of the field can also be read back once to get the current status of the pins.
DIGITAL_OUTPUT_04_STATE	DBu8	READ/WRITE	1	CONF_DIGITAL_OUTPUT_ON/CONF_DIGITAL_OUTPUT_OFF	This field is used to turn ON or OFF the digital output 04. The status of the field can also be read back once to get the current status of the pins.

Below is the sample code for turning ON/OFF the Digital output DB variable.

```
state = CONF_DIGITAL_OUTPUT_ON;
/* Set the open drive Digital Output */
Set_DL(DIGITAL_OUTPUT_01_STATE, &state);
state = CONF_DIGITAL_OUTPUT_OFF;
/* Set the open drive Digital Output */
Set_DL(DIGITAL_OUTPUT_01_STATE, &state);
```

7.2.4 Digital Output Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.2.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_DIO_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the values in the Database.

7.2.5 Digital Output task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the digital output module in the configuration file. Please see section 7.2.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_DIO_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.2.6 Digital Output Sample Configuration

```
/*!
 * DIO Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_DIGITAL_OUTPUT              PS_ENABLE
#if (SDK_SERVICE_DIGITAL_OUTPUT == PS_ENABLE)
/*!
 * DIO Task Periodicity 100ms
 */
```

```
#define PS_DIO_TASK_TIMEOUT                          100
/*!
 * DIO Task Priority
 * osPriorityNone          =  0,
 * osPriorityIdle          =  1,
 * osPriorityLow           =  8,
 * osPriorityLow1          =  8+1,
 *           ,,                          ,,
 *           ,,                          ,,
 * osPriorityISR           = 56,
 * osPriorityError         = -1,
 * osPriorityReserved      = 0x7FFFFFFF
 */
#define PS_DIO_TASK_PRIORITY                         osPriorityIdle
/*!
 * Select the DIGITAL_OUTPUT_CFG 00 : CONF_LOW_SIDE_DRIVER
 *                              01 : CONF_HIGH_SIDE_DRIVER
 *                              02 : CONF_OPEN_DRIVE_DRIVER
 */

#define DIGITAL_OUTPUT_CFG_01                 CONF_OPEN_DRIVE_DRIVER
#define DIGITAL_OUTPUT_CFG_02                 CONF_LOW_SIDE_DRIVER
#define DIGITAL_OUTPUT_CFG_03                 CONF_HIGH_SIDE_DRIVER
#define DIGITAL_OUTPUT_CFG_04                 CONF_HIGH_SIDE_DRIVER

#endif     //SDK_SERVICE_DIGITAL_OUTPUT
```

7.3 Configurable Inputs Module

The User would be able to use the below functionalities of the keypad module via the DB variables and configuration file.

7.3.1 Configurable Inputs module Enable/Disable

The SDK provides the user the ability to enable/disable the configurable functionality by modifying the default configuration file. Please see section 7.3.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_CFG_INPUT	PS_ENABLE/ PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the Configurable input module in the SDK **PS_DISABLE: -** Disables the configurable input module in the SDK

7.3.2 Configurable Inputs task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the configurable inputs module in the configuration file. Please see section 7.3.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_CFG_INPUT_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.3.3 Configurable Inputs Task Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware for the configured inputs and update it in the DB so that when the user reads the DB, he will receive the latest updated data. This default configuration can be done in the AI280_config.h. Please see section 7.3.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_CFG_INPUT_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.3.4 Configurable Inputs – Configure the number of samples

The AI280 SDK user can configure the number of samples to be considered for the average calculation of the readings from the hardware before it is updated to the database. This would improve the accuracy of the data updated in the DB. This default configuration can be done in the AI280_config.h. Please see section 7.3.6 for sample configuration.

Sr.No	Variables	Options	Default Value	Description
1	CONF_INPUT_01_NUMB_SAMPLES	MIN VALUE: 1 MAX VALUE: 255	1	User can set the number of sample values to be considered for the average calculation for configurable input 1
2	CONF_INPUT_02_NUMB_SAMPLES	MIN VALUE: 1 MAX VALUE: 255	1	User can set the number of sample values to be considered for the average calculation for configurable input 2
3	CONF_INPUT_03_NUMB_SAMPLES	MIN VALUE: 1 MAX VALUE: 255	1	User can set the number of sample values to be considered for the average calculation for configurable input 3
4	CONF_INPUT_04_NUMB_SAMPLES	MIN VALUE: 1 MAX VALUE: 255	1	User can set the number of sample values to be considered for the average calculation for configurable input 4
5	CONF_INPUT_05_NUMB_SAMPLES	MIN VALUE: 1 MAX VALUE: 255	1	User can set the number of sample values to be considered for the average calculation for configurable input 5

The user can dynamically set/get the number of samples to be considered for the average calculation for configurable input during the run time. To do so, the user can use the below DB variables.

Field ID	Data Type	Permission	Size	Options	Description
CFG_INPUT_01_NUMB_SAMPLES	DBu8	READ/WRITE	1	1 -255 range	This field is used to set number of samples to get an average value. The field is also used to read the number of samples set.
CFG_INPUT_02_NUMB_SAMPLES	DBu8	READ/WRITE	1	1 -255 range	This field is used to set number of samples to get an average value. The field is also used to read the number of samples set.
CFG_INPUT_03_NUMB_SAMPLES	DBu8	READ/WRITE	1	1 -255 range	This field is use to set number of samples to get an average value. The field is also used to read the number of samples set.
CFG_INPUT_04_NUMB_SAMPLES	DBu8	READ/WRITE	1	1 -255 range	This field is use to set number of samples to get an average value. The field is also used to read the number of samples set.
CFG_INPUT_05_NUMB_SAMPLES	DBu8	READ/WRITE	1	1 -255 range	This field is use to set number of samples to get an average value. The field is also used to read the number of samples set.

The below code snippet shows how the sample configuration can be altered from the application code,

```
if(update_sample)
{
        int num_sample = 10;
        /* Set the sample count to 10 for configurable input 1 */
        res  = DL_Set(CFG_INPUT_01_NUMB_SAMPLES,&num_sample);
}
```

7.3.5 Configurable Inputs configuration

The AI280 SDK user can configure the 5 available configurable inputs as per his desired requirement as supported by the platform. To do so, he can configure the below parameters in the configuration file. This default configuration can be done in the AI280_config.h. Please see section 7.3.6 for sample configuration.

Sr.No	Variables	Default Values	Default State	Description
1	CONF_INPUT_TYPE_01	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V,	CI_DIGITAL_STB	User can configure the configurable input type 1 for input voltage, input frequency, input resistance, Digital STB and digital STG
2	CONF_INPUT_TYPE_02	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V,	CI_DIGITAL_STG	User can configure the configurable input type 2 for input voltage, input frequency, input resistance, Digital STB and digital STG
3	CONF_INPUT_TYPE_03	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V,	CI_INPUT_RESISTANCE	User can configure the configurable input type 3 for input voltage, input frequency, input resistance, Digital STB and digital STG
4	CONF_INPUT_TYPE_04	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE,	CI_INPUT_RESISTANCE	User can configure the configurable input type 4 for input voltage, input frequency, input resistance, Digital STB and digital STG

		CI_DIGITAL_ST G, CI_DIGITAL_ST B, CI_INPUT_VOL TAGE_HIGH, CI_INPUT_VOL TAGE_LOW_6V,		
5	CONF_INPUT_T YPE_05	CI_INPUT_FRE QUENCY, CI_INPUT_RESI STANCE, CI_DIGITAL_ST G, CI_DIGITAL_ST B, CI_INPUT_VOL TAGE_HIGH, CI_INPUT_VOL TAGE_LOW_6V,	CI_INPUT_RES ISTANCE	User can configure the configurable input type 5 for input voltage, input frequency, input resistance, Digital STB and digital STG

The user also has the ability to run time configure the 5 available configurable inputs as per his desired requirement as supported by the platform. To do so the user can use the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
CFG_INPUT_01_TYP E	DBu8	READ/WRI TE	1	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIG H, CI_INPUT_VOLTAGE_LO W_6V	This field is used to Configure the input to any one of the types suggested. The field is also used to turn off the input. The status of the CFG_Input#01 type can be read using this field.
CFG_INPUT_02_TYP E	DBu8	READ/WRI TE	1	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG,	This field is used to Configure the input to any one of the types

				CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V	suggested. The field is also used to turn off the input. The status of the CFG_Input#02 type can be read using this field.
CFG_INPUT_03_TYPE	DBu8	READ/WRITE	1	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V, CI_INPUT_OFF,	This field is used to Configure the input to any one of the types suggested. The field is also used to turn off the input. The status of the CFG_Input#03 type can be read using this field.
CFG_INPUT_04_TYPE	DBu8	READ/WRITE	1	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V	This field is used to Configure the input to any one of the types suggested. The field is also used to turn off the input. The status of the CFG_Input#04 type can be read using this field.
CFG_INPUT_05_TYPE	DBu8	READ/WRITE	1	CI_INPUT_FREQUENCY, CI_INPUT_RESISTANCE, CI_DIGITAL_STG, CI_DIGITAL_STB, CI_INPUT_VOLTAGE_HIGH, CI_INPUT_VOLTAGE_LOW_6V	This field is used to Configure the input to any one of the types suggested. The field is also used to turn off the input. The status of the CFG_Input#05 type can be read using this field.

The below code snippet shows how the Configurable inputs type can be configured during the run time.

```
uint8_t input1_type;
```

/* Read the current configuration for Configurable input 1 */

```
Get_DL(CFG_INPUT_01_TYPE , &input1_type);
/* If it is currently configured as frequency, change it to resistance */
if(CI_INPUT_FREQUENCY == input1_type)
{
      input1_type  = CI_INPUT_RESISTANCE;
      Set_DL(CFG_INPUT_01_TYPE , &input1_type);
}
```

Once the user configures the various configurable inputs the platform service will read the data from the hardware every time the task time out occurs and update the below DB variables. The user can then access the same by using the DL_get/DL_set API's.

For example, if he has configured the Configurable input 1 as CI_INPUT_VOLTAGE_HIGH then the user will have to read the CFG_INPUT_01_VOLTAGE_32V DB entry to read the voltage value in milli volts.

Field ID	Data Type	Permission	Size	Options	Description
CFG_INPUT_01_FREQUENCY	DBu32	READ	4	10Hz-20000Hz range	This field is used to read the frequency of CFG_Input#01. The frequency is read in hertz.
CFG_INPUT_01_VOLTAGE_32V	DBu16	READ	2	0-32000 range	This field is used to read the high voltage of CFG_Input#01. The voltage is read in milli-volts
CFG_INPUT_01_VOLTAGE_LOW_6V	DBu16	READ	2	0-6V	This field is used to read the low voltage of CFG_Input#01. The voltage is read in volts
CFG_INPUT_01_RESISTANCE	DBu16	READ	2	1ohm – 500ohm range	This field is used to read the

					Resistance of CFG_Input#01. Resistance is read in ohms
CFG_INPUT_01_DIGITAL_STG	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#01. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_01_DIGITAL_STB	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#01. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_02_FREQUENCY	DBu32	READ	4	10Hz-20000Hz range	This field is used to read the frequency of CFG_Input#02. The frequency is read in milli-hertz.
CFG_INPUT_02_VOLTAGE_32V	DBu16	READ	2	0-32000 range	This field is used to read the voltage of CFG_Input#02. The voltage is read in milli-volts
CFG_INPUT_02_VOLTAGE_LOW_6V	DBu16	READ	2	0-6V	This field is used to read the low voltage of CFG_Input#02. The voltage is read in volts

CFG_INPUT_02_RESISTANCE	DBu16	READ	2	1ohm – 500ohm range	This field is used to read the Resistance of CFG_Input#02. Resistance is read in ohms
CFG_INPUT_02_DIGITAL_STG	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#02. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_02_DIGITAL_STB	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#02. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_03_FREQUENCY	DBu32	READ	4	0-20000 range	This field is used to read the frequency of CFG_Input#03. The frequency is read in milli-hertz.
CFG_INPUT_03_VOLTAGE_32V	DBu16	READ	2	0-32000 range	This field is used to read the voltage of CFG_Input#03. The voltage is read in milli-volts
CFG_INPUT_03_VOLTAGE_LOW_6V	DBu16	READ	2	0-6V	This field is used to read the low voltage of CFG_Input#03

					The voltage is read in volts
CFG_INPUT_03_RESISTANCE	DBu16	READ	2	1ohm – 500ohm range	This field is used to read the Resistance of CFG_Input#03. Resistance is read in ohms
CFG_INPUT_03_DIGITAL_STG	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#03. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_03_DIGITAL_STB	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#03. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_04_FREQUENCY	DBu32	READ	4	0-20000 range	This filed is used to read the frequency of CFG_Input#04. The frequency is read in milli-hertz.
CFG_INPUT_04_VOLTAGE_32V	DBu16	READ	2	0-32000 range	This field is used to read the voltage of CFG_Input#04. The voltage is read in milli-volts

CFG_INPUT_04_VOLTAGE_LOW_6V	DBu16	READ	2	0-6V	This field is used to read the low voltage of CFG_Input#04. The voltage is read in volts
CFG_INPUT_04_RESISTANCE	DBu16	READ	2	1ohm – 500ohm range	This filed is used to read the Resistance of CFG_Input#04. Resistance is read in ohms
CFG_INPUT_04_DIGITAL_STG	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#04. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_04_DIGITAL_STB	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#04. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_05_FREQUENCY	DBu32	READ	4	0-20000 range	This filed is used to read the frequency of CFG_Input#05. The frequency is read in milli-hertz.
CFG_INPUT_05_VOLTAGE_32V	DBu16	READ	2	0-32000 range	This field is used to read the voltage of CFG_Input#05.

					The voltage is read in milli-volts
CFG_INPUT_05_VOLTAGE_LOW_6V	DBu16	READ	2	0-6V	This field is used to read the low voltage of CFG_Input#05. The voltage is read in volts
CFG_INPUT_05_RESISTANCE	DBu16	READ	2	1ohm – 500ohm range	This filed is used to read the Resistance of CFG_Input#05. Resistance is read in ohms
CFG_INPUT_05_DIGITAL_STG	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#05. If TRUE = digital active and FALSE = digital inactive
CFG_INPUT_05_DIGITAL_STB	DBu8	READ	1	TRUE / FALSE	This field is used to read the Digital Input level of CFG_Input#05. If TRUE = digital active and FALSE = digital inactive

The below code sample shows the configuration values read from CI1 during runtime,

```
uint8_t input1_type;
uint8_t val;
if(CI_INPUT_FREQUENCY == input1_type)
{
    val = 0;
    Get_DL(CFG_INPUT_01_FREQUENCY , (uint8_t *)&val);
}
else
if(CI_INPUT_VOLTAGE_HIGH == input1_type)
```

```
        {
                val = 0;
                Get_DL(CFG_INPUT_01_VOLTAGE_32V , (uint8_t *)&val);
        }
        else
        if(CI_INPUT_VOLTAGE_LOW_6V == input1_type)
        {
                val = 0;
                Get_DL(CFG_INPUT_01_VOLTAGE_LOW_6V , (uint8_t *)&val);
        }
        else
        if(CI_INPUT_RESISTANCE == input1_type)
        {
                val = 0;
                Get_DL(CFG_INPUT_01_RESISTANCE , (uint8_t *)&val);
        }
        else
        if(CI_DIGITAL_STG == input1_type)
        {
                val = 0;
                Get_DL(CFG_INPUT_01_DIGITAL_STG , (uint8_t *)&val);
        }
        else
        if(CI_DIGITAL_STB == input1_type)
        {
                val = 0;
                Get_DL(CFG_INPUT_01_DIGITAL_STB , (uint8_t *)&val);
        }
```

7.3.6 Configurable Inputs Default Configuration

```
/*!
 * Config input Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE)
Macros
 */
#define SDK_SERVICE_CFG_INPUT                        PS_ENABLE
#if (SDK_SERVICE_CFG_INPUT == PS_ENABLE)
/*!
 * Config_input Task Periodicity          100ms
 */
#define PS_CFG_INPUT_TASK_TIMEOUT             100
/*!
 * CFG Input Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
```

```
 * osPriorityLow          =  8,
 * osPriorityLow1         =  8+1,
 *          ,,                        ,,
 *          ,,                        ,,
 * osPriorityISR           = 56,
 * osPriorityError         = -1,
 * osPriorityReserved      = 0x7FFFFFFF
 */
#define PS_CFG_INPUT_TASK_PRIORITY               osPriorityIdle
#define CONF_INPUT_TYPE_01                       CI_DIGITAL_STB
#define CONF_INPUT_TYPE_02                       CI_DIGITAL_STG
#define CONF_INPUT_TYPE_03                       CI_INPUT_RESISTANCE
#define CONF_INPUT_TYPE_04                       CI_INPUT_RESISTANCE
#define CONF_INPUT_TYPE_05                       CI_INPUT_RESISTANCE
#define CONF_INPUT_01_NUMB_SAMPLES               1
#define CONF_INPUT_02_NUMB_SAMPLES               1
#define CONF_INPUT_03_NUMB_SAMPLES               1
#define CONF_INPUT_04_NUMB_SAMPLES               1
#define CONF_INPUT_05_NUMB_SAMPLES               1
#endif      //SDK_SERVICE_CFG_INPUT
#endif      //SDK_SERVICE_CFG_INPUT
```

7.4 LED Module

The User would be able to use the below functionalities of the LED module via the DB variables and configuration file.

7.4.1 LED module Enable/Disable

The SDK provides the ability to the user to enable/disable the LED functionality by modifying the default configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_LED	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the LED module in the SDK **PS_DISABLE: -** Disables the LED module in the SDK

7.4.2 LED Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_LED_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.4.3 LED task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the LED module in the configuration file. Please see section 6.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_LED_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR,	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

		osPriorityError, osPriorityReserved		

7.4.4   Maximum LED'S Configuration

The AI280 SDK supports a maximum of 2 LED's and the user has the ability to configure the MAX LEDs supported by the device in the configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	MAX_LED_NUM	1 or 2	2	User can operate maximum 2 LED

7.4.5   Configuring RED LED Enable/Disable

The SDK provides the ability to the user to enable/disable the RED LED functionality by modifying the default configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	CONF_RED_LED_PS_STATE	PS_ENABLE/ PS_DISABLE	PS_ENABLE	User can enable/Disable the RED LED

7.4.6   Configuring RED LED State

The AI280 SDK supports the user to configure the default state of the RED LED and this can be done by modifying the below parameter in the configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	CONF_RED_LED_STATE	LED_CONF_ON/ LED_CONF_OFF	LED_CONF_ON	User can turn ON/OFF the RED LED

During runtime, the user can read and modify the RED LED state by reading and writing to the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
LED_RED_STATE	DBu8	READ/WRITE	1	ON / OFF	This field is used to enable/disable the LED_RED.

| | | | | | ON = Light up the LED, OFF = Turn off the LED. |

Below code snippet shows how the RED LED can be read and written into the DB.

```
uint8_t state;
#if (SDK_SERVICE_LED == PS_ENABLE)
#if(CONF_RED_LED_PS_STATE == PS_ENABLE)
        /* Get the RED LED Status from the DB */
        Get_DL(LED_RED_STATE, &state);
        if (LED_ON == state)
        {
                /* LED is on */
                state = LED_OFF;
                /* Set the RED LED Status from the DB */
                Set_DL(LED_RED_STATE, &state);
        }
        else
        {
                /* LED is OFF; */
        }
#endif
        #endif
```

7.4.7   Configuring RED LED blinking

The AI280 SDK supports the user to configure the RED LED blinking time period in milli second and this can be done by modifying the below parameter in the configuration file. Please see section 7.4.11 for sample configuration.

The value configured here is multiplied by 250ms to get the blinking period. So, if we have set a value of 2 here, between every blink there will be a (2*250ms = 500ms) time lag.

If this value is set as 0 then the blinking is disabled.

Sr. No	Variables	Options	Default State	Description
1	CONF_RED_LED_BLINKING_MS	0-1000	1 (Recommended Value)	User can change the Blink time period for the RED LED

During runtime, the user can read and modify the RED LED state and RED LED blinking time period by reading and writing to the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
LED_RED_BLINKING	DBu16	READ/WRITE	2	0-65535 range	This field is used to set and read back the Blinking period in milli seconds.

Below code snippet shows how the RED LED can be read from the DB.

```
     uint8_t state;
     uint16_t blink;
#if (SDK_SERVICE_LED == PS_ENABLE)
#if(CONF_RED_LED_PS_STATE == PS_ENABLE)
          blink = 0;
     /* Get the RED LED blink period from the DB */
     Get_DL(LED_RED_BLINKING, (uint8_t *)&blink);
     state = LED_ON;
     /* Set the RED LED Status to the DB */
     Set_DL(LED_RED_STATE, &state);
#endif
       #endif
```

7.4.8   Configuring AMB LED Enable/Disable

The SDK provides the user the ability to enable/disable the AMBER LED functionality by modifying the default configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	CONF_AMB_LED_PS_STATE	PS_ENABLE/ PS_DISABLE	PS_ENABLE	User can enable/Disable the AMB LED

7.4.9   Configuring AMB LED State

The AI280 SDK supports the user to configure the default state of the AMBER LED and this can be done by modifying the below parameter in the configuration file. Please see section 7.4.11 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	CONF_AMB_LED_STATE	LED_CONF_ON/ LED_CONF_OFF	LED_CONF_ON	User can turn ON/OFF the AMB LED

During runtime, the user can read and modify the AMB LED state by reading and writing to the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
LED_AMB_STATE	DBu8	READ/WRITE	1	ON / OFF	This field is used to enable/disable the LED_AMB. ON = Light up the LED, OFF = Turn off the LED.

Below code snippet shows how the AMB LED can be read and written into the DB.

```
        uint8_t state;
#if (SDK_SERVICE_LED == PS_ENABLE)
#if(CONF_AMB_LED_PS_STATE == PS_ENABLE)
        /* Get the RED LED Status from the DB */
        Get_DL(LED_AMB_STATE, &state);
        if (LED_ON == state)
        {
                /* LED is on */
                state = LED_OFF;
                /* Get the RED LED Status from the DB */
                Set_DL(LED_AMB_STATE, &state);
        }
        else
        {
                /* LED is OFF; */
        }
#endif
        #endif
```

7.4.10 Configuring AMB LED blinking

The AI280 SDK supports the user to configure the AMBER LED blinking time period in milli second and this can be done by modifying the below parameter in the configuration file. Please see section 7.4.11 for sample configuration.

The value configured here is multiplied by 250ms to get the blinking period. So, if we have set a value of 2 here, between every blink there will be a (2*250ms = 500ms) time lag. If this value is set as 0 then the blinking is disabled.

Sr. No	Variables	Options	Default State	Description
1	CONF_AMB_LED_BLINKING_MS	0-1000	1 (Recommended Value)	User can change the Blink time period for the AMB LED

During runtime, the user can read and modify the AMBER LED state and AMBER LED blinking by reading and writing to the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
LED_AMB_BLINKING	DBu16	READ/WRITE	2	0-65535 range	This field is used to set and read back the Blinking period in milli seconds.

Below code snippet shows how the AMB LED can be read and written into the DB.

```
        uint8_t state;
        uint16_t blink;
#if (SDK_SERVICE_LED == PS_ENABLE)
#if(CONF_AMB_LED_PS_STATE == PS_ENABLE)
        blink = 0;
        /* Get the AMB LED blink period from the DB */
        Get_DL(LED_AMB_BLINKING, (uint8_t *)&blink);
        blink = 2;
        /* Set the AMB LED blink period from the DB */
        Set_DL(LED_AMB_BLINKING, (uint8_t *)&blink);
        state = LED_ON;
        /* Set the RED LED Status to the DB */
        Set_DL(LED_AMB_STATE, &state);
#endif
        #endif
```

7.4.11 LED Sample Configuration

```
/*!
 * LED Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_LED                              PS_ENABLE
#if (SDK_SERVICE_LED == PS_ENABLE)
/*!
 * LED Task Periodicity      100ms
 */
#define PS_LED_TASK_TIMEOUT                          100
/*!
 * LED Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *          ,,                           ,,
 *          ,,                           ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
```

```
 * osPriorityReserved      = 0x7FFFFFFF
 */
#define PS_LED_TASK_PRIORITY                        osPriorityIdle
/*!
 * Maximum Number of LED required for this application
 *
 * MACOR Supported
 *
 * MAX_LED_NUM  : This hardware support maximum of 2 LED's
 *
 * CONF_xx_LED_PS_STATE                      PS_ENABLE
 *                                           PS_DISABLE
 *
 * CONF_xx_LED_STATE                         LED_CONF_OFF
 *                                           LED_CONF_ON
 *
 * CONF_xx_LED_BLINKING_MS                   <0-65535>
 *
 */
#define MAX_LED_NUM                             2
#define CONF_RED_LED_PS_STATE                 PS_ENABLE
#define CONF_RED_LED_STATE                  LED_CONF_ON
#define CONF_RED_LED_BLINKING_MS                1
#define CONF_AMB_LED_PS_STATE                 PS_ENABLE
#define CONF_AMB_LED_STATE                  LED_CONF_ON
#define CONF_AMB_LED_BLINKING_MS                1
#endif
```

7.5 Power Monitor Module

The User would be able to use the below functionalities of the power monitor module via the DB variables and configuration file.

7.5.1 Power Monitor module Enable/Disable

The SDK provides the user the ability to enable/disable the power monitor functionality by modifying the default configuration file. Please see section 7.5.5 for sample configuration.

If the Configurable inputs is disabled then the power monitor module will also be disabled in the configuration file.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_POWER_MONITOR	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the power monitor module in the SDK **PS_DISABLE: -** Disables the power monitor module in the SDK

7.5.2 Power Monitor Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.5.5 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_POWER_MONITOR_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.5.3 Power Monitor task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the Power Monitor module in the configuration file. Please see section 7.5.5 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_POWER_MONITOR_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.5.4 Power Monitor Functionality Support

The AI280 SDK supports the below values which are monitored by the power monitor module. They are,

1) Battery LEVEL
2) Ignition Status
3) THERMOSTAT

During runtime, the user can read the Battery level, Ignition status and Thermostat level by reading the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
BATTERY_LEVEL	DBu16	READ	2	Voltage in milli volts	This field is used to read the BATTERY_LEVEL in milli-volts.
IGNITION_STATUS	DBu8	READ	1	ON/OFF	This field is used to read the status of IGNITION_STATUS.
THERMOSTAT_LEVEL	float	READ	4	temperature in Celsius	This field is used to read the THERMOSTAT_LEVEL in Celsius.

The below code snippet shows how to read the Battery Level, Ignition Status and the Thermostat level.

```
uint8_t val;
void PWRMNTRView::trigger()
{
#if (SDK_SERVICE_POWER_MONITOR == PS_ENABLE)
    uint16_t val = 0;
    float val_thermostat = 0;
    switch(key_position)
```

```
        {
            case 1:
                /* Get ignition Status */
                Get_DL(IGNITION_STATUS, (uint8_t*)&val);
                break;
            case 2:
                /* Get Temperature level */
                Get_DL(THERMOSTAT_LEVEL, (uint8_t*)&val_thermostat);
                break;
            case 3:
                /* Get Battery level */
                Get_DL(BATTERY_LEVEL, (uint8_t*)&val);
                break;
        }
#endif
}
```

7.5.5 Power Monitor sample configuration

```
/* Power Monitor Platform Service Enable (PS_ENABLE) / Disable (PS_DISABLE)
Macros */
#define SDK_SERVICE_POWER_MONITOR                       PS_ENABLE
#if (SDK_SERVICE_POWER_MONITOR == PS_ENABLE)
#if ((SDK_SERVICE_POWER_MONITOR == PS_ENABLE) && (SDK_SERVICE_CFG_INPUT ==
PS_DISABLE))
#undef SDK_SERVICE_POWER_MONITOR
#define SDK_SERVICE_POWER_MONITOR                       PS_DISABLE
#endif
/*!
 * Power Monitor Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *         ,,                          ,,
 *         ,,                          ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
 * osPriorityReserved    = 0x7FFFFFFF
 */
#define PS_POWER_MONITOR_TASK_PRIORITY          osPriorityIdle
/*!
 * Power Monitor Task Periodicity   100ms
 */
#define PS_POWER_MONITOR_TASK_TIMEOUT           100
```

```
#endif      //SDK_SERVICE_POWER_MONITOR
```

7.6 Bluetooth Low Energy (BLE) Module

The AI280 SDK User would be able to use the below functionalities of the BLE module via the DB variables and configuration file.

7.6.1   BLE module Enable/Disable

The SDK provides the user the ability to enable/disable the BLE functionality by modifying the default configuration file. Please see section 7.6.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_BLE	PS_ENABLE/ PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the BLE module in the SDK **PS_DISABLE: -** Disables the BLE module in the SDK

7.6.2   BLE Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.6.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_BLE_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.6.3   BLE Monitor task Priority

The AI280 SDK supports the below task priorities, and the user can modify the task priority for the BLE module in the configuration file. Please see section 7.6.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_BLE_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityBelowN	osPriorityBelowNormal7 (Recommended	User can select any one of the priorities based on the application requirement

		ormal7, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	Value)	

7.6.4 BLE module device Name configuration

The AI280 SDK user can configure the device name of BLE. This default configuration can be done in the AI280_config.h

Sr. No	Variables	Options	Default State	Description
1	BLE_DEVICE_NAME	Any name as per the user requirement	"Max12345678"	This field is used to set and read the BLE device name. The maximum length is 20 characters.

The user would be able to read the BLE module name during run time via the DB variables shown below.

Field ID	Data Type	Permission	Size	Description	Comments
BLE_DEVICE_NAME	DBu8	READ/WRITE	20	"devicename"	This field is used to set and read the BLE device name. The maximum length is 20 characters.

The below code snippet shows how to read the BLE name,

```
#if (SDK_SERVICE_BLE == PS_ENABLE)
        uint8_t name;
        /* Get the BLE device name */
        Get_DL(BLE_DEVICE_NAME, &name);
```

7.6.5 BLE module RX/TX

The AI280 SDK allows the users to use the BLE channel to send or receive data. To do so please use the below variables.

To read incoming data over the BLE channel, the user will need to monitor the RX BUFFER STATUS variable and see if there is any pending data available and if yes read the data and then update the RX STATUS.

To send data over the BLE channel, the user will fill the TX buffer and then update the status as true. The platform will then send the data over BLE and then clear the status when the data is sent.

Field ID	Data Type	Permission	Size	Options	Description
BLE_TX_STATUS	DBu8	READ	1	TRUE/FALSE	This field is used read and write the BLE_TX_STATUS. One needs to write TRUE to send data. The same is cleared when data is sent
BLE_TX_BUFFER_STATUS	DBu8	READ	1	FULL/NO_FULL	This field is used to read the Status of the TX buffer.
BLE_TX_DATA	DBu8	WRITE	64	Data to be send to BLE	This field contains the BLE TX buffer data.
BLE_RX_STATUS	DBu8	READ/WRITE	1	TRUE/FALSE	This field is used read and write the BLE_RX_STATUS. One needs to read TRUE to receive data. The same is cleared when data is sent
BLE_RX_BUFFER_STATUS	DBu8	READ	1	MSG_PENDING/EMPTY	This field is used to read the Status of the RX buffer.
BLE_RX_DATA	DBu8	READ	64	DATA received	This field contains the BLE RX buffer data.
BLE_RX_DATA_SIZE	DBu8	READ	1	(Only applicable for USER_DATA_MODE)	This field is used the read the size of BLE RX data.

The sample code below suggests the process to read the RX Data received.

```
Unicode::UnicodeChar buffer1[64];
uint8_t rxbuffer1[64];
Unicode::UnicodeChar trxbuffer3[64];
      char str4[50];
void BLEView::trigger()
{
#if (SDK_SERVICE_BLE == PS_ENABLE)
      uint8_t status ;
      /* Get the RX status */
      Get_DL(BLE_RX_BUFFER_STATUS, &status);
      if(BLE_RX_MSG_PENDING == status)
      {
            /* Clear the memory */
```

```
            memset(&rxbuffer1[0], 0x00, sizeof(rxbuffer1));
            /* Read the Rx data from the DB */
            Get_DL(BLE_RX_DATA, (uint8_t *)&rxbuffer1[0]);
            memset(&trxbuffer3[0], 0x00, sizeof(trxbuffer3));
            Unicode::strncpy(&trxbuffer3[0], (const char*)&rxbuffer1[0] ,
            strlen((const char*)rxbuffer1));
            memset(RCVTEXTBBuffer, 0x00, sizeof(RCVTEXTBBuffer));
            Unicode::snprintf(RCVTEXTBBuffer, RCVTEXTB_SIZE, "%c %c %c %c
%c",       trxbuffer3[0], trxbuffer3[1] ,trxbuffer3[2],trxbuffer3[3],
      trxbuffer3[4]);
            RCVTEXTB.invalidate();
            status = TRUE;
            /* Clear the RX buffer */
            Set_DL(BLE_RX_STATUS, &status);
        }
#endif
      this->getRootContainer().invalidate();
}
```

The sample code below suggests the process to send the Data over Bluetooth,

```
      /*set the BLE tx data */
      Set_DL(BLE_TX_DATA , (uint8_t *)&buffer1[0]);
       status = TRUE;
       /* Clear the RX buffer */
       Set_DL(BLE_TX_STATUS, &status);
```

7.6.6 BLE sample configuration

```
/*!
 * BLE Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_BLE                             PS_ENABLE
#if (SDK_SERVICE_BLE == PS_ENABLE)
/*!
 * BLE Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *          ,,                          ,,
 *          ,,                          ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
 * osPriorityReserved    = 0x7FFFFFFF
 */
```

```
#define PS_BLE_TASK_PRIORITY
osPriorityBelowNormal7
/*!
 * BLE Task Periodicity 100ms
 */
#define PS_BLE_TASK_TIMEOUT                          100
/*!
 * BLE Device Name
 */
#define BLE_DEVICE_NAME                    "Max12345678"
#endif      //SDK_SERVICE_BLE
```

7.7 Timer Module

The AI280 SDK User would be able to use the below functionalities of the Timer module via the DB variables and configuration file.

7.7.1 Timer Module Enable/Disable

The SDK provides the user the ability to enable/disable the Timer functionality by modifying the default configuration file. Please see section 7.7.7 for sample code snippet.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_S WTIMER	PS_ENABLE/PS _DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the timer module in the SDK **PS_DISABLE: -** Disables the timer module in the SDK

7.7.2 Timer Module Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.7.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_SWT_TASK_T IMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.7.3 Timer Module Task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the timer module in the configuration file. Please see section 7.7.7 for sample code snippet.

Sr. No	Variables	Options	Default State	Description
1	PS_SWT_TASK_P RIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError,	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

		osPriorityReserved			

7.7.4 Timer Start or Stop

The AI280 SDK supports six software timers. The user can start or stop the timers during run time and also get the current status of the timer. To do so he can read or write the timer state using the below DB variables.

Field ID	Data Type	Permission	Size Bytes	Description	Comments
TIMER_STATUS_01	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).
TIMER_STATUS_02	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).
TIMER_STATUS_03	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).
TIMER_STATUS_04	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).
TIMER_STATUS_05	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).
TIMER_STATUS_06	DBu8	READ/WRITE	1	START/STOP	This field is used set and read the Timer state (START/STOP).

Below code snippet shows how the timer state can be set

```
uint8_t state;
Get_DLTIMER_STATUS_01, &state);
if(state == 1)
{
state = 2;
Set_DL(TIMER_STATUS_01, &state);
}
else if(state == 2)
{
```

```
state = 1;
set_DL(TIMER_STATUS_01, &state);
}
```
Once the timer expires the SDK updates the timer callback parameter in the DB with the status as `CALLBACK_OCCURED` and the user can monitor the same to know if the timer has expired. He can use the below DB variables to do the same.

Field ID	Data Type	Permission	Size Bytes	Description	Comments
TIMER_CALLBACK_01	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.
TIMER_CALLBACK_02	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.
TIMER_CALLBACK_03	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.
TIMER_CALLBACK_04	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.
TIMER_CALLBACK_05	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.
TIMER_CALLBACK_06	DBu8	READ/WRITE	1	CALLBACK_CLEAR/ CALLBACK_OCCURED	This field is used set and clear the Timer state.

The below code snippet shows you how you can read the S/W timer status,

```
#if (SDK_SERVICE_SWTIMER == PS_ENABLE)
```

```
    uint8_t timer1_state = 0;
    uint8_t rtc1_val = 0;
    uint8_t timeout_val = 0;
    Get_DL(TIMER_CALLBACK_01, &timer1_state);
    if(CALLBACK_OCCURED == timer1_state)
    {
        /* Timer expired */
    }
#endif
```

7.7.5   Timer Mode Configuration

The S/W timers can be configured as single shot and periodic. During runtime, the user can read or write timer mode variable in the DB to update/get the configuration of the S/W timers.

Field ID	Data Type	Permission	Size Bytes	Description	Comments
TIMER_MODE_01	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)
TIMER_MODE_02	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)
TIMER_MODE_03	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)
TIMER_MODE_04	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)
TIMER_MODE_05	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)
TIMER_MODE_06	DBu8	READ/WRITE	1	ONESHOT/ PERIODIC	This field is used to set and read Timer Mode. (ONESHOT/PERIODIC)

Below code snippet shows how the timer state can be set and read,

```
uint8_t shot;
GET_DB(TIMER_MODE_01, (uint8_t *)&shot);
if(shot == 0)
{
```

```
        shot = 1;
}
else if(shot == 1)
{
        shot = 0;
}
/* Set the Timer Mode_1 */
SET_DB(TIMER_MODE_01, (uint8_t *)&shot);
```

7.7.6   Timer Timeout Configuration

During runtime, the user can set or get the timeout period for the S/W timers using the below DB variables.

Please note that the timer timeout can be increased in steps of 50ms. And the max timeout value should be lesser than 65535.

Field ID	Data Type	Permission	Size Bytes	Description	Comments
TIMER_TIMEOUT _01	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in milli second
TIMER_TIMEOUT _02	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in milli second
TIMER_TIMEOUT _03	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in milli second
TIMER_TIMEOUT _04	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in milli second
TIMER_TIMEOUT _05	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in milli second
TIMER_TIMEOUT _06	DBu16	READ/WRITE	2	time in milli seconds	This field is used to get/set the timeout in millisecond

Below code snippet shows how the timer timeout can be set,

```
uint32_t sw_timeout;
uint32_t timeout;
if ((timeout > 0) && (timeout <= 1300))
{
timeout --;
sw_timeout = (timeout * 50);
}
else
```

```
{
}
/* Set the Timer Timeout_1 */
SET_DB(TIMER_TIMEOUT_01, (uint8_t *)&sw_timeout);
```

Please note that the timer timeout can be increased in steps of 50ms. And the max timeout value should be lesser than 65535 hence the max counter in the above loop is restricted to 1300.

7.7.7 Timer sample configuration

```
/*!
 * SWTIMER Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_SWTIMER                          PS_ENABLE
#if (SDK_SERVICE_SWTIMER == PS_ENABLE)
/*!
 * SWTIMER Task Priority
 * osPriorityNone          =  0,
 * osPriorityIdle          =  1,
 * osPriorityLow           =  8,
 * osPriorityLow1          =  8+1,
 *          ,,                              ,,
 *          ,,                              ,,
 * osPriorityISR           = 56,
 * osPriorityError         = -1,
 * osPriorityReserved      = 0x7FFFFFFF
 */
#define PS_SWT_TASK_PRIORITY                         osPriorityIdle
/*!
 * SWTimer Task Periodicity   100ms
 */
#define PS_SWT_TASK_TIMEOUT                          100

#endif        //SDK_SERVICE_SWTIMER
```

7.8 RTC Module

The AI280 SDK User would be able to use the below functionalities of the RTC module via the DB variables and configuration file.

7.8.1 RTC Module Enable/Disable

The SDK provides the ability to the user to enable/disable the RTC functionality by modifying the default file. Please see section 7.8.8 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_RTC	PS_ENABLE/PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the RTC module in the SDK **PS_DISABLE: -** Disables the RTC module in the SDK

7.8.2 RTC Timeout Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.8.8 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_RTC_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.8.3 RTC Task Priority

The AI280 SDK supports the below task priority and the user can modify the task priority for the RTC module in the configuration file. Please see section 7.8.8 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_RTC_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR,	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

		osPriorityError, osPriorityReserved		

The user would be able to use the below functionalities of the RTC module via the DB variables and configuration file.

7.8.4 RTC Date and Time Configuration

The user can get or set the real time clock using the following DB variables.

To set the RTC, the user has to set individually each of the RTC parameters and then call the SET_RTC DB variable to set the RTC TIME.

Field ID	Data Type	Permission	Size	Description	Comments
RTC_SECOND	DBu8	READ/WRITE	1	0-59	Valid values to set the real time second are from 0 to 59
RTC_MINUTE	DBu8	READ/WRITE	1	0-59	Valid values to set the real time minute are from 0 to 59
RTC_HOUR	DBu8	READ/WRITE	1	0-24	Valid values to set the real time hour are from 0 to 24
RTC_DATE	DBu8	READ/WRITE	1	1-31	Valid values to set the real time day are from 1 to 31
RTC_WEEK_DAY	DBu8	READ/WRITE	1	1-7	Valid values to set the real time week day are from 1 = Monday to 7= Sunday
RTC_MONTH	DBu8	READ/WRITE	1	1-12	Valid values to set the real time month are from 1 to 12
RTC_YEAR	DBu8	READ/WRITE	1	00-99	Valid values to set the real time year are from 0 to 99
SET_RTC	DBu8	READ/WRITE	1	SET_RTC	(User need to set the above RTC parameters and then enable the SET RTC to set the time)

The below snapshot is a sample for updating the RTC Time

```
uint8_t hours;
uint8_t Minutes;
uint8_t Seconds;
uint8_t WeekDay;
```

```
      uint8_t date;
      uint8_t Month;
      uint8_t Year;
      uint8_t res;
Set_DL(GET_RTC_SECOND, &Seconds);
Set_DL(GET_RTC_MINUTE, &Minutes);
Set_DL(GET_RTC_HOUR, &Hours);
Set_DL(GET_RTC_DATE, &Date);
Set_DL(GET_RTC_WEEK_DAY, &WeekDay);
Set_DL(GET_RTC_MONTH, &Month);
Set_DL(GET_RTC_YEAR, &Year);
res = 1;
      Set_DL(SET_RTC, &res);
```

The sample code below is an example of reading the RTC values.

```
      void RTCNXTView::trigger()
      {
      #if (SDK_SERVICE_RTC == PS_ENABLE)
            uint8_t Seconds;
            uint8_t Minutes;
            uint8_t Hours;
        tickCounter++;
        if( 10 <= tickCounter)
        {
              tickCounter = 0;
                /* Get the RTC DB */
                Get_DL(GET_RTC_SECOND, &Seconds);
                Get_DL(GET_RTC_MINUTE, &Minutes);
                Get_DL(GET_RTC_HOUR, &Hours);
              //screenViewBase::setupScreen();
              digitalHours = Hours;
              digitalMinutes = Minutes;
              digitalSeconds = Seconds;
            digitalClock1.setTime24Hour(digitalHours, digitalMinutes,
    digitalSeconds);
              digitalClock1.invalidate();
        }
      #endif
      }
```

7.8.5 RTC Time Format

The SDK supports the 12- and 24-hour time format. The user can read/update the RTC Time format during run time using the below DB variables. Below are their definitions,

```
#define FORMAT_12_HOUR              1
```

```
#define FORMAT_24_HOUR          0
```

Field ID	Data Type	Permission	Size	Description	Comments
RTC_TIME_FORMAT	DBu8	READ/WRITE	1	AM/PM	RTC Time Format (AM/PM)

```
uint8_t format;
/* Get the RTC time format */
  Get_DL(RTC_TIME_FORMAT, &format);
                  format  = FORMAT_24_HOUR;
 /* Set the RTC  time format */
        Set_DL(RTC_TIME_FORMAT, &format);
```

7.8.6 RTC Alarm Date and Time

The SDK platform supports 2 alarms and they can be configured by the user during run time. To set an alarm the user will need to configure the below parameters of the alarm and then enable the SET_ALARM.

Field ID	Data Type	Permission	Size	Description	Comments
RTC_ALARM_A_SECOND	DBu8	READ/WRITE	1	0-59	Valid values to set the alarm are from 0 to 59. From 60 to 255 the values are don't care to set the alarm.
RTC_ALARM_A_MINUTE	DBu8	READ/WRITE	1	0-59	Valid values to set the alarm are from 0 to 59. From 60 to 255 the values are don't care to set the alarm.
RTC_ALARM_A_HOUR	DBu8	READ/WRITE	1	0-24	Valid values to set the alarm are from 0 to 24. From 25 to 255 the values are don't care to set the alarm.
RTC_ALARM_A_DAY	DBu8	READ/WRITE	1	1-31	Valid values to set the alarm are from 1 to 31. From 32 to 255 the values are don't care to set the alarm.
RTC_ALARM_A_WEEK_DAY	DBu8	READ/WRITE	1	1-7	Valid values to set the alarm are from 1 to 7. From 8 to 255 the values are don't care to set the alarm.
RTC_ALARM_A_MONTH	DBu8	READ/WRITE	1	1-12	Valid values to set the alarm are from 1 to 12. From 13 to 255 the values are don't care to set the alarm.

RTC_ALARM_A_YEAR	DBu8	READ/WRITE	1	00-99	Valid values to set the alarm are from 0 to 99. From 100 to 255 the values are don't care to set the alarm.
SET_ALARM_A	DBu8	READ/WRITE	1	(ON/OFF)	(User need to set the above ALARM parameters and then enable the SET_ALARM1 to set the alarm time)

Field ID	Data Type	Permission	Size	Options	Description
RTC_ALARM_B_SECOND	DBu8	READ/WRITE	1	0-59	Valid values to set the alarm are from 0 to 59. From 60 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_MINUTE	DBu8	READ/WRITE	1	0-59	Valid values to set the alarm are from 0 to 59. From 60 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_HOUR	DBu8	READ/WRITE	1	0-24	Valid values to set the alarm are from 0 to 24. From 25 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_DAY	DBu8	READ/WRITE	1	1-31	Valid values to set the alarm are from 1 to 31. From 32 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_WEEK_DAY	DBu8	READ/WRITE	1	1-7	Valid values to set the alarm are from 1 to 7. From 8 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_MONTH	DBu8	READ/WRITE	1	1-12	Valid values to set the alarm are from 1 to 12. From 13 to 255 the values are don't care to set the alarm.
RTC_ALARM_B_YEAR	DBu8	READ/WRITE	1	00-99	Valid values to set the alarm are from 0 to 99. From 100 to 255 the values are don't care to set the alarm.
SET_ALARM_B	DBu8	READ/WRITE	1	(ON/OFF)	(User need to set the above ALARM parameters and then enable the SET_ALARM2 variable)

The below code snippet show how we can set the alarm,

```
uint8_t ahours;

uint8_t aminutes;

uint8_t aseconds;

uint16_t awkdays;

uint8_t ares;

Set_DL(RTC_ALARM_A_HOUR, &ahours);

Set_DL(RTC_ALARM_A_MINUTE, &aminutes);

Set_DL(RTC_ALARM_A_SECOND, &aseconds);

Set_DL(RTC_ALARM_A_WEEK_DAY, (uint8_t *)&awkdays);

Set_DL(SET_ALARM_A, &ares);
```

Once the alarm is set the user can read the ALARM_STATUS DB variable to know the status of the alarm as seen in the below table. Once the alarm occurs the status variable will be updated to OCCURRED. After the user reads the status, he will need to reset the same in the DB.

Field ID	Data Type	Permission	Size	Description	Comments
ALARM_A_ST ATUS	DBu8	READ/WRITE	1	(1: OCCURRED/ 0: NOTOCCUR RED)	Alarm1status (OCCURRED/ NOTOCCURRED)
ALARM_B_ST ATUS	DBu8	READ/WRITE	1	(1: OCCURRED/ 0: NOTOCCUR RED)	Alarm2status (OCCURRED/ NOTOCCURRED)

The below code snippet shows the alarm status,

```
uint8_t alarm_status;
/* Read the Alarm A Status from the DB */
   res = Get_DL(ALARM_A_STATUS, &alarm_status);
   if (ALARM_OCCURED == alarm_status)
     {
         alarm_status = 0;
         /* Set the ALARM A status*/
         res = Set_DL(ALARM_A_STATUS, &alarm_status);
     }
```

7.8.7   RTC Alarm Time Format

The SDK supports the 12- and 24-hour time format. The user can read/update the RTC Alarm format during run time using the below DB variables.

Below are their definitions,

```
#define FORMAT_12_HOUR               1
#define FORMAT_24_HOUR               0
```

Field ID	Data Type	Permission	Size	Description	Comments
RTC_ALARM_A_TIME_FORMAT	DBu8	READ/WRITE	1	AM/PM	RTC ALARMA time format (AM/PM)

Field ID	Data Type	Permission	Size	Description	Comments
RTC_ALARM_B_TIME_FORMAT	DBu8	READ/WRITE	1	AM/PM	RTC ALARMB time format (AM/PM)

7.8.8   RTC Sample Configuration

```
/*RTC Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros */
#define SDK_SERVICE_RTC                          PS_ENABLE
#if (SDK_SERVICE_RTC == PS_ENABLE)
/*!
 * RTC Task Periodicity 100ms
 */
#define PS_RTC_TASK_TIMEOUT                          100
/*!
 * RTC Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *          ,,                        ,,
 *          ,,                        ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
 * osPriorityReserved    = 0x7FFFFFFF
 */
#define PS_RTC_TASK_PRIORITY                        osPriorityIdle
#endif       //SDK_SERVICE_RTC
```

7.9 EEPROM Module

The AI280 SDK User would be able to use the below functionalities of the EEPROM module via the DB variables and configuration file.

7.9.1   EPROM Module Enable/Disable

The SDK provides the user the ability to enable/disable the EEPROM functionality by modifying the default file. Please see section 7.9.5 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_E EPROM	PS_ENABLE/PS _DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the EEPROM module in the SDK **PS_DISABLE: -** Disables the EEPROM module in the SDK

7.9.2   EEPROM Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.9.5 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_EE_TASK_TI MEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the configured inputs in the Database.

7.9.3   EEPROM Module Task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the timer module in the configuration file. Please see section 6.9.5 for sample code snippet.

Sr. No	Variables	Options	Default State	Description
1	PS_EE_TASK_PR IORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR,	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

		osPriorityError, osPriorityReserved.		

7.9.4   EEPROM Placeholder

The AI280 SDK supports the below size of the placeholder and the user can modify them in the configuration file. Please see section 7.9.5 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SIZE_OF_PLACE HOLDER	1 to 65535	100	User can select the size of the placeholder based on the application requirement

The SDK has currently defined 300 placeholders but the user can use 65535 placeholders. This can be used as a reference for all the additional elements that the user can use. The user would be able to read and write into the EEPROM places holder using the below DB variables. These place holders are defined in the EE_PH_DB.h file. The user can add additional variables here.

Field ID	Data Type	Permission	Size Bytes	Options	Description
EE_CAL01	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Place holder 1
EE_CAL02	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Place holder 2
EE_CAL03	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Place holder 3
EE_CAL04	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Place holder 4
EE_CAL05	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Place holder 5
…	…	…	…	…	
EE_CAL300	EEPROM_t	READ/ WRITE	variable	Place holder for EEPROM variable defined in EEPROM map	Considered placeholder for worst case scenario of each variable of 1 byte size

One the placeholder is defined is necessary to set the parameters of the EEPROM variable in the EE_User_define.h.

```
·*/¤¶
int8_t·EE_CAL01_default·=·10;¤¶
¤¶
EE_Element_info·····EE_user_elements[]·=¤¶
{¤¶
·····/*·ID,» Size,» CRC_enable,»»» Redundancy,»»» Default_data_enable,» Default_data·*/¤¶
·····{EE_CAL01·,·sizeof(EE_CAL01_default),·TRUE,·1,·TRUE,·&EE_CAL01_default},¤¶
»   {EE_CAL02·,·sizeof(EE_CAL01_default),·TRUE,·2,·TRUE,·&EE_CAL01_default},¤¶
»   {EE_CAL03·,·sizeof(EE_CAL01_default),·TRUE,·3,·TRUE,·&EE_CAL01_default},¤¶
»   {EE_CAL04·,·sizeof(EE_CAL01_default),·TRUE,·4,·TRUE,·&EE_CAL01_default},¤¶
¤¶
```

The parameters to be set is size, CRC enable, redundancy (multiple copies of the variable), enable default data and a pointer to the default data (if the reading of the variable fails it going to report the default data).

The functionality of the EEPROM platform service if all the parameters are enabled is the following: the data is going to be stored in the variable and the redundancy copies, if the principal variable fail to write the redundancy variable will be used until it fails and then a default value will be reported.

To make the EEPROM platform service update the values in the external EEPROM is necessary to set a break point in the core/Maximatecc/EEPROM_Paltformservice.c in the following section.
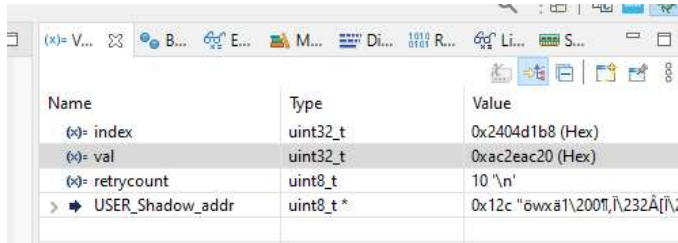
```
157   /*¶
158⊖ void·EEPROM_Shadow_Init()¤¶
159  {¤¶
160  »   uint32_t·index;¤¶
161  ····uint32_t·val;¤¶
162  ····uint8_t··retrycount·=·10;¤¶
163  ····uint8_t··*USER_Shadow_addr;¤¶
164  ¤¶
165  ····/*·Read·the·EEPROM·First·page·to·check·the·Magic·number·*/¤¶
166  ····/*·Check·Magic·number·is·present·in·the·EEPROM··········*/¤¶
167  »   /*·If·it·is·present,·the·EEPROM·is·already·initialized··*/¤¶
168  »   /*·with·the·USER·data·································*/¤¶
169  ¤¶
170  ····while(retrycount·>·0)¤¶
171  ····{¤¶
172  ·····»   TakeSPIBusLock();¤¶
173  ·····»   EEPROM_SPI_ReadBuffer((uint8_t·*)&val,·0x00,·MAGIC_NUMBER_SIZE);¤¶
174  ·····»   GiveSPIBusLock();¤¶
175  ·····»   if·(EEPROM_MAGIC_NUMBER·==·val)¤¶
176  ·····»   »   break;¤¶
177  ¤¶
178  ·····»   retrycount--;¤¶
179  ····}¤¶
180  ¤¶
181  |   if·(EEPROM_MAGIC_NUMBER·!=·val)¤¶
182  ····{¤¶
183  »   »   /*·This·is·the·first·time·writing,·So·initialize·the·shadow·memory·*/¤¶
184  »   »   Initialize_EEPlaceHolder();¤¶
185  »   »   Init_Shadow_memory();¤¶
186  »   }¤¶
187  »   else¤¶
188  »   {¤¶
```

After the breakpoint is reached in the window of local variables (upper right of the screen) the val value should be modified to make the EEPROM platform service format the external EEPROM, this is only necessary at debug stage and only when a new variable is defined.

To access the variables the user must use the START_EEPROM value + offset. For example, to access the variable to access the variable *EE_CAL01* , the user will use the OFFSET as START_EEPROM + EE_CAL01.

The sample code below gives an example to access the Placeholders for EEPROM.

```c
uint8_t value;
if(KEY4_SHORT_PRESS == val)
    {
        Get_DL((START_EEPROM+ EE_CAL01),(uint8_t *)&value);
    }
```

7.9.5   EEPROM Sample Configuration

```c
/*!
 * EEPROM Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_EEPROM                              PS_ENABLE
#if (SDK_SERVICE_EEPROM == PS_ENABLE)
/*!
 * EEPROM Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *            ,,                          ,,
 *            ,,                          ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
 * osPriorityReserved    = 0x7FFFFFFF
 */
#define PS_EE_TASK_PRIORITY                             osPriorityIdle
/*!
 * EEPROM Task Periodicity    100ms
```

```
 */
#define PS_EE_TASK_TIMEOUT                         100
/*!
 * EEPROM Place holder size
 */
#define SIZE_OF_PLACEHOLDER                        100
#endif        //SDK_SERVICE_EEPROM
```

7.10 Watch Dog Module

The User would be able to use the below functionalities of the watch dog module via the DB variables and configuration file.

7.10.1 Watch dog module Enable/Disable

The SDK provides the user the ability to enable/disable the watch dog module functionality by modifying the default configuration file. Please see section 7.10.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_WATCHDOG	PS_ENABLE/ PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the watch dog module in the SDK **PS_DISABLE: -** Disables the watch dog module in the SDK

7.10.2 Watch dog Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any else perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.10.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_WD_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the database.

7.10.3 Watch Dog Task Priority

The AI280 SDK supports the below task priorities, and the user can modify the task priority for the Watchdog module in the configuration file. Please see section 7.10.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_WD_TASK_PRIORITY	osPriorityNone, osPriorityIdle osPriorityLow, osPriorityLow1, osPriorityISR,	osPriority Idle (Recomm ended	User can select any one of the priorities based on the application requirement

		osPriorityError, osPriorityReserved	Value)	

7.10.4 Watchdog User Task Enable\Disable

The SDK provides the user the ability to enable/disable the watch dog functionality by modifying the default configuration file. Please see section 7.10.7 for sample configuration.

S.No	Variables	Options	Default State	Description
1	USER_TASK_WD0	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#0 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#0 watchdog module in the SDK
2	USER_TASK_WD1	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#1 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#1 watchdog module in the SDK
3	USER_TASK_WD2	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#2 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#2 watchdog module in the SDK
4	USER_TASK_WD3	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#3 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#3 watchdog module in the SDK

5	USER_TASK_WD4	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#4 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#4 watchdog module in the SDK
6	USER_TASK_WD5	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#5 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#5 watchdog module in the SDK
7	USER_TASK_WD6	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#6 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#6 watchdog module in the SDK
8	USER_TASK_WD7	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#7 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#7 watchdog module in the SDK
9	USER_TASK_WD8	PS_ENABLE/ PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the user task#8 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#8 watchdog module in the SDK
10	USER_TASK_WD9	PS_ENABLE/	PS_DISABLE	**PS_ENABLE: -**

		PS_DISABLE		Enables the user task#9 watchdog module in the SDK **PS_DISABLE: -** Disables the user task#9 watchdog module in the SDK

7.10.5   Watchdog Feed Timer Configuration

Watchdog is used for automatic correction of temporary hardware/software faults by resetting the MCU. The AI280 SDK allows the user to configure the watchdog timer. Once this timer expires the watchdog service would check if all the registered tasks are if any of the tasks has not ping the watchdog service then it would reset the MCU. This timer value can be configured using the below parameter. Please see section 7.10.7 for sample configuration.

The user can configure the watchdog timer with different prescaler values as supported by the platform and they correspond to equivalent time. For example, when configured as IWDG_PRESCALER_256 the watchdog module expects to be refreshed every 40-50 secs else it would reset the MCU.

Sr. No	Variables	Options	Default State	Description
1	WATCHDOG_FEED_TIME	IWDG_PRESCALER_4 IWDG_PRESCALER_8 IWDG_PRESCALER_16 IWDG_PRESCALER_32 IWDG_PRESCALER_64 IWDG_PRESCALER_128 IWDG_PRESCALER_256	IWDG_PRESCALER_256	Watchdog feed time triggers a reset sequence when it is not refreshed within the expected time window

7.10.6 Watchdog Ping Functionality

The SDK watchdog service will reset the MCU if it finds that any of the threads are not functional. Hence as a user task it would be the user's responsibility to keep pinging the watchdog service and updating the alive status. During runtime, the user can write to the below DB variable to report the alive status to the watchdog service.

Each user task has a corresponding watchdog ping variable that it needs to update. For example, user task 1 would use the WDO_PING variable as it has enabled the USER_TASK_WD0 variable in the configuration file.

Field ID	Data Type	Permission	Size	Description	Comments
WD0_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WDO_PING variable informs the platform service that task 0 is alive.
WD1_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD1_PING variable informs the platform service that task 1 is alive.
WD2_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD2_PING variable informs the platform service that task 2 is alive.
WD3_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD3_PING variable informs the platform service that task 3 is alive.
WD4_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD4_PING variable informs the platform service that task 4 is alive.
WD5_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD5_PING variable informs the platform service that task 5 is alive.
WD6_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD6_PING variable informs the platform service that task 6 is alive.
WD7_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD7_PING variable informs the platform service that task 7 is alive.
WD8_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD8_PING variable informs the platform service that task 8 is alive.
WD9_PING	DBu8	READ/WRITE	1	TASK_ID (1-10)	Setting the task ID to the WD9_PING variable informs the platform service that task 9 is alive.

The sample code gives an example to ping for user task 5,

```
uint8_t state;
if(user_task_wd5 == 1)
```

```
{
#if(USER_TASK_WD5 == PS_ENABLE)
      state = 6; // where 6 is the task ID
      Set_DL(WD5_PING , &state);
#endif
}
```

7.10.7 Watchdog Sample Configurations

```
/*!
 * Watchdog Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_WATCHDOG                             PS_DISABLE
#if (SDK_SERVICE_WATCHDOG == PS_ENABLE)
/*!
 * Watchdog Task Periodicity        100ms
 */
#define PS_WD_TASK_TIMEOUT                              100
/*!
 * Watchdog Task Priority
 * osPriorityNone           =  0,
 * osPriorityIdle           =  1,
 * osPriorityLow            =  8,
 * osPriorityLow1           =  8+1,
 *          ,,                          ,,
 *          ,,                          ,,
 * osPriorityISR          = 56,
 * osPriorityError        = -1,
 * osPriorityReserved     = 0x7FFFFFFF
 */
#define PS_WD_TASK_PRIORITY                            osPriorityIdle
/*!
 * Watchdog Reset timer
 */
#define PS_WD_RESET_TIMER                             500
/*
 * IWDG_PRESCALER_4
 * IWDG_PRESCALER_8
 * IWDG_PRESCALER_16
 * IWDG_PRESCALER_32
 * IWDG_PRESCALER_64
 * IWDG_PRESCALER_128
 * IWDG_PRESCALER_256
 */
#define WATCHDOG_FEED_TIME                            IWDG_PRESCALER_256
```

```
/*!
 * Watchdog external task ping_id
 *
 * MAX Supported USER Watchdog is 10
 */
#define USER_TASK_WD0                    PS_DISABLE
#define USER_TASK_WD1                    PS_DISABLE
#define USER_TASK_WD2                    PS_DISABLE
#define USER_TASK_WD3                    PS_DISABLE
#define USER_TASK_WD4                    PS_DISABLE
#define USER_TASK_WD5                    PS_DISABLE
#define USER_TASK_WD6                    PS_DISABLE
#define USER_TASK_WD7                    PS_DISABLE
#define USER_TASK_WD8                    PS_DISABLE
#define USER_TASK_WD9                    PS_DISABLE
#endif      //SDK_SERVICE_WATCHDOG
```

7.11 Power Mode Module

The User would be able to use the below functionalities of the Power Mode module via the DB variables and configuration file.

7.11.1 Power mode module Enable/Disable

The SDK provides the user the ability to enable/disable the power mode functionality by modifying the default configuration file. Please see section 7.11.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_PM	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the power mode module in the SDK **PS_DISABLE: -** Disables the power mode module in the SDK

7.11.2 Power Mode Time Out Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any and perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.11.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_PM_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the database.

7.11.3 Power Mode task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the Power Mode module in the configuration file. Please see section 7.11.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_PM_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1,	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

		osPriorityISR, osPriorityError, osPriorityReserved		

7.11.4 Power Mode Wake Up Source Configuration

The AI280 SDK allows the user to configure the wake-up source, so that the device can exit from the low power mode. To do so, he can configure the below parameters in the configuration file. This default configuration can be done in the AI280_config.h. Please see section 7.11.7 for sample configuration.

The platform support wakes up from the below sources and they can be configured using the below parameters.

1) Keypad
2) RTC
3) Ignition
4) CAN

Sr. No	Variables	Options	Default State	Description
1	KEYPAD01_WAKEUP_SOURCE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the keypad 1 as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the keypad 1 as a wake-up source to exit low power mode.
2	KEYPAD02_WAKEUP_SOURCE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the keypad 2 as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the keypad 2 as a wake-up source to exit low power mode.
3	KEYPAD03_WAKEUP_SOURCE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the keypad 3 as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the keypad 3 as a wake-up source to exit low power mode.
4	KEYPAD04_WAKEUP_SOURCE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the keypad 4 as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the keypad 4 as a wake-up source to exit low power mode.
5	RTC_WAKEUP_SOURCE_STATE	PS_ENABLE	PS_DISABLE	**PS_ENABLE: -** Enables the RTC as a wake-up source to exit low power mode. **PS_DISABLE: -**

			PS_DISABLE		Disables the RTC as a wake-up source to exit low power mode.
6	IGN_WAKEUP_S OURCE	PS_ENABLE PS_DISABLE	PS_ENABLE		**PS_ENABLE: -** Enables the ignition as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the ignition as a wake-up source to exit low power mode.
7	CAN_WAKEUP_S OURCE	PS_ENABLE PS_DISABLE	PS_ENABLE		**PS_ENABLE: -** Enables the CAN as a wake-up source to exit low power mode. **PS_DISABLE: -** Disables the CAN as a wake-up source to exit low power mode.

7.11.5 Power Mode RTC Timeout

The AI280 SDK allows the user to configure the RTC as a wake-up source, so that the device can exit from the low power mode. He also can set the timeout for the RTC to wake up the system. To do so, he can configure the below parameters in the configuration file. This default configuration can be done in the AI280_config.h. Please see section 7.11.7 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	RTC_WAKEUP_S OURCE_TIMEOU T	MIN VALUE: 0 MAX VALUE: 65535	10000ms (Recommende d Value)	User can set the RTC wake up timeout using the configuration.

7.11.6 Power Mode Enable

The user would be able to enter the power mode during runtime using the below the DB variable (POWER_MODE). The platform supports the below three power mode configurations.

1) STOP MODE

Field ID	Data Type	Permission	Size Bytes	Description	Comments
POWER_MODE	DBu8	READ/WRI TE	1	STOP	This field is used to set and read the Power mode configuration

The below code snippet shows how the user can enable the different power mode configuration.

```c
uint8_t pm_state;
#if (SDK_SERVICE_PM == PS_ENABLE)
                /* Stop Mode */
```

```
                    pm_state = PM_STOP_MODE;
                    Set_DL(POWER_MODE, &pm_state);
        }
#endif
```

7.11.7 Power Mode default Configurations

```
/*!
 * Power Management Platform service Enable (PS_ENABLE) / Disable
(PS_DISABLE) Macros
 */
#define SDK_SERVICE_PM                          PS_ENABLE
#if (SDK_SERVICE_PM == PS_ENABLE)
/*!
 * Power Management Task Periodicity      100ms
 */
#define PS_PM_TASK_TIMEOUT                      100
/*!
 * GPIO Wake up Source
 */
#define KEYPAD01_WAKEUP_SOURCE                  PS_ENABLE
#define KEYPAD02_WAKEUP_SOURCE                  PS_ENABLE
#define KEYPAD03_WAKEUP_SOURCE                  PS_ENABLE
#define KEYPAD04_WAKEUP_SOURCE                  PS_ENABLE
/*!
 * RTC Wake up Source
 */
#define RTC_WAKEUP_SOURCE_STATE                 PS_DISABLE
#define RTC_WAKEUP_SOURCE_TIMEOUT               10000
/*!
 * IGN Wake up Source
 */
#define IGN_WAKEUP_SOURCE                              PS_ENABLE
#endif //SDK_SERVICE_PM
```

7.12 LCD Module

The User would be able to use the below functionalities of the LCD module via the DB variables and configuration file.

7.12.1 LCD mode module Enable/Disable

The SDK provides the user the ability to enable/disable the LCD functionality by modifying the default configuration file. Please see section 7.12.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_LCD	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the lcd module in the SDK **PS_DISABLE: -** Disables the lcd module in the SDK

7.12.2 LCD Module Timeout Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any and perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.12.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_LCD_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the database.

7.12.3 LCD task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the LCD module in the configuration file. Please see section 7.12.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_LCD_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.12.4 LCD State

The AI280 SDK supports the user to configure the default state of the LCD (either OFF/ON) and this can be done by modifying the below parameter in the configuration file. Please see section 7.12.6 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	CONF_LCD_STATE	LCD_CONF_ON LCD_CONF_OFF	LCD_CONF_ON	User can configure the LCD state as ON/OFF.

The user can also get/set the default state of the LCD (either OFF/ON) during runtime using the below DB variable

Field ID	Data Type	Permission	Size	Description	Comments
LCD_STATE	DBu8	READ/WRITE	1	ON/OFF	This field sets and reads back the Turn ON or OFF the LCD display

The below code snippet shows how you can set or get the LCD state,

```
uint8_t state;
/* Get the LCD State value from the DB */
Get_DL(LCD_STATE , &state);
if (LCD_CONF_OFF == state)
    {
        state = LCD_CONF_ON;
        /* Set the LCD state to ON */
        Set_DL(LCD_STATE, &state);
    }
```

7.12.5   LCD Brightness

The AI280 SDK supports the user to configure the LCD brightness, and this can be done by modifying the below parameter in the configuration file. Please see section 7.12.6 for sample configuration.

The below configuration means the screen is at 30% brightness level. If the user needs full brightness, then it will need to set it at 100.

Sr. No	Variables	Options	Default State	Description
1	CONF_LCD_BRIGHTNESS	0-100	30%	User can configure the LCD brightness

The user would be able to read and modify the below functionalities of the LCD module via the DB variables and configuration file.

Field ID	Data Type	Permission	Size	Description	Comments
LCD_BRIGHT NESS	DBu8	READ/WRITE	1	0-100	This field sets the percentage of brightness from 0 to 100 (full brightness) for the LCD.

The sample code gives an example to set the brightness of the LCD.

```
        uint16_t brightness_value;
        uint8_t state;
void LCDView::brightnessinc()
{
        #if (SDK_SERVICE_LCD == PS_ENABLE)
        brightness_value++;
        if (!(LCD_BRT_MAX >= brightness_value))
        brightness_value = LCD_BRT_MAX;
        Set_DL(LCD_BRIGHTNESS, (uint8_t *)&brightness_value);
        #endif
          }
```

The sample code gives an example to get the brightness of the LCD.

```
LCDView::LCDView()
{
#if (SDK_SERVICE_LCD == PS_ENABLE)
        brightness_value = 0;
        /* Get the LCD Brightness value from the DB */
        Get_DL(LCD_STATE , &state);
        Get_DL(LCD_BRIGHTNESS, (uint8_t *)&brightness_value);
#endif
}
```

7.12.6 LCD Sample Configurations

```
/*!
 * LCD Platform service Enable (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_LCD                          PS_ENABLE
#if (SDK_SERVICE_LCD == PS_ENABLE)
/*!
 * LCD Task Periodicity 100ms
 */
#define PS_LCD_TASK_TIMEOUT                          100
/*!
 * LCD Task Priority
 * osPriorityNone        = 0,
 * osPriorityIdle        = 1,
```

```
 * osPriorityLow          =  8,
 * osPriorityLow1         =  8+1,
 *          ,,                        ,,
 *          ,,                        ,,
 * osPriorityISR          = 56,
 * osPriorityError        = -1,
 * osPriorityReserved     = 0x7FFFFFFF
 */
#define PS_LCD_TASK_PRIORITY                        osPriorityIdle
/*!
 * MACOR Supported
 *
 * CONF_LCD_STATE               LCD_CONF_ON/
 *                                      LCD_CONF_OFF
 *
 * CONF_LCD_BRIGHTNESS          <0 - 100>
 */
#define CONF_LCD_STATE                              LCD_CONF_ON
#define CONF_LCD_BRIGHTNESS                         30
#endif      //SDK_SERVICE_LCD
```

7.13 CAN Module

The SDK supports one can channel CAN1.

Please note that standalone CAN module will be disabled when J1939 is enabled in the configuration file. To use CAN in a standalone mode J1939 has to be disabled in the configuration file.

Below code snippet from the AI280_config.h that shows the same,

```
#if ((SDK_SERVICE_J1939 == PS_ENABLE) && (SDK_SERVICE_FDCAN == PS_ENABLE))

#undef SDK_SERVICE_FDCAN

#define SDK_SERVICE_FDCAN                        PS_DISABLE

#endif
```

7.13.1 CAN Module Configuration Support

The SDK provides the user the ability to enable/disable the CAN functionality by modifying the default configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_FDCAN	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the FDCAN module in the SDK **PS_DISABLE: -** Disables the FDCAN module in the SDK

7.13.2 CAN Enable/Disable

The SDK provides the user the ability to enable/disable the CAN functionality by modifying the default configuration file. Please see section 7.13.13   for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_ENABLE	PS_ENABLE PS_DISABLE	PS_ENABLE	**PS_ENABLE: -** Enables the FDCAN1 module in the SDK **PS_DISABLE: -** Disables the FDCAN1 module in the SDK

7.13.3 CAN Module Timeout Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_FDCAN_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the database.

7.13.4 CAN task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the CAN module in the configuration file. Please see section 6.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_FDCAN_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can select any one of the priorities based on the application requirement

7.13.5 CAN Baudrate

The AI280 SDK supports the below Baud rate and the user can modify the Baud rates for the CAN module in the configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_BAUDRATE	BAUDRATE_50K/ BAUDRATE_100/ BAUDRATE_125 /BAUDRATE_250 /BAUDRATE_500 /BAUDRATE_100 0	BAUDRATE_5 0K	User can set the Baudrate for CAN1

The AI280 SDK user can change the baud rate during runtime using the below DB variables,

Field ID	Data Type	Permission	Size	Options	Description
CAN_CH0_BAUDRATE	DBu32	READ/ WRITE	1	Default 250Kbaud Supported baud rates: AUTO, 50K, 100K, 125K, 250K, 500K, 1M	CAN Channel0 Baud-rate

The below code snippet shows how the baud rate can be changed during the runtime,

```
uint8_t can1_buf[10];
/*
 * CAN1 Supporting baud rate
 * BAUDRATE_50K
 * BAUDRATE_100K
 * BAUDRATE_125K
 * BAUDRATE_250K
 * BAUDRATE_500K
 * BAUDRATE_1000K
 */
can1_buf[0] = BAUDRATE_250K;
Set_DL(CAN_CH0_BAUDRATE, &can1_buf[0]);
break;
```

7.13.6 CAN Identifier Configurations

The AI280 SDK supports the below configuration parameters for the CAN and the user can modify the same in the configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_IDENTIFIER	User Configurable ID	0x19FEFCFE	User can configure the CAN1 Identifier
2	FDCAN1_IDTYPE	FDCAN_EXTENDED_ID/ FDCAN_STANDRD_ID	FDCAN_EXTENDED_ID	User can configure the CAN1 IDTYPE as Extended/Standard
3	FDCAN1_ID	FDCAN_STANDARD_ID/	FDCAN_STANDA	User can configure the CAN1

		FDCAN_EXTENDED_ID	RD_ID	ID

7.13.7 CAN Channel configurations

The AI280 SDK supports the below filter properties for the CAN module in the configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_RXBUF FERIND	0-65535	1	User can configure the Rx buffer
2	FDCAN1_TXFRA METYPE	FDCAN_DATA_FRAME/ FDCAN_REMOTE_FRAM E	FDCAN_DATA_ FRAME	User can configure the TX frame type as Data Frame or Remote Frame
3	FDCAN1_DATAL ENGTH	FDCAN_DLC_BYTES_0 FDCAN_DLC_BYTES_1 FDCAN_DLC_BYTES_2 FDCAN_DLC_BYTES_3 FDCAN_DLC_BYTES_4 FDCAN_DLC_BYTES_5 FDCAN_DLC_BYTES_6 FDCAN_DLC_BYTES_7 FDCAN_DLC_BYTES_8 FDCAN_DLC_BYTES_12 FDCAN_DLC_BYTES_16 FDCAN_DLC_BYTES_20 FDCAN_DLC_BYTES_24 FDCAN_DLC_BYTES_32 FDCAN_DLC_BYTES_48 FDCAN_DLC_BYTES_64	FDCAN_DLC_B YTES_8	User can configure the length of the data
4	FDCAN1_ERROR STATEIND	FDCAN_ESI_ACTIVE/ FDCAN_ESI_PASSIVE	FDCAN_ESI_AC TIVE	User can configure The errors state as Active or Passive
5	FDCAN1_BITRA TESWITCH	FDCAN_BRS_ON/FDCAN _BRS_OFF	FDCAN_BRS_O N	User can configure the Bit rate switch on/off
6	FDCAN1_FDFOR MATE	FDCAN_CLASSIC_CAN FDCAN_FD_CAN	FDCAN_FD_CA N	User can configure the FDCAN format.
7	FDCAN1_TXEVE NTFIFOCONTRO L	FDCAN_STORE_TX_EVE NTS/ FDCAN_NO_TX_EVENTS	FDCAN_STORE _TX_EVENTS	User can configure the event FIFO control
8	FDCAN1_MESSA GEMARKER	0-65535	0	User can configure message marker

9	FDCAN1_RECEIVE_TASK_DELAY	0 – 500 ms	100 ms	User can configure the delay for the receive task

7.13.8 CAN Filter Configurations

The AI280 SDK supports the CAN Filter configurations, and the user can modify the same in the configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_FILTERINDEX	0-65535	0	User can configure the filter index
2	FDCAN1_FILTERTYPE	FILTER_DUAL or FILTER_RANGE or FILTER_MASK or FILTER_RANGE_NO_EIDM	FDCAN_FILTER_DUAL	User can configure the filter Type
3	FDCAN1_FILTERCONFIG	FDCAN_FILTER_DISABLE FDCAN_FILTER_TO_RXFIFO0 FDCAN_FILTER_TO_RXFIFO1 FDCAN_FILTER_REJECT FDCAN_FILTER_HP FDCAN_FILTER_TO_RXFIFO1_HP FDCAN_FILTER_TO_RXBUFFER	FDCAN_FILTER_TO_RXBUFFER	User can configure the filter
4	FDCAN1_FILTERID1	0-65535	FDCAN_DEFAULT_FILTERID1	User can configure the default filter id 1
5	FDCAN1_FILTERID2	0-65535	FDCAN_DEFAULT_FILTERID2	User can configure the default filter id 2

The user can also read and write to the below CAN filter properties during the runtime using the CAN DB variables,

Field ID	Data Type	Permission	Size	Description	Comments
CAN_CH0_FILTER_INDEX_ENABLE	DBu8	READ/WRITE	1	Enable / Disable receive data with filter. Index range 0-32 (CAN_MODE_EXTENDED_ID | CAN_MODE_BUS_MONITORING_EXTENDED_ID) Index range 0-64 (CAN_MODE_STANDARD_ID |	CAN Channel0 filter index state

				CAN_MODE_BUS_MONITORING_ STANDARD_ID)	
CAN_CH0_FI LTER_INDEX _ID	DBu32	READ/WRITE	4	FIFO ID use to filter	CAN Channel0 filter index ID
CAN_CH0_FI LTER_INDEX _IDMASK	DBu32	READ/WRITE	4	ID MASK use to filter	CAN Channel0 index ID Mask

The below code sample show how the filter index is updated from the application,

```
uint8_t can1_buf[10];
can1_buf[0] = ENABLE;
Set_DL(CAN_CH0_FILTER_INDEX_ENABLE, &can1_buf[0]);
can1_buf[0] = 10;
Set_DL(CAN_CH0_FILTER_INDEX_ID, &can1_buf[0]);
```

7.13.9 CAN Receive Task Delay

The AI280 SDK supports the configuration of the CAN Task Delay and the user can modify the CAN1 Receive Task Delay for the CAN module in the configuration file. Please see section 7.13.13 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	FDCAN1_RECEIVE_ TASK_DELAY	0-500 MS	100ms (Recommended Value)	User can configure the delay for the receive task

7.13.10 CAN Channel Modes and States

The user would be able to query the database to get the mode and current state of the CAN channels using the below CAN module DB variables.

Field ID	Data Type	Permis sion	Size	Options	Description
CAN_CH0 _MODE	DBu8	READ/ WRITE	1	Default CAN_MODE_EXTENDED_ID Options available: CAN_MODE_STANDARD_ID, CAN_MODE_EXTENDED_ID, CAN_MODE_BUS_MONITORING_STAND ARD_ID,	This field sets and reads the CAN Channel0 Modes

				CAN_MODE_BUS_MONITORING_EXTEN DED_ID,	
CAN_CH0_STATES	DBu8	READ	1	CAN States available: CAN_BUS_OFF, CAN_BUS_ON, CAN_STATE_PASSIVE, CAN_STATE_UNCHANGED,	This field is used to read the CAN Channel 0 state
CAN_CH0_COMM_STATE_EVENTS	DBu8	READ	1	Communication state events available: STATE_EVENT_NONE, STATE_EVENT_BUS_OFF, STATE_EVENT_BUS_OFF_RECOVERY, STATE_EVENT_BUS_ON, STATE_EVENT_PASSIVE, STATE_EVENT_ACTIVE, STATE_EVENT_OVERRUN, STATE_EVENT_QUEUE_FULL, STATE_EVENT_QUEUE_OVERFLOW, STATE_EVENT_QUEUE_EMPTY, STATE_EVENT_DRIVER_ERROR	This field is used to read the CAN Channel0 communication state event

The below code snippet shows how we can access the CAN mode and states,

```c
uint8_t can1_buf[10];
/*
 * Get the CAN1 state and update the mode
 */
Get_DL(CAN_CH0_STATES, &can1_buf[0]);
can1_buf[0] = CAN_MODE_STANDARD_ID;
Set_DL(CAN_CH0_MODE, &can1_buf[0]);
/*
 * Get the CAN1 communication state event
 */
Get_DL(CAN_CH0_COMM_STATE_EVENTS, &can1_buf[0]);
```

7.13.11　CAN Channel Reset

The user would be able to reset the CAN channel and CAN driver using the below CAN module DB variables.

Field ID	Data Type	Permission	Size	Options	Description
CAN_CH0_RESET	DBu8	READ/WRITE	1	TRUE = Resets the CAN controller and Bus Off mode.	This variable supports the reset of the CAN Channel 0

| CAN_CH0_DRIVER_RESET | DBu8 | READ/WRITE | 1 | TRUE = Reinitialize the CAN driver if Driver Error is Set. | This variable supports the reset of the CAN Channel 0 driver |

The below code snippets show how the user can reset the CAN channel and driver,

```c
uint8_t can1_buf[10];
/*
 * Reset the CAN CHANNEL 0
 */
case 4:
    Can1_buf[0] = TRUE;
    Set_DL(CAN_CH0_RESET, &can1_buf[0]);
    break;
/*
 * Reset the CAN Channel Driver 0
 */
case 5:
    Can1_buf[0] = TRUE;
    Set_DL(CAN_CH0_DRIVER_RESET, &can1_buf[0]);
    break;
```

7.13.12 CAN module RX/TX

The AI280 SDK allows the users to use the CAN channel to send or receive data. To do so please use the below variables.

To read incoming data over the CAN channel, the user will need to monitor the RX DATA SIZE variable and see if there is any pending data available and if yes read the data.

To send data over the CAN channel, the user will fill the TX buffer and send the data over CAN. The SDK will take care of handling the pending data.

The CAN default data packet size is defined as 64 bytes hence the user is expected to create a buffer of this size while reading the data.

Field ID	Data Type	Permission	Size	Description	Comments
CAN1_RX_BYTE_COUNT	DBu8	READ	1	CAN1 RX Byte count value	CAN Channel0 RX Byte count
CAN1_TX_BYTE_COUNT	DBu8	READ	1	CAN1 TX Byte count value	CAN Channel0 TX Byte count
CAN1_RX_DATA_IS_AVAILABLE	DBu8	READ	1	CAN1 data available flag	CAN channel0 RX data available flag
CAN1_RX_DATA_SIZE	DBu8	READ	1	CAN1 RX data size	CAN Channel0 RX data size

The sample code for sending and receiving the data from CAN is shown below,

```
      uint8_t can_rxbuffer1[64];
      uint8_t can1_buf[10];
if(CAN_BUS_OFF != can1_buf[0])
{
      Get_DL(CAN1_RX_DATA_IS_AVAILABLE, &can1_buf[0]);
}
#if (SDK_SERVICE_FDCAN == PS_ENABLE)
uint8_t status;
/* Get the CAN RX status */
Get_DL(CAN1_RX_DATA_SIZE, &status);
if(0 != status)
{
      memset(&can_rxbuffer1[0], 0x00, sizeof(can_rxbuffer1));
      /* Read the Rx data from the DB */
      Get_DL(CAN1_RX_DATA, (uint8_t *)&can_rxbuffer1[0]);
}
/*To send the data over the CAN bus, set the CAN tx data */
Set_DL(CAN1_TX_DATA , (uint8_t *)&can1buf[0]);
```

7.13.13   CAN Sample Configuration.

```
/***************************************************************************
*
*
*                    FDCAN Module
*
***************************************************************************
/*!
 * FDCAN Platform service (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_FDCAN                               PS_ENABLE
#if (SDK_SERVICE_FDCAN == PS_ENABLE)
#define FDCAN1_ENABLE                            PS_ENABLE
/*!
 * FDCAN Task Periodicity          100ms
 */
#define PS_FDCAN_TASK_TIMEOUT                     100
/*!
 * FDCAN Task Priority
 * osPriorityNone          =  0,
 * osPriorityIdle          =  1,
```

```
 * osPriorityLow          =  8,
 * osPriorityLow1         =  8+1,
 *          ,,                              ,,
 *          ,,                              ,,
 * osPriorityISR          = 56,
 * osPriorityError        = -1,
 * osPriorityReserved     = 0x7FFFFFFF
 */
#define PS_FDCAN_TASK_PRIORITY                    osPriorityIdle
#if (FDCAN1_ENABLE == PS_ENABLE)
/*
 * BAUDRATE_50K
 * BAUDRATE_100K
 * BAUDRATE_125K
 * BAUDRATE_250K
 * BAUDRATE_500K
 * BAUDRATE_1000K
 */
#define FDCAN1_BAUDRATE                            BAUDRATE_50K
#define FDCAN1_IDENTIFIER                          0x19FEFCFE
//0x111
#define FDCAN1_IDTYPE                              FDCAN_EXTENDED_ID
#define FDCAN1_TXFRAMETYPE                         FDCAN_DATA_FRAME
#define FDCAN1_DATALENGTH                          FDCAN_DLC_BYTES_8
#define FDCAN1_ERRORSTATEIND            FDCAN_ESI_ACTIVE
#define FDCAN1_BITRATESWITCH            FDCAN_BRS_ON
#define FDCAN1_FDFORMATE                           FDCAN_FD_CAN
#define FDCAN1_TXEVENTFIFOCONTROL       FDCAN_STORE_TX_EVENTS
#define FDCAN1_MESSAGEMARKER            0
#define FDCAN1_ID                       FDCAN_STANDARD_ID
#define FDCAN1_FILTERINDEX              0
#define FDCAN1_FILTERTYPE               FDCAN_FILTER_DUAL
#define FDCAN1_FILTERCONFIG
     FDCAN_FILTER_TO_RXBUFFER
#define FDCAN1_FILTERID1
     FDCAN_DEAFULT_FILTERID1
#define FDCAN1_FILTERID2
     FDCAN_DEAFULT_FILTERID2
#define FDCAN1_RXBUFFERIND                         1
/* CAN1 Receive Task Delay */
#define FDCAN1_RECEIVE_TASK_DELAY                  100
#endif      /*    FDCAN1_ENABLE    */
#endif      //SDK_SERVICE_FDCAN
```

7.14 J1939

7.14.1 J1939 Module Configuration Support

The SDK provides the user the ability to enable/disable the J1939 functionality by modifying the default configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	SDK_SERVICE_J1939	PS_ENABLE PS_DISABLE	PS_DISABLE	**PS_ENABLE: -** Enables the J1939 module in the SDK **PS_DISABLE: -** Disables the J1939 module in the SDK

7.14.2 J1939 Module Timeout Configuration

The AI280 SDK user can configure the timeout value of the task such that, every time the timeout occurs the task would go and read the hardware and update it in the DB so that when the user reads the DB, he will receive the latest updated data if any or perform any other routine tasks. This default configuration can be done in the AI280_config.h. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_J1939_TASK_TIMEOUT	MIN VALUE: 50 MAX VALUE: 500	100ms (Recommended Value)	The user can configure the timeout value of task so that the platform service would go and read the hardware and update the database.

7.14.3 J1939 task Priority

The AI280 SDK supports the below task priorities and the user can modify the task priority for the J1939 module in the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	PS_J1939_TASK_PRIORITY	osPriorityNone, osPriorityIdle, osPriorityLow, osPriorityLow1, osPriorityISR, osPriorityError, osPriorityReserved	osPriorityIdle (Recommended Value)	User can configure this macro to default priority

7.14.4 J1939 Claim Address Enable/Disable

The AI280 SDK supports the address claim functionality can be enabled or disable from the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	J1939_CLAIM_ADDRESS	PS_ENABLE/ PS_DISABLE	PS_ENABLE	User can enable/disable this macro to J1939 claim address

7.14.5 J1939 CAN Enable/Disable

The AI280 SDK supports enabling CAN1 Channel for J1939 and it can be enabled or disabled from the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	J1939_CAN1_ENABLE	PS_ENABLE/ PS_DISABLE	PS_ENABLE	User can enable/disable J1939 CAN1

7.14.6 J1939 Claim Address

The AI280 SDK supports the address claim for channel CAN1 for J1939 and it can be configured as any value between 0-255 based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	J1939_CAN1_CLAIM_ADDRESS	0-255	23	User can configure this macro to the required claim address

7.14.7 J1939 CAN Bit Rate

The AI280 SDK supports the bit rate configuration for the CAN1 for J1939 and it can be configured based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	J1939_CAN1_BITRATE	0/250/500/100	250	User can configure this macro to default J1939 CAN 1 BITRATE

7.14.8 J1939 Diagnostics Support

The AI280 SDK supports the enabling and disabling the diagnostics support for J1939 and they can be configured based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration. The SDK Currently supports the below messages,

1) DM1
2) DM2

Sr. No	Variables	Options	Default State	Description
1	EMTOS_J1939_DM1	PS_ENABLE PS_DISABLE	PS_ENABLE	User can enable/disable the Emotas J1939 DM1
2	EMTOS_J1939_DM2	PS_ENABLE PS_DISABLE	PS_ENABLE	User can enable/disable the Emotas J1939 DM2

7.14.9 J1939 Dynamic Address Claim

The AI280 SDK supports the dynamic address claim CAN1 for J1939 and it can be configured based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	J1939_CAN1_ADDRESS_CLAIM_DYNAMIC	PS_ENABLE PS_DISABLE	PS_ENABLE	User can enable/disable CAN1 dynamic address claim

7.14.10   J1939 Dynamic Address Claim Next Address Configuration

The AI280 SDK supports the dynamic address claim next address for CAN1 for J1939 and they can be configured based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration.

When dynamic address claim is set to 1, search will start from this value and up to claim the address.

Sr. No	Variables	Options	Default State	Description
1	J1939_CAN1_ADDRESS_CLAIM_NEXT_ADDRESS	0-255	80	User can configure the start address for address claim in dynamic mode for CAN1

7.14.11   J1939 Configure Number of PGN's supported

The AI280 SDK supports the configuration of the number of RX and TX PGNS on CAN1 for J1939 and they can be configured based on your requirement from the configuration file. Please see section 7.14.14 for sample configuration.

Please note that the SDK can support a maximum of 300 PGNs including RX and TX over CAN1 put together.

Sr. No	Variables	Options	Default Value	Description
1	CAN1_NUMBER_PGNS_RX	MIN VALUE = 0 MAX VALUE = 300	18	User can configure the number of RX PGNS supported on CAN1

| 3 | CAN1_NUMBER_PGNS_TX | MIN VALUE = 0 MAX VALUE = 300 | 1 | User can configure the number of TX PGNS supported on CAN1 |

### 7.14.12 J1939 PGN and SPN Configuration

The AI280 SDK supports up to 300 PGN values and the same can be configured in the J1939 stack and their values will be read and updated in the Data layer for the application's access.

The user can replace existing PGNs' or configure new PGN and SPN values for CAN1 for J1939 during compile time and same can be accessed from the DB during runtime.

7.14.12.1 J1939 Source Code

The J1939 source and its configuration files are available in the below path as highlighted in the following diagram.

Middlewares/Third_Party/J1939/*
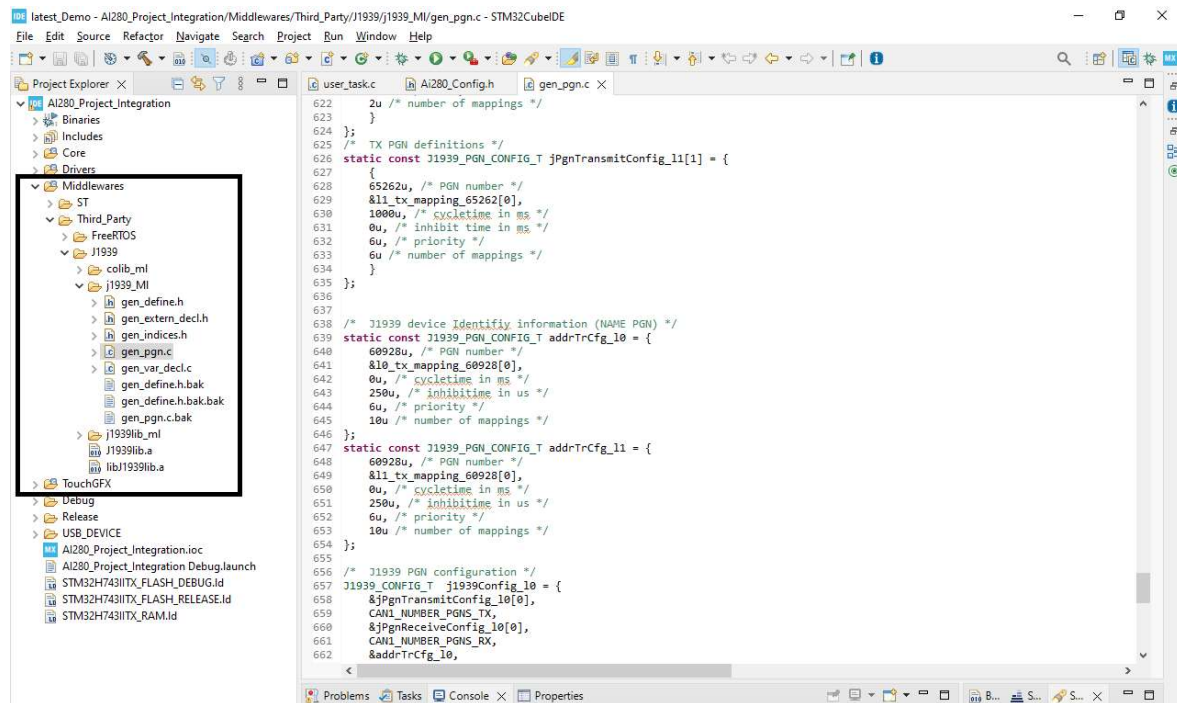
7.14.12.2 J1939 Supported PGN List

The AI280 SDK supports the below PGN values currently.

Sr. No	PGN	SIGNAL NAME
1	61444	Electronics Engine Controller 1
2	65110	Diesel Exhaust Fluid Tank 1 Information
3	65276	Dash Display
4	65272	Transmission Fluids 1
5	64774	Direct Lamp Control Command 2
7	65213	Fan Drive
8	65237	Alternator Information
9	65252	Shutdown
10	64892	Diesel Particulate filter control 1
11	65128	Vehicle Fluids VF
12	65237	Alternator information
13	65252	Shutdown
14	64998	Hydraulic Braking system
15	65089	Lighting command
16	65274	Brakes 1
17	64586	SCR System Cleaning
18	64523	Electronics Engine Controller 20
19	64525	Fire Pump Statistics 1
20	64529	Total Gaseous fuel information

7.14.12.3   J1939 Add PGN Configuration

The first step would be for the user to add the new PGN's in the configuration file.
The PGN Configuration is available in the below structure which can be found in the file Middlewares/Third_Party/J1939/j1939_ML/gen_pgn.c. This defines generic PGN structure for CAN1.

```c
/* J1939 PGN configuration */
J1939_CONFIG_T  j1939Config_l0 = {
      &jPgnTransmitConfig_l0[0],
      CAN1_NUMBER_PGNS_TX,
      &jPgnReceiveConfig_l0[0],
      CAN1_NUMBER_PGNS_RX,
      &addrTrCfg_l0,
      J1939_CAN1_CLAIM_ADDRESS
};
```

The same file has the TX and RX structures where the user can include their own PGN's into the specific channel structure based on their requirement. Please note that for each PGN added, the minimum configuration that needs to be provided is,

1) PGN Number
2) SPN Mapping structure
3) Cycle time in milli seconds
4) Inhibit time in milli seconds
5) Priority
6) Number of mappings

The below code snippet shows sample PGN definitions.

```c
/* TX PGN definitions */
static const J1939_PGN_CONFIG_T
      jPgnTransmitConfig_l0[CAN1_NUMBER_PGNS_TX] =
 {
    {
      65262u, /* PGN number */
      &l0_tx_mapping_65262[0],
      10000u, /* cycletime in ms */
      0u, /* inhibit time in ms */
      6u, /* priority */
      6u /* number of mappings */
    }
};
/* RX PGN definitions */
static const J1939_PGN_CONFIG_T
      jPgnReceiveConfig_l0[CAN1_NUMBER_PGNS_RX]=
{
    {
      61444u, /* PGN number */
      &rx_mapping_61444[0],
      10u, /* cycletime in ms */
      0u, /* inhibit time in ms */
      3u, /* priority */
```

```
      9u /* number of mappings */
       }
};
```

7.14.12.4   J1939 Add SPN Configuration

For each PGN included by the user, the supported SPN structure will need to be included in the same file. The SPN definition structure will include the below information,

1) DB Buffer with index location
2) SPN Data Size
3) SPN Number
4) Data Type
5) If it's a dynamic PGN include the dynamic variable.

For example, we have the PGN 61444 included in the RX structure and hence the PGN's supported by 61444 are defined in the below code snippet.

```
/* +++PGN 61444/0xf004 EEC1 Electronic Engine Controller 1 */
static  const J1939_MAPPING_T rx_mapping_61444[] = {
      { &dl_ju8[ENGINE_TORQUE_MODE - (START_J1939U8BIT + 1) ], 4u, 899u,
J1939_DTYPE_U4 }  /* Engine Torque Mode */,
      { &dl_ju8[ACTUAL_ENGINE_PERCENT_TORQUE_FRACTIONAL - (START_J1939U8BIT +
1)], 4u, 4154u, J1939_DTYPE_U4 }   /* Actual Engine - Percent Torque
(Fractional) */,
      { &dl_ju8[DRIVERS_DEMAND_ENGINE_PERCENT_TORQUE - (START_J1939U8BIT +
1)], 8u, 512u, J1939_DTYPE_U8 }    /* Drivers Demand Engine - Percent Torque
*/,
      { &dl_ju8[ACTUAL_ENGINE_PERCENT_TORQUE - (START_J1939U8BIT + 1)], 8u,
513u, J1939_DTYPE_U8 }  /* Actual Engine - Percent Torque */,
      { &temp_speed, 16u, 190u, J1939_DTYPE_U16 }     /* Engine Speed */,
      { &dl_ju8[SOURCE_ADDRESS_OF_DEVICE_ENGINE_CONTROL - (START_J1939U8BIT +
1)], 8u, 1483u, J1939_DTYPE_U8 }    /* Source Address of Controlling Device
for Engine Control */,
      { &dl_ju8[ENGINE_STARTER_MODE - (START_J1939U8BIT + 1)], 4u, 1675u,
J1939_DTYPE_U4 }  /* Engine Starter Mode */,
      { &mv_u8[0], 4u, 10001u, J1939_DTYPE_U4 },
      { &dl_ju8[ENGINE_DEMAND_PERCENT_TORQUE - (START_J1939U8BIT + 1)], 8u,
2432u, J1939_DTYPE_U8 } /* Engine Demand Percent Torque */
};
```

The #define for the SPN values can be added in the J1939Data_Layer.h. The path for this file and the defines are highlighted in the below image,
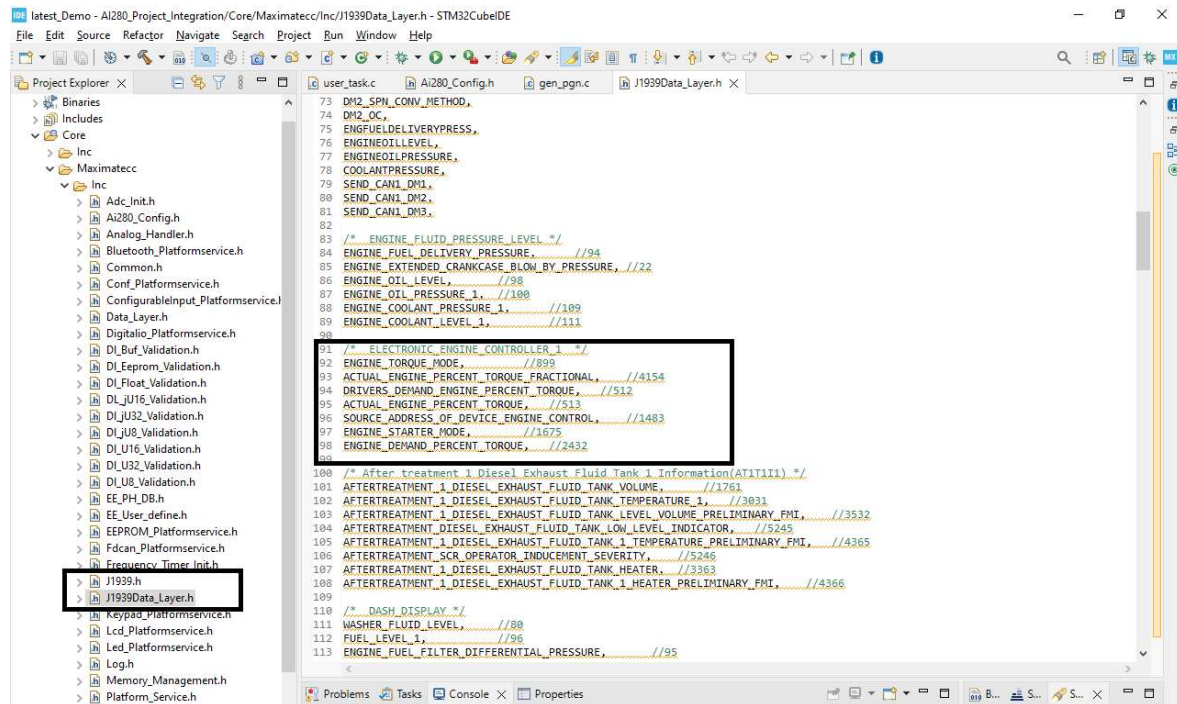
7.14.12.5   Translate the SPN's before storing in DB

Once the user has successfully defined the PGN's and added them in the required structure the SDK will take care of capturing them. But before the SPN's are updated in the DB you will need to add the code specific to the PGN which translates them to a value that can directly be used by the application.

The below code snippet shows a sample translation done in the SDK to support the SPN (Engine speed which is part of the PGN 61444,

```c
static void receivePgn(
            CO_LINE_TYPE canLine,
            UNSIGNED32  pgn,
            UNSIGNED8   node,
            J1939_RECEIVE_STATE_T state
             )
{
    float tval;
    if(pgn == 61444)
    {
/* Translate the raw data into a value that is understood by the
application. */
        tval = temp_speed * 0.125;
        dl_ju16[ENGINE_SPEED - (START_J1939U16BIT + 1)] = (uint16_t)tval;
```

```
        }
}
```
The above ENGINE SPEED PGN is the actual engine speed which is calculated over a minimum crankshaft and is at a resolution of 0.125 rpm per bit. It occupies 2 bytes of data and hence is stored in the dl_ju16 array.

For any new PGN added by the user, a similar translation may be required based on the J1939 stack specification and PGN definition support.

7.14.12.6   Access the new SPN's from DB

The application user can directly access the DB variables of each SPN value that he has configured in the stack.

```
uint8_t val;
```
The below code snippet shows the example for the same.
```
/* Get ENGINE_SPEED */
Get_DL(ENGINE_SPEED , (uint16_t*)&val);
/* Get ENGINEOILLEVEL */
Get_DL(ENGINEOILLEVEL, (uint8_t*)&val);
/* Get ENGINEOILPRESSURE */
Get_DL(ENGINEOILPRESSURE, (uint8_t*)&val);
/* Get COOLANTPRESSURE */
Get_DL(COOLANTPRESSURE, (uint8_t*)&val);
```

7.14.13   J1939 Diagnostic Message Configuration

The AI280 SDK supports the J1939 diagnostic messages which help the user understand the current state of the device.

7.14.13.1   J1939 DM1 and DM2 Support in SDK

The AI280 SDK supports the configuration of the DM1, DM2 messages. These messages are already configured in the stack and the SDK DB has entries for each of these variables. Hence the user can directly access these SPN's from the application by accessing the below DB variables.

Field ID	Data Type	Permission	Size	Description	Comments
DM1_PROTECT_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_AMBER_WARN_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_RED_STOP_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_MAL_FUNC_IND_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable

DM1_FLASH_PROT_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1__FLASH_AMBER_WARN_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_FLASH_RED_STOP_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_FLASH_MAL_FUNC_IND_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_FMI	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_SPN_CONV_METHOD	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM1_OC	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable

Field ID	Data Type	Permission	Size	Description	Comments
DM2_PROTECT_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_AMBER_WARN_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_RED_STOP_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_MAL_FUNC_IND_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_FLASH_PROT_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2__FLASH_AMBER_WARN_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_FLASH_RED_STOP_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_FLASH_MAL_FUNC_IND_LAMP	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable

DM2_FMI	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_SPN_CON V_METHOD	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable
DM2_OC	DBu8	READ	1	Raw Data from J1939	Raw data value from J1939 is updated into this DB Variable

The user will need to set the two variables from the Application to request the DM messages, shown in the below code snippet,

```
/* Set the CAN channel address and enable the DM1 messages */
    val = 0x84;
    Set_DL(CAN1_DEST_ADD, (uint8_t *)&val);
    val = 1;
    Set_DL(SEND_CAN1_DM1, (uint8_t *)&val);
/* Set the CAN channel address and enable the DM2 messages */
    val = 0x84;
    Set_DL(CAN1_DEST_ADD, (uint8_t *)&val);
    val = 1;
    Set_DL(SEND_CAN1_DM2, (uint8_t *)&val);
```

One the SDK receives the above messages, the DM packets will be process from the J1939 stack and then the values updated in the DB. The below code snapshot shows how the DM variables can be accessed from the touch GFX application.

```
/* Get DM1 protect Lamp diagnostic data */
Get_DL(DM1_PROTECT_LAMP , (uint8_t*)&val);
```

7.14.13.2 J1939 Additional DM Support

The SDK also supports the user to add support for additional DM messages. To do the same the user will have to configure the stack.

The user can add the SPN definitions in the J1939Data_layer.h file as shown in the below snapshot.
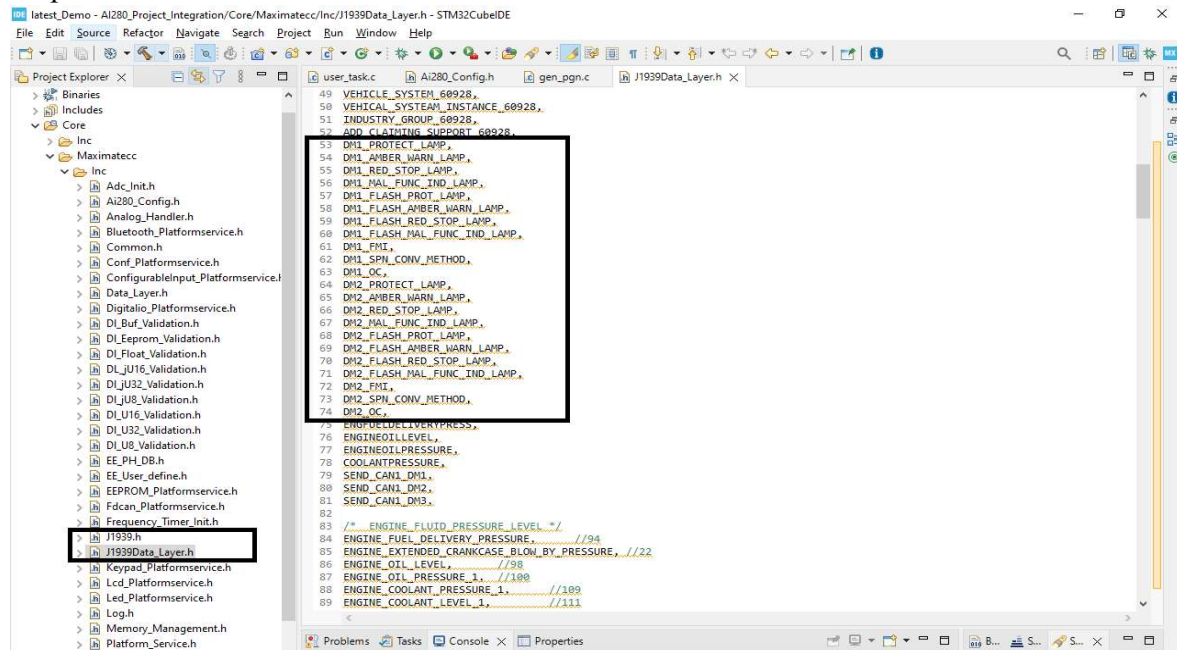
For example, to add the support for DM2, the following #defines were added in this file,

```
DM2_PROTECT_LAMP,
DM2_AMBER_WARN_LAMP,
DM2_RED_STOP_LAMP,
DM2_MAL_FUNC_IND_LAMP,
DM2_FLASH_PROT_LAMP,
DM2_FLASH_AMBER_WARN_LAMP,
DM2_FLASH_RED_STOP_LAMP,
DM2_FLASH_MAL_FUNC_IND_LAMP,
DM2_FMI,
DM2_SPN_CONV_METHOD,
DM2_OC,
```

The user will need to add the below variables so that the user can use these variables to enable the DM from the UI.
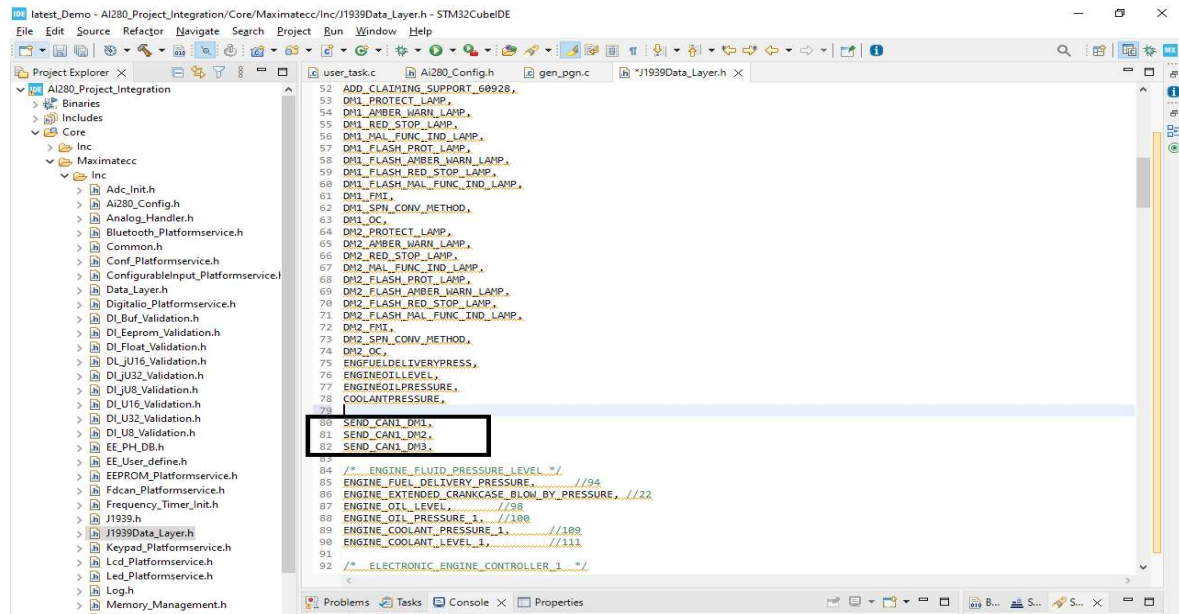
For example, to add the support for DM2, the following #defines were added in this file,

SEND_CAN1_DM2,

The next step would be to add a 32-bit entry as shown in the below image so that the access to the DM variables can be enabled.
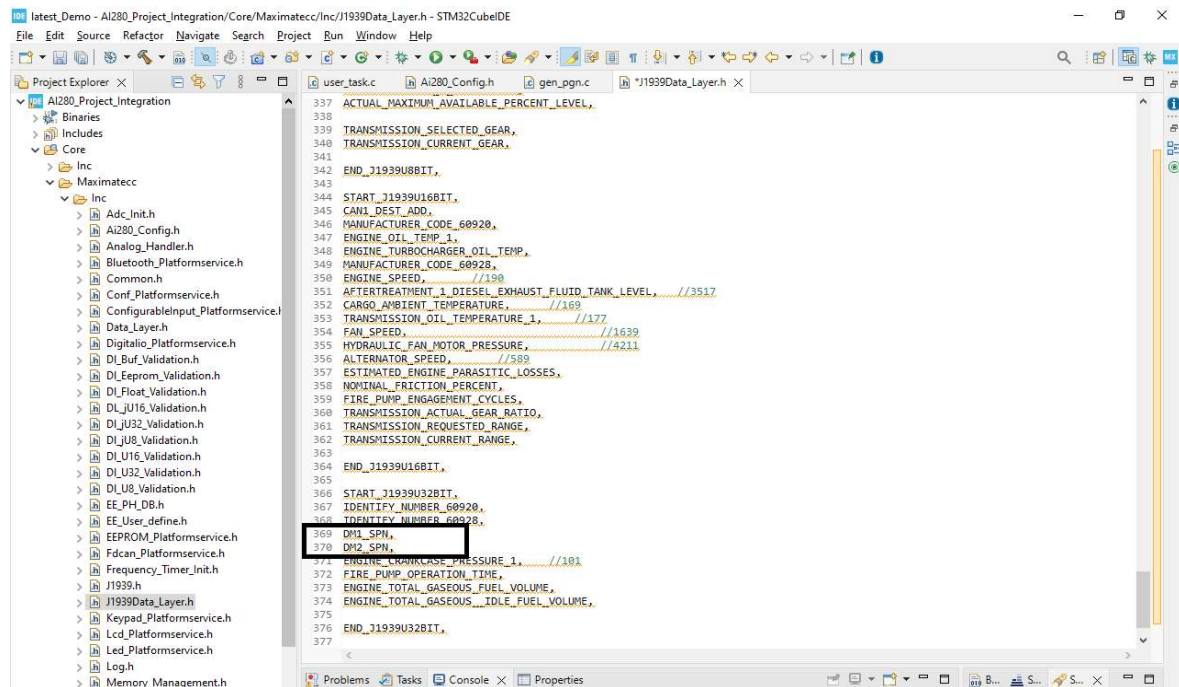
The next step would be to enable it in the code. Please navigate to j1939.c as shown in the below image and add the details of the new DM message in the **diagnosticReceive** API as shown below. The code snippet shows the entries for DM1 and DM2 messages.

```c
/********************************************************************/
/*
 * registered function for diagnostic requests
 */
static void diagnosticReceive(
            UNSIGNED8   canLine,
            UNSIGNED32  pgn,               /* PGN requested */
            UNSIGNED8   srcNode                /* requested node */
      )
{
RET_T retVal;
//UNSIGNED32      spn;
#if (EMTOS_J1939_DM1 == PS_ENABLE)
      if (pgn == J1939_PGN_DM1)  {
            printf("DM1 received\n");
            do {
                  retVal = j1939DiagnosticGet_DM1(canLine,
                        &dl_ju8[DM1_PROTECT_LAMP - (START_J1939U8BIT + 1)],
                        &dl_ju8[DM1_AMBER_WARN_LAMP - (START_J1939U8BIT +
1)],
                        &dl_ju8[DM1_RED_STOP_LAMP - (START_J1939U8BIT + 1)],
                        &dl_ju8[DM1_MAL_FUNC_IND_LAMP - (START_J1939U8BIT +
1)],
                        &dl_ju8[DM1_FLASH_PROT_LAMP - (START_J1939U8BIT +
1)],
                        &dl_ju8[DM1_FLASH_AMBER_WARN_LAMP - (START_J1939U8BIT
+ 1)],
                        &dl_ju8[DM1_FLASH_RED_STOP_LAMP - (START_J1939U8BIT +
1)],
                        &dl_ju8[DM1_FLASH_MAL_FUNC_IND_LAMP -
(START_J1939U8BIT + 1)],
                        &dl_ju32[DM1_SPN - (START_J1939U32BIT + 1)],
                        &dl_ju8[DM1_FMI - (START_J1939U8BIT + 1)],
                        &dl_ju8[DM1_SPN_CONV_METHOD - (START_J1939U8BIT +
1)],
                        &dl_ju8[DM1_OC - (START_J1939U8BIT + 1)]);
            } while (retVal == RET_SERVICE_BUSY);
#if (EMTOS_J1939_DM2 == PS_ENABLE)
            //j1939RequestPgn(canLine, J1939_PGN_DM2, 0x84);
#endif
```

```
        }
#endif
#if (EMTOS_J1939_DM2 == PS_ENABLE)
      if (pgn == J1939_PGN_DM2)  {
            printf("DM2 received\n");

            do {
                retVal = j1939DiagnosticGet_DM2(canLine,
                     &dl_ju8[DM2_PROTECT_LAMP - (START_J1939U8BIT + 1)],
                     &dl_ju8[DM2_AMBER_WARN_LAMP - (START_J1939U8BIT +
1)],
                     &dl_ju8[DM2_RED_STOP_LAMP - (START_J1939U8BIT + 1)],
                     &dl_ju8[DM2_MAL_FUNC_IND_LAMP - (START_J1939U8BIT +
1)],
                     &dl_ju8[DM2_FLASH_PROT_LAMP - (START_J1939U8BIT +
1)],
                     &dl_ju8[DM2_FLASH_AMBER_WARN_LAMP - (START_J1939U8BIT
+ 1)],
                     &dl_ju8[DM2_FLASH_RED_STOP_LAMP - (START_J1939U8BIT +
1)],
                     &dl_ju8[DM2_FLASH_MAL_FUNC_IND_LAMP -
(START_J1939U8BIT + 1)],
                     &dl_ju32[DM2_SPN - (START_J1939U32BIT + 1)],
                     &dl_ju8[DM2_FMI - (START_J1939U8BIT + 1)],
                     &dl_ju8[DM2_SPN_CONV_METHOD - (START_J1939U8BIT +
1)],
                     &dl_ju8[DM2_OC - (START_J1939U8BIT + 1)]);
            } while (retVal == RET_SERVICE_BUSY);
      }
#endif
}
```

Once this is enabled, the SDK will capture the new DM packets and store them in the DB. The TouchGFX user can access them from the DB as described in the previous section.

7.14.13.3   J1939 DM1 API Configuration

The below API support has been enabled in the J1939 file to be get the DM messages.

7.14.14 J1939 Sample Configuration

```
/***********************************************************************
*
 *
 *                         J1939 Module
 *


 ***********************************************************************
/
/*!
 * J1939 Platform service (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define SDK_SERVICE_J1939                              PS_ENABLE
//PS_DISABLE
#if ((SDK_SERVICE_J1939 == PS_ENABLE) && (SDK_SERVICE_FDCAN == PS_ENABLE))
#undef SDK_SERVICE_FDCAN
#define SDK_SERVICE_FDCAN                              PS_DISABLE
#endif
#if (SDK_SERVICE_J1939 == PS_ENABLE)
/*!
 * J1939 Task Periodicity          100ms
 */
define PS_J1939_TASK_TIMEOUT                           100
```

```
/*!
 * J1939 Task Priority
 * osPriorityNone        =  0,
 * osPriorityIdle        =  1,
 * osPriorityLow         =  8,
 * osPriorityLow1        =  8+1,
 *          ,,                        ,,
 *          ,,                        ,,
 * osPriorityISR         = 56,
 * osPriorityError       = -1,
 * osPriorityReserved    = 0x7FFFFFFF
 */
#define PS_J1939_TASK_PRIORITY                        osPriorityIdle
/*!
 * J1939_CLAIM_ADDRESS
 */
#define J1939_CLAIM_ADDRESS                      PS_ENABLE
/*
 * State BLE J1939 Debug data
 */
#define BLE_J1939_DEBUG_DATA                     PS_ENABLE
#define J1939_CAN1_ENABLE                        PS_ENABLE
/*!
 * J1939_CLAIM_ADDRESS will be between 0 to 255
 */
#define J1939_CAN1_CLAIM_ADDRESS                     23
/*!
 * J1939_ADDRESS_CLAIM_DYNAMIC will search address when set to 1, Fixed = 0
 */
#define J1939_CAN1_ADDRESS_CLAIM_DYNAMIC       1
/*!
 * J1939_ADDRESS_CLAIM_NEXT_ADDRESS When dynamic set to 1, search will
starting from this value and up
 */
#define J1939_CAN1_ADDRESS_CLAIM_NEXT_ADDRESS      80
/*!
 * J1939_CAN1_BITRATE will be between 0/250/500/1000
 */
#define J1939_CAN1_BITRATE                      250
/*!
 * J1939 ENABLE/Disable DMx
 */
#define EMTOS_J1939_DM1                            PS_ENABLE
#define EMTOS_J1939_DM2                            PS_ENABLE
```

```
//#define EMTOS_J1939_DM11                              PS_ENABLE
/*!
 * J1939 Set number of PGN'S to receive and transmit
 */
#define CAN1_NUMBER_PGNS_RX                     18u
#define CAN1_NUMBER_PGNS_TX                     1u
/*!
 * J1939 Set COB handlers
 */
#define CAN1_CO_COB_COUNTS                      36u
#endif      //SDK_SERVICE_J1939
```

7.15 Throughput Module

The User would be able to use the below functionalities of the Throughput module via the DB variables and configuration file.

7.15.1 Throughput Enable/Disable

The SDK provides the user the ability to enable/disable the throughput functionality by modifying the default configuration file. Please see section 7.15.4   for sample configuration.

Sr. No	Variables	Options	Default State	Description
1	THROUGH_PUT-SERVICE	PS_ENABLE/ PS_DISABLE	PS_ENABLE	**PS_ENABLE:** Enables the Throughput module in the SDK **PS_DISABLE:** Disables the Throughput module in the SDK

7.15.2 Throughput Absolute Time

During Run time, the below DB variables are used for displaying the absolute time of each software module.

Field ID	Data Type	Permission	Size	Options	Description
KEYPAD_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
DIO_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
CI_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
POWERMODE_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
LED_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time

POWER_MONITOR_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
DEFAULT_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
BLUETOOTH_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
RTC_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
SW_TIMER_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
EEPROM_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
WATCHDOG_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
LCD_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
CAN_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time based on their application
J1939_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time
TOUCHGFX_ABSTIME	DBu32	READ	1	0-FFFFFFFF	The total time that the task has actually been executing is the Absolute time

7.15.3 Throughput Percentage Time

During Run time, the below DB variables are used for displaying the Percentage time of each software module.

Field ID	Data Type	Permission	Size	Options	Description
KEYPAD_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
DIO_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
CI_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
POWERMODE_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
LED_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
POWER_MONITOR_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
DEFAULT_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
BLUETOOTH_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
RTC_PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time

SW_TIMER _ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
EEPROM_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
WATCHDO G_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
LCD_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time Processing Time
CAN_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
J1939_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time
TOUCHGF X_ PERTIME	DBu8	READ	1	0-100	Provides the Percentage of the total Processing Time rather than an absolute time

7.15.4 Throughput Sample Configuration

```
/*!
 * Through Put service (PS_ENABLE) / Disable (PS_DISABLE) Macros
 */
#define THROUGH_PUT_SERVICE    PS_ENABLE
#if (THROUGH_PUT_SERVICE == PS_ENABLE)
        #endif //THROUGH_PUT_SERVICE
```

8 Application Details

The MAXAI280 SDK S/W release package includes a sample application per module which demonstrates the functionalities of the modules and can be used for reference by the user to understand how to interact with the SDK.

It also includes a graphical demo application which can be used as a reference to understand how to use multiple modules in a single application and tie them to various graphical UI elements that are available in the touch GFX screen.

8.1 Sample Application Project Details

This sample application per module gives you a walk-through of the test procedure for each module available. The sample application can be used as a basis to understand what functionalities are available in each module and how the user can interact with the individual modules.

This section includes a brief description about sample application user interface and the minute details of each module, which includes module description, module screen navigation and the test procedures.

8.1.1 Introduction

To open the sample application project please follow the same procedure followed to open the blank project file as described in Section 3.

Once you have successfully compiled and flashed the Sample project on the AI280 hardware, you can reboot the hardware to run the application.

Initially after the device is turned ON, A Home Screen will be displayed on the LCD screen which contains the list of all the available modules. The Home Screen sample image is given below:

8.1.2 Home Screen Navigation:

In the Home Screen you can find the list of all the modules available in the AI280 SDK i.e., Keypad, Power Monitor, RTC, EEPROM, LCD, Digital o\p, SW Timer, Config input, LED, BLE, Power Mode, J1939, Flash, through put, WD and CAN.

Along with the modules there are four key navigators (i.e., Previous, Next, Enter, Back) which will allow the user to move front, back, up and down, enter into a specific module and exit from the specific module. The four key navigators are operated using the four built-in buttons, which are located to both sides of the screen (left and right) and are represented as Key1, Key2, Key3, and Key4 respectively. Each button has a specific functionality which is mentioned in below table.

Button Name	Functionality
Key1	Go to previous
Key2	Go to next
Key3	Enter/Select
Key4	Back

8.1.3 Keypad Module

The keypad module sample application is shown in the below screen,
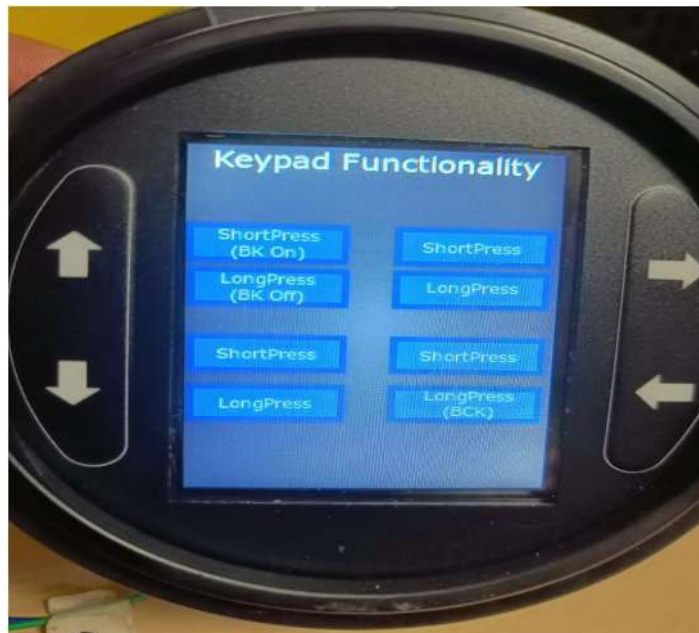
8.1.3.1 Module Description:

This Keypad module is basically designed to test the functioning of all the four keys present on the device. The testing can be done for each key (i.e., All the four keys) to check whether it's working properly based on their specific functionality.

8.1.3.2 Module Navigation:

To go to the Keypad module, In the Home Screen navigate to the keypad module using Key 1 and Key 2 and then select the keypad using Key3 i.e., Enter, which will take you to Keypad Functional screen.

On this Screen you will find four Keypad options i.e., Keypad1, Keypad2, Keypad3 and Keypad4. Each Keypad consists of two Blocks below them, where one block is used for short press test update and the other Block is used for long press test update.
Note: Short Press denotes a single click on the key.
       Long Press denotes click and hold the key.

8.1.3.3 Module Test Procedure:

Test Case	Keypad Action	Description	Test Procedure	Expected Result
KeyPad 1	Short Press	This key is used to turn ON the Backlight which are present on the keys	Click on key1 and check if the light on all the keys turns ON.	Light on the button Glows
KeyPad 1	Long Press	This key is used to turn OFF the Backlight which are present on the keys	Click and Hold Key1. Then check if the light on all the keys turns OFF.	Light on the button Turns OFF
KeyPad 1	Continuous Press	This key is used to turn OFF the Backlight which are present on the keys	Click and Hold Key1. Then check if the light on all the keys turns OFF.	Light on the button Turns OFF
KeyPad 2	Short Press	This is specifically used to check whether the key is functioning properly.	Click on Key2 and then check if the short press block below key2 gets highlighted	Keypad2 Short press block present on the LCD will be Highlighted
KeyPad 2	Long Press	This is specifically used to check whether the key is functioning properly.	Click and Hold on Key2.Then check if long press block below key2 gets highlighted	Long press block present on the LCD will be Highlighted

KeyPad 2	Continuous Press	This is specifically used to check whether the key is functioning properly.	Click and Hold on Key2.Then check if long press block below key2 gets highlighted	Continuous press block present on the LCD will be Highlighted
KeyPad 3	Short Press	This is specifically used to check whether the key is functioning properly.	Click on Key3 and then check if the short press block below key3 gets highlighted	Short press block present on the LCD will be Highlighted
KeyPad 3	Long Press	This is specifically used to check whether the key is functioning properly.	Click and Hold on Key3.Then check if long press block below key3 gets highlighted	Long press block present on the LCD will be Highlighted
KeyPad 3	Continuous Press	This is specifically used to check whether the key is functioning properly.	Click and Hold on Key3.Then check if long press block below key3 gets highlighted	Continuous press block present on the LCD will be Highlighted
KeyPad 4	Short Press	This is specifically used to check whether the key is functioning properly.	Click on Key4 and then check if the short press block below key4 gets highlighted	Short press block present on the LCD will be Highlighted
KeyPad 4	Long Press	This key is used to Exit from the Keypad Functionality screen and go to Home Screen	Click and Hold on Key3.Then check if the Screen gets exited from Keypad functionality and goes to home screen.	Back\Exit
KeyPad 4	Continuous Press	This key is used to Exit from the Keypad Functionality screen and go to Home Screen	Click and Hold on Key3.Then check if the Screen gets exited from Keypad functionality and goes to home screen.	Back\Exit

8.1.4   Power Monitor

The power monitor module sample application is shown in the below screen,

8.1.4.1 Module Description:

This Power Monitor test is basically used to provide the temperature, battery level and ignition status updates.

8.1.4.2 Module Navigation:

To go to Power Monitor Test screen, from Sample Application Screen navigate to Power Monitor block using Key1 and Key2, then by using Key3 enter into the Power Monitor Test Screen

In Power Monitor Test screen there are 3 blocks available they are:

1) IGNITION STATE. (IGN_STATE)
2) TEMPERATURE LEVEL. (TEMP_LEVEL)
3) BATTERY LEVEL. (BAT_LEVEL)

8.1.4.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result	Units
IGNITION STATE	This functionality is used for testing whether the external device supply is ON or OFF	For checking ignition value, the user will need to provide some external supply. And then by using Key1 navigate to IGN_STATE and then click on Key3 to test	If the external supply provided is turned On then it will give ON or else OFF	-
TEMPERATURE LEVEL	This functionality provides the update about the temperature level present in the device area.	Using Key1 and Key2 navigate to TEMP_LEVEL then click on Key3 (Enter). Now check the result that is displayed at the blank space.	The room temperature where the device is located will be updated.	Celsius
BATTERY LEVEL.	This functionality updates the battery level of the device.	Using Key1and Key2 navigate to BAT_LEVEL then click on Key3 (Enter). Now check the result that is displayed at the blank space.	The battery level of the device will be displayed.	mv

8.1.5   RTC

The RTC module sample application is shown in the below screen,

8.1.5.1 Module Description:

RTC is the real time clock which supports the RTC configuration for the AI280 board and also supports two alarms.

8.1.5.2 Module Navigation:

To go to RTC, from "Sample Application" navigate to RTC using key1 and Key2 then click on key3 to enter into RTC screen.

RTC has the three blocks as given below:

1) RTC Time
2) ALARM_A
3) ALARM_B

8.1.5.3 Sub Screens:

The three blocks contain Sub Screens which are used to set the time or alarm. These sub screens are,

RTC screen: The RTC screen consist of HR: MIN:SEC: WKDAYS:DD:MM: YY blocks which is used to set the Hour's, Minutes, Seconds, Week Days, Date, Month, Year respectively using the Key1 and Key4 Short Press. And to display it on the main Screen of RTC long press Key4.

ALARM_A screen: The ALARM_A screen is routed by selecting ALARM_A from RTC main screen. This screen consists of HR: MIN:SEC: WK_DAY, which are again for Hour's, Minutes, Seconds, Week Days respectively and can be set using the Key1 and key4 Short Press. And there is an empty block available below which Displays ALARM_OCCURED message when ALARM_A is triggered in the platform.

ALARM_B screen: The ALARM_B screen is routed by selecting ALARM_B from RTC main screen. This screen consists of HR: MIN:SEC: WK_DAY, which are again for Hour's, Minutes, Seconds, Week Days respectively and can be set using the Key1 and key4 Short Press. And there is an empty block available below which Displays ALARM_OCCURED message when ALARM_B is triggered in the platform.

8.1.5.4 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
RTC Time	This functionality is used to set the time of the device	Using key1 navigate to RTC Time block and then enter using key3.Now in the Sub Screen, by using Key1 set all the required details then by a short press on key4 you can set the time and later give a long press exit from the present screen and go to RCT main screen.	The updated time will be displayed on the screen.
ALARM_A	This functionality is used to set alarm on the device	Using key1 navigate to Alarm A block and then enter using key3.Now in the Sub Screen, by using Key1 set all the required details and then by a short press on Key4 set the alarm.	When the alarm gets triggered in the platform the empty Block under ALARM_A will display ALARM_OCCUR ED message
ALARM_B	This functionality is used to set alarm on the device	Using key1 navigate to Alarm B block and then enter using key3.Now in the Sub Screen, by using Key1 set all the required details and then by a short press on Key4 set the alarm.	When the alarm gets triggered in the platform then the empty Block under ALARM_B will display ALARM_OCCUR ED message.

8.1.6 LCD

The LCD module sample application is shown in the below screen,

8.1.6.1 Module Description:

LCD functionality is basically designed to check if the screen is functioning properly. The verification is done based on two factors one is by turning ON and OFF the screen and the other by increasing or decreasing the brightness of the screen.

8.1.6.2 Module Navigation:

To go to LCD test, from "Sample Application" screen select LCD using Key1 and Key2 and later enter into the LCD Test Screen by using Key3. In the LCD Test there is a session which is used to update the brightness value of the display.

8.1.6.3 Module Test Procedure:

Test Case	Keypad Action	Description	Test Procedure	Expected Result
Brightness Value	Increase	This functionality is used to increase the brightness of the Device	Click on key1(Brightness+) and check the results	The brightness value would be increase from the previous value
Brightness Value	Decrease	This functionality is used to decrease the brightness of the Device	Click on key2(Brightness-) and check the results	The brightness value would be decrease from the previous value
LCD ON\OFF	-	This functionality is used to Turn ON/OFF the LCD Screen	Click on the ON\OFF button given on the LCD Screen.	The LCD display will turn ON in case of ON and Turn Off in case of OFF

8.1.7 Digital Output

The digital output module sample application is shown in the below screen,

8.1.7.1 Module Description:

There are two digital pins in the device digital high and digital low. This Digital Output module is used to set this pin values.

8.1.7.2 Module Navigation:

To go to the Digital Output screen, from "Sample Application" Screen select Digital Output using Key1 and Key2, then enter the Digital Output screen using key3.On the Digital output screen there are 4 blocks they are as given below:

High Side
Low Side
Status ON
Status OFF
Note: There are three configurable modes available in Digital output they

Mode	High pin	Low pin
High side	1	0
Low side	0	1
Open Drive	0	0

8.1.7.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result

Status OFF	This functionality is used to set the system into Open Drive configuration mode	Using Key1 and Key2 navigate to Status OFF block and click on Key3. Now check the results	The system will go to OPEN Drive state which means the high pin and low ping will be low.
Status ON	This functionality is used to set the Digital output pin state to the previous state which was present before the system was set as OFF status	Using Key1 and Key2 navigate to Status OFF block and click on Key3. Now check the results	The pin status gets updated according to the previous state.
High side	This is used to show what pin state the system is operation on at present.	This will be tested along with status ON\OFF	High Side block in the display will get highlighted and the high pin will be 1 and low pin will be 0.
Low side	This is used to show what pin state the system is operation on at present	This will be tested along with status ON\OFF	Low Side block in the display will get highlighted and low pin will be 1 and high pin will be 0.
Open Drive	This is used to set the system into Open Drive	This will be tested along with status ON\OFF	The system will go to OPEN Drive state which means the high pin and low ping will be low.

8.1.8   Software Timer

The software timer module sample application is shown in the below screen,

8.1.8.1 Module Description:

The software timer is mainly designed to interact with the timer module. This will be consisting of timer reset, timer trigger and also to read the current timer counter value. There are a total of six timers available from Software Timer1 to Software timer6.

8.1.8.2 Module Navigation:

To go to Software Timer, from "Sample Application" screen navigate to Software Timer Using Key1 and Key2, later click on Key3 to enter to Software Timer Test screen. In this screen there will be six blocks available which are used for six different timers.

8.1.8.3 Sub Screen:

Sub Screen_1: From Software test screen when a timer is selected it will be redirected to Sub Screen_1 which has a name of the related Timer number. In this screen there are two blocks available they are:

1) Single Shot
2) Inactive
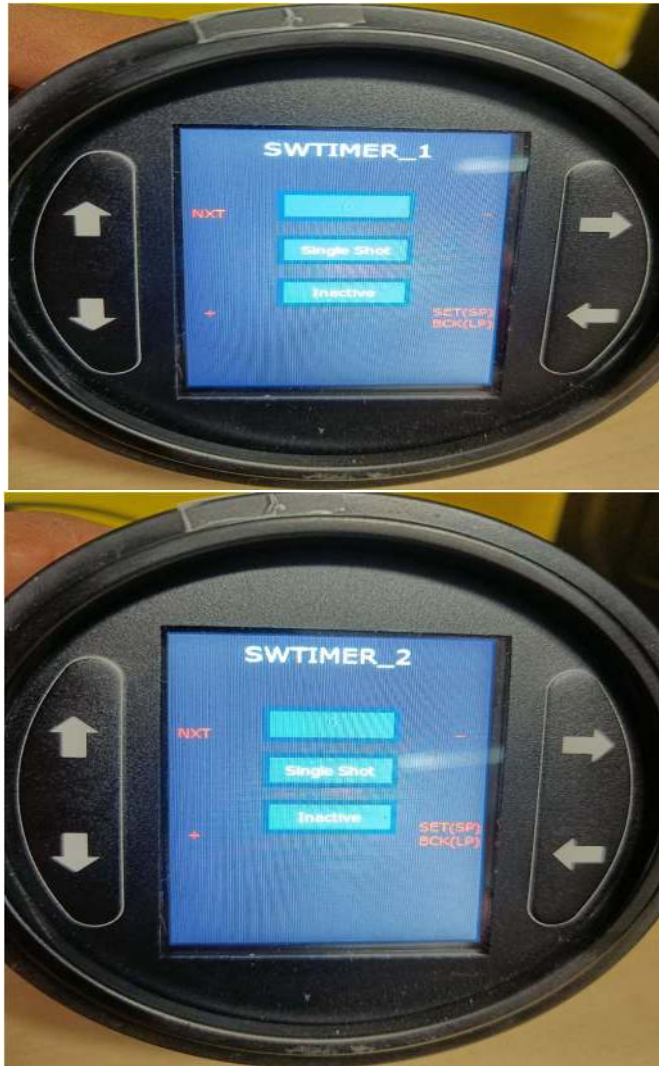
8.1.8.4 Module Test Procedure:

Test Case	Keypad Action	Description	Test Procedure	Expected Result
Single/ Inactive	Short press	This functionality is used to stop the timer	User can navigate to the timer using key1(NEXT) or key2(PREVIOUS) and can select the any one timer by pressing key3(ENTER), User can go back to previous screen using key 4(BACK).	Timer gets stopped

			Once the user navigates to the SW Timer 1, and press the key3 (ENTER), SWTIMER_1 GUI screen appears, where user can set the timer according to the requirement.	
Continuous/ Active	Long Press	This functionality is used to set single shot timer and inactive state	User can navigate to the timer using key1(NEXT) or key2(PREVIOUS) and can select the any one timer by pressing key3(ENTER), User can go back to previous screen using key 4(BACK). Once the user navigates to the SW Timer 1, and press the key3 (ENTER), SWTIMER_1 GUI screen appears, where user can set the timer according to the requirement.	Timer will be started.

8.1.9 Configurable Inputs

The below screen shot shows the configurable input screen.

8.1.9.1 Module Description:

The Configurable input are basically designed to configure the various input channels and then read the latest Configurable Input values according to the initial configuration done. The channels which are available for configuring are voltage, resistance, digital STB, digital STG, resistance and frequency. The MAXAI280 supports 5 configurable inputs.

8.1.9.2 Module Navigation:

To go to configurable Input, from "Sample Application" screen navigate to Config Input block using Key1 and Key2, now using Key3 enter to Configurable Input Test Screen.

In Configurable Inputs test screen, there are five blocks available which has five AI values from AI1 Value to AI5 Value. This block represents different input channels they are as given below:

1) Voltage
2) Digital STB
3) Digital STG
4) Resistance
5) Frequency

Note: From AI1 to AI5 the configurable inputs can be assigned to below mentioned channel:

1) Voltage_32v

2) Voltage_6v
3) Frequency
4) Resistance
5) Digital STG
6) Digital STB

8.1.9.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
AI1 to AI5	AI1 to AI5 are values which are assigned to the configurable input channels 1 to 5 that are available for configurable inputs functionality	Using Key1(Long press) navigate to the specific block from AI1 to AI5 then by using Key1(Short press select the desired channel) and then check UI above the channel for the updated value based on the configuration. If you navigate to AI1 you will be able to configure it as resistance, voltage low and high, frequency, digital input. For example, if you configure it as resistance, you will see the input resistance value on the screen.	The AI value of the specified channel will be displayed on the screen

8.1.10 LED

8.1.10.1 Module Description:

The LED functionality is designed to test the LED light which is In-Built in the device. The LED testing can be done in three modes they are:

1) Turn On
2) Turn OFF
3) Blink

The MAXAI280 supports 2 LEDs. They are,

1) RED
2) AMBER

8.1.10.2 Module Navigation:

To go to LED screen, from "Sample Application" screen navigate to LED using Key1 and Key2.Later enter into the LED test screen using Key3. In this screen there are two functionalities available, they are:

1) RED LED
2) AMB LED

8.1.10.3 Sub Screen:

The two functionalities which are available in LED test screen, are again having individual sub screen, they are explained in detail below:

Sub Screen_1: When the user selects from RED LED and AMB LED, Sub Screen_1 will be opened which again has two functionalities, they are:

1) ON/OFF
2) BLINK

Sub Screen_2: When the user selects the ON and OFF functionality then Sub Screen_2 will be displayed on this screen it updates if the LED is ON/OFF.

Sub Screen_3: When the user selects the Blink functionality then Sub Screen_3 will be displayed on this screen it updates if it's turned On and at what speed will the LED blink.

8.1.10.4 Module Test Procedure:

Test Case	Mode	Description	Test Procedure	Expected Result
RED LED	ON/OFF	This functionality is used to Turn ON the RED LED present on the Bottom right Corner of the device	From the main Screen (LED test Screen) select RED LED Using Key1 and Key2, user will be navigated to Sub Screen_1 there select ON\OFF using Key1 and Key2, Now the user will be redirected to Sub Screen_2, now using Key1 turn On /OFF the LED based on requirement.	If the User Selects ON mode, then the LED light Glow's. And if the user selects OFF mode, then the LED light which was glowing will turn OFF
RED LED	BLINK	This functionality is used to blink the LED.	From the main Screen (LED test Screen) select RED LED Using Key1 and Key2, user will be navigated to Sub Screen_1 there select Blink using Key1 and Key2, Now the user will be redirected to Sub Screen_2, now	If the User selects the Blink On mode, then the LED starts blinking at the specified rate.

			using key2 and Key3 increase or decrease the blink speed and Using Key1 set Blink ON\OFF	
AMB LED	ON/OFF	This functionality is used to Turn ON the AMB LED present on the Bottom right Corner of the device	From the main Screen (LED test Screen) select AMB LED Using Key1 and Key2, user will be navigated to Sub Screen_1 there select ON\OFF using Key1 and Key2, Now the user will be redirected to Sub Screen_2, now using Key1 turn ON /OFF the LED based on requirement.	If the User Selects ON mode, then the LED light Glow's. And if the user selects OFF mode, then the LED light which was glowing will turn OFF
AMB LED	BLINK	This functionality is used to blink the LED.	From the main Screen (LED test Screen) select AMB LED Using Key1 and Key2, user will be navigated to Sub Screen_1 there select Blink using Key1 and Key2, Now the user will be redirected to Sub Screen_2, now using key2 and Key3 increase or decrease the blink speed and Using Key1 set Blink ON\OFF	If the User selects the Blink On mode, then the LED starts blinking at the specified rate.
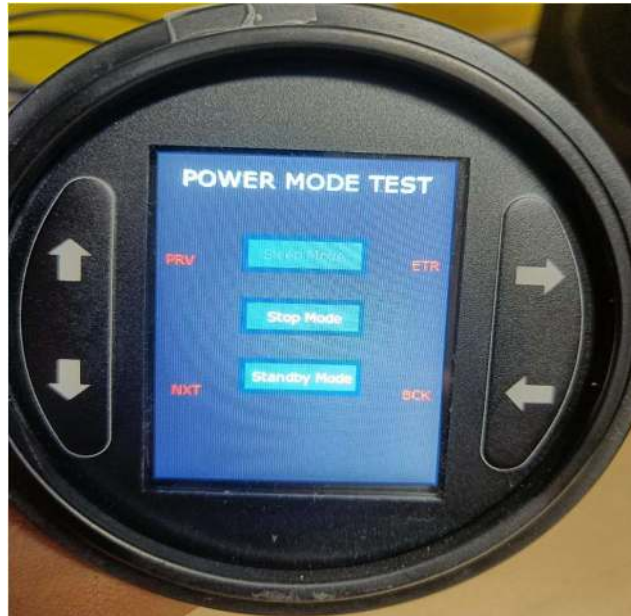
8.1.11 Power Mode

8.1.11.1 Module Description:

The Power Mode Module is basically designed to check the low power functional mode of the device. There are three different functionality modes available for Power Mode Module which indicates the present state of the device. Currently the AI280 devices supports only the Stop mode and the user can enter/exit the stop mode functionality based on their requirement.

To exit the stop mode functionality the user can configure one of the below inputs,

1) RTC
2) Keypad
3) Ignition

8.1.11.2 Module Navigation:

To go to the Power Mode Test, from "Sample Application" screen select Power Mode Test by using Key1 and Key2, after that enter into Power Mode Test Screen by using Key3. In the Power Mode Test user can test the Stop mode functionality. This would enable the user to enter the low power mode.

8.1.11.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
Stop Mode	This functionality is used to stop all the functions inside the system and it is waiting into the same mode until an interrupt will occur and activate the device.	Using Key1 and Key2 navigate us to Stop Mode; now by pressing Key3 will set the device into Stop mode.	The running functionality will be kept on halt, and the system will be set to Stop mode i.e., system will turn OFF
Keypad	This is a wake-up source which is used to activate the system from the power mode	Click on any one of the keys then the system will turn on Note: Keypad 4 is not working	The device will be activated back
RTC	This is a wake-up source which is used to activate the system from the power mode	The user needs to set a timer (After how long the system has to restart) in the configuration file and wait till the timer reached the desired time gap.	The device will be activated back
Ignition	This is a wake-up source which is used to activate the system from the power mode	When there is any activity performed in the power monitor module then the system will be activated	The device will be activated back
CAN	This is a wake-up source which is used to activate the system from the power mode	The CAN must send the signals to the device in case the user wants to activate the system using CAN	The device will be activated back

8.1.12 EEPROM

The EEPROM module sample application is shown in the below screen,
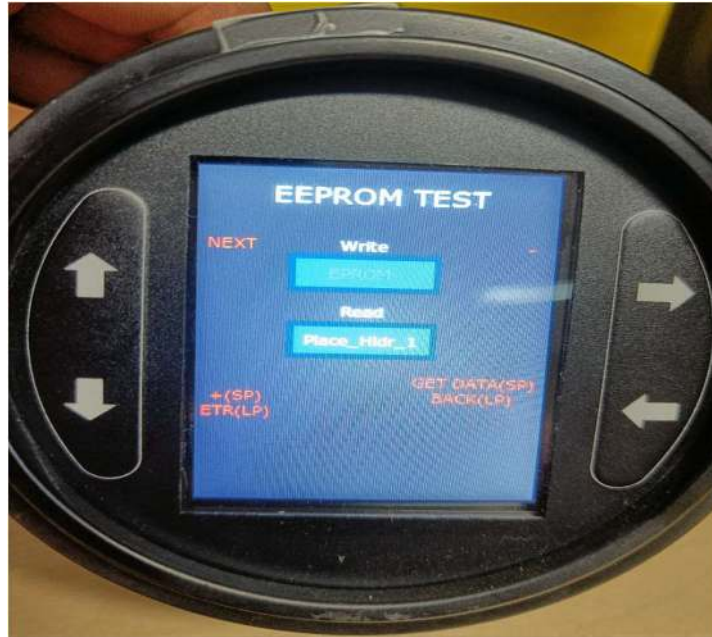
8.1.12.1 Module Description:

The functionality is basically designed to store the data in EEPROM memory and read the same when required. The user will provide the input value to the place holder and output will be stored into the EEPROM memory. User can provide values up to 65535 place holders.

8.1.12.2 Module Navigation:

To go to the EEPROM Test, from "Sample Application" screen select EEPROM Test by using Key1 and Key2, after that enter into the EEPROM Test Screen by using Key3. In the EEPROM Test user can interact with the EEPROM module for below functionality.

1) To write and store the data in EEPROM memory.
2) To read the stored data from EEPROM memory.

8.1.12.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
Write Mode	This functionality is used to write and store the data into the EEPROM memory.	Using Key1 and Key2 navigates to Write Mode; now by pressing Key3 will set the device to the write mode.	The input data provided at the place holder will be stored at the EEPROM memory.
Read Mode	This functionality is used to read the stored data from the EEPROM memory.	Using Key1 and Key2 navigates to Read Mode; now by pressing Key3 will set the device to the read mode.	User can read the data stored at the EEPROM memory.

8.1.13 Watchdog

The watchdog module sample application is shown in the below screen,

8.1.13.1 Module Description:

The functionality is basically designed to monitor the state of the device. Watchdog reset depends on the Pre-scaler. The Pre-scaler value will be provided within the range of 4 to 256.

Once the user provides the pre-scaler value 256, the system will reset after 50 seconds. User has to go to the config.h file to enable or disable any property on the Board.

8.1.13.2 Module Navigation:

To go to the Watchdog Test, from "Sample Application" screen select Watchdog Test by using Key1 and Key2, after that enter into the Watchdog Test Screen by using Key3. The user can select any watchdog (WD0 – WD9). In the Watchdog Test user can interact with the Watchdog module for below functionality.

1) To enable/disable the watchdog functionality.
2) To feed the watchdog manually.

8.1.13.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
Watchdog	This functionality is used to enable a property on the system.	Using Key1 and Key2 navigates to Watchdog Mode; now by pressing Key3 will select the specific Watchdog and see whether it is enabled or disabled.	Based on the selected Watchdog, whatever the default state is available (Enabled/Disabled) for the specific watchdog, that will be updated to the system.
Watchdog Ping	This functionality is used to feed or refresh the task after every second.	Once the user selects any watchdog from WD0 to WD9 and hit stop ping, it will stop the feeding.	Hardware will go into reset mode after a few seconds.

8.1.14 BLE

8.1.14.1 Module Description:

The functionality is basically designed to establish a BLE connection between two devices and send/receive data. To communicate using Bluetooth connection between the two devices, users can pass the BLE terminal commands and get the acknowledgement on the same screen.

8.1.14.2 Module Navigation:

To go to the BLE Test, from "Sample Application" screen select BLE Test by using Key1 and Key2, after that enter into the BLE Test Screen by using Key3. In the BLE Test, user can interact with the BLE module for below functionality:

1) Scan
2) Connect
3) Read
4) Write

8.1.14.3 Steps to Scan and Connect:

The User can use the BLE Application through mobile phone using below mentioned steps.

Step 1: Once the user opens the BLE application, they need to click on START SCANNING. Here the user will find MAX device name advertised as "maxAI12345678" as shown below.

Users can connect these devices by clicking on the CONNECT option. Once it is done the user will navigate to the SERVICE screen.
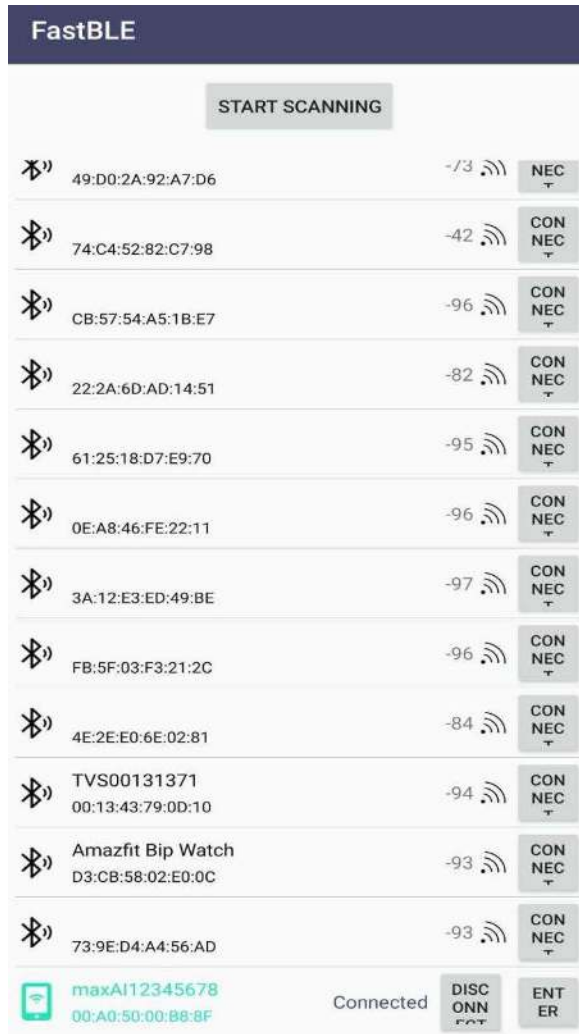
Step 2: In SERVICE Screen the user will find Service (0). Once the user will select the Service (0), it will move to the characteristic screen as shown below:
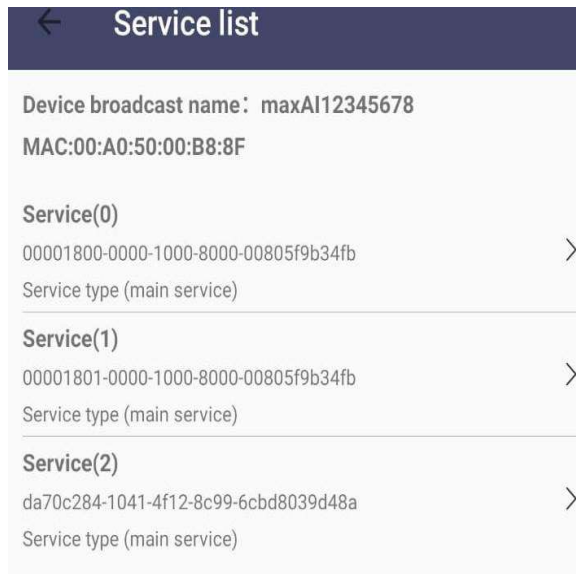
Step 3: In the characteristic screen user will find characteristic (0). Once the user selects the characteristic (0), the user will move to BLE terminal screen as shown below.
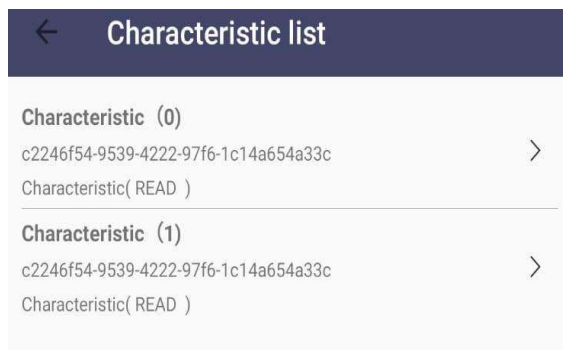
Step 4: In the BLE Terminal Screen, user will be able to write or read data from the MAXAI280 device using the BLE.

8.1.14.4 Steps to Read and Write:

Step 1: The user will write the message in the mobile application in the form of Hexadecimal and that message will be sent to the GUI

Step 2: That message will be received to the GUI screen which will be as show below

Step 3: If the user wants to send the message from the GUI, then, from the GUI user can select any of the available default messages using Key2 and later by clicking on Key3 user can set that message.

Step 4: Once the message is sent from the GUI the User has to click on Read button on the mobile application and the message will be displayed on the screen in Hexadecimal format.

8.1.14.5 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
Scan	This Functionality is used to detect the device and display the device name on the mobile application	User needs to open the FastBLE application in the phone, click on the START SCANNING option. User needs to check for the MAX device named "maxAI12345678". Once it is detected, then we need to click on "CONNECT" option. To establish a connection between two devices, the user needs to send the start request packet along with header packet attached.	Once the request packet is received from the user, system will send the positive response packet back to the user to give the confirmation message as the device was scanned successfully.
Connect	This functionality is used to establish a connection between the device and the mobile application	Once the device is detected through the mobile application, user need to connect the device by clicking on "Connect" which	A connection will be established between the mobile application and device

		is present on the mobile application	
Write	This functionality is designed such that when the user sends a hexadecimal data from the mobile application on that, data will be received to the GUI in string format.	User needs to enter the hexadecimal value at the write column which is present in the mobile application and check the result in the GUI	A string message will be sent to the GUI which will be the converted form of the hexadecimal message that is being sent from the mobile application.
Read	This functionality is used to send message from the GUI and will be received by the mobile application.	User can select the default messages (There are four default message) which has to be sent to the mobile application using key2( ) and set it by pressing key4(OK) Now in the mobile application, click on "Read" command and check the results.	The message sent from the GUI will be converted to hexadecimal and that message will be displayed on the mobile application.

8.1.15   CAN

8.1.15.1 Module Description:

The CAN is a control area network which is basically used to control the ECU (Electronic control unit). Can acts as a master controller and based on the functionality it sends the request and receives the messages.

8.1.15.2 Module Navigation:

To go to CAN, from "Sample Application" screen navigate to CAN using Key1 and Key2, and enter into CAN test Screen using Key3. In CAN test screen there are two blocks present they are:

CAN_1

8.1.15.3 Sub Screen

Sub Screen_1: When the user selects from CAN_1 in CAN test screen user will be redirected to Specified functionality test screen which will be out Sub Screen_1. In this Sub Screen_1 there are two packets available they are:
Send Pckt
Rx Pckt
Steps to send and Receive packets:
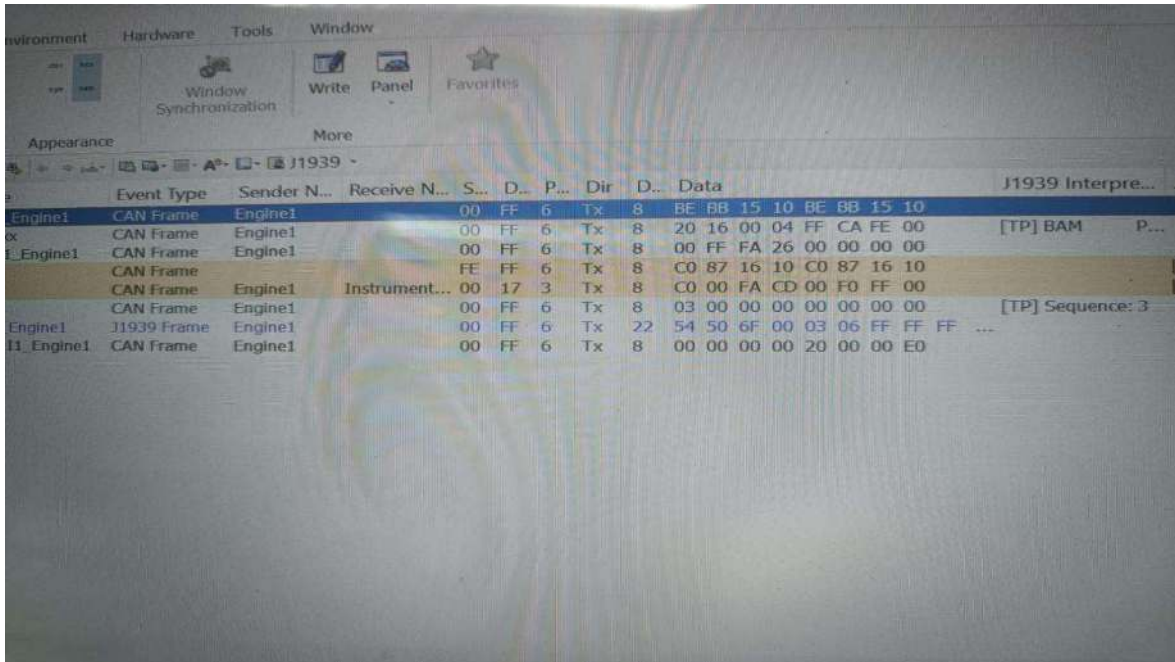
For CAN1:

Step 1: The required message will be sent in the form of a packet and the packet will be sent to the device via CAN Terminal.

Step 2: When the packet will be received by the device the packet message will be in hexadecimal format.

Step 3: User needs to select any of the message which is available by default using Key2 and Key3 and set it as the sending packet by clicking on Key4.

Step 4: The message which was sent from the device will be received at the CAN terminal as shown below.

8.1.15.4 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
CAN1	This Functionality is used to check the packet, which was sent from the externally connected CAN analyser.	In the CAN test screen go to CAN1 using key1 and Key2 then enter the CAN1 Testing screen using Key3. And check the result	The message packet which was sent from the CAN analyser will be will be received at the Rx Packet area in string format
CAN State	This functionality is used to read the state of the CAN	When the user clicks on Key1 for once then the CAN state will be read.	The was just written as an example for user to understand the usage of can state DB variable.
Filter Index/CAN mode	This functionality is used to update the Filter details and the mode in which is operation on.	When The user double clicks on Key1 then the CAN state will be updated.	The user can check if the CAN filter is enabled and the filter index is 20 and the standard mode is set in the DB for can channel 1.
CAN Baud rate	This functionality is used to set the device Baud rate	When The user clicks on Key1 for Tree times, then The CAN Baud rate function will be updated.	The user can check if the CAN 1 baud rate is updated to 250K.
CAN Drive Reset	This functionality is used to reinitialize the applicant.	When The user clicks on Key1 for four times, then The CAN Drive Reset function will be implemented.	The user can check this update on the DB variable
CAN Reset	This functionality is used to set the device into power down mode	When The user clicks on Key1 for five times, then The CAN Reset function will be implemented.	The user can check this update on the DB variable

8.1.16 J1939

8.1.16.1 Module Description:

J1939 module is used to interface with the J1939 stack and receive the PGN functionality values and update the values to the GUI. J1939 is also used to provide Diagnosis message to the user.

8.1.16.2 Module Navigation:

To go to JI939 functionality, from "Sample Application" screen navigate to J1939 using Key1 and Key2, later enter into J1939 screen using Key3.In J1939 test screen there are different PGN present they are:

1) EngFuelDeliveryPress
2) EngineOilLevel
3) EngineOilPressure
4) Coolant Pressure

There are three Diagnosis message available they are:

1) DM1
2) DM2

Sub Screen_1: When the user selects DM1 using Key1 in J1939 test Screen and enters to Sub Screen_1 by clicking Key3 then a screen consisting for eight diagnosis messages will be displayed which are listed as follows:

Sub Screen_2: When the user selects DM2 using Key1 in J1939 test Screen and enters to Sub Screen_1 by clicking Key3 then a screen consisting for four diagnosis messages will be displayed which are listed as follows:

Note: To go back to main screen (J1939 Test screen) User need to click on Key4.

8.1.16.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
DM1	DM as in Diagnosis message are provided for the user to send messages to the database through CAN	User can set to DM1 using Key2 and Key3 to enable the SDK to capture diagnostic message to the database through CAN which was updated to CAN via J1939	The SDK will start capturing the diagnostic message that's and update those on to the screen.

DM2	DM as in Diagnosis message are provided for the user to send messages to the database through CAN	User can set to DM2 using Key2 and Key3 to enable the SDK to capture diagnostic message to the database through CAN which was updated to CAN via J1939	The SDK will start capturing the diagnostic message that's and update those on to the screen.
EngFuelDeliveryPress	This functionality is used to provided data about the Engine Fuel	For testing purpose, the device can be connected to the external CAN analyser to receive this PNG values	The Fuel delivery pressure value will be displayed on the specified space in the UI.
EngineOilLevel	This functionality is used to provided data about the engine Oil level	For testing purpose, the device can be connected to the external CAN analyser to receive this PNG values	. The Oil level value will be displayed on the specified space in the UI.
EngineOilPressure	This functionality is used to provided data about the Oil Pressure	For testing purpose, the device can be connected to the external CAN analyser to receive this PNG values	The Oil pressure value will be displayed on the specified space in the UI.
Coolant Pressure	This functionality is used to provided data about the coolant pressure	For testing purpose, the device can be connected to the external CAN analyser to receive this PNG values	The Coolant pressure value will be displayed on the specified space in the UI.

8.1.17 Throughput

8.1.17.1 Module Description:

The main functionality of through put is to constantly update the absolute time and percentage time used by each module until the device is working

8.1.17.2 Module Navigation:

To go to Throughput, from "Sample Application" screen navigate to Throughput using Key1 and Key2. Later enter the throughput test screen using Key3. In this throughput test screen, all the modules are listed for which there are two functionalities which are being updated, they are:

1) Absolute Time 2) Percentage Time

8.1.17.3 Module Test Procedure:

Test Case	Description	Test Procedure	Expected Result
Absolute Time	This functionality gives the total 'time' that the task has been executing (the total time that the task has been in the Running state). It is up to the user	This functionality is designed to constantly update the Absolute Time used by the individual module in the UI without any intervention from the user.	Absolute time will be constantly updated

	to select a suitable time base for their application.		
Percentage Time	This functionality will provide essentially the same information but as a percentage of the total processing time rather than as an absolute time.	This functionality is designed to constantly update the Absolute Time used by the individual module in the UI without any intervention from the user.	Percentage will be constantly updated

8.2 Details of Demo Application

The Demo application is a combined TouchGFX application which will provide insight into how we can combine the different services of the SDK and write a wholesome application.

8.2.1 Difference between Sample Application and Demo Application

The Sample Application was written to help the AI280 SDK User to understand the functionalities of the individual modules and use them as per their requirement. The home screen helps navigate to all the available modules present on the "Sample Application" screen, which can be tested by entering into a specific module whereas in case of Demo Application there are five screens available which has all the modules integrated within the screens based on their functionality. And the screens can be switched using the panel button functionality mentioned below in 7.2.2.

8.2.2 Panel Button Functionality

Initially when the device is turned ON, the main interface is displayed which would be the screen1, now to shift from one screen to another screen and to interact with each screen the below keys are available,

Block Name	Function	Key press instructions	Description
Key1	Back	Short Press	This key is used to go back to the previous screen.
Key2	Inc++	Short Press	This key is used to increment the value of a specific module
Key2	SET	Long Press	This key is used to update the changes
Key3	Dec--	Short Press	This key is used to decrement the value of a specific module
Key3	SEL NEXT	Long Press	This key is used to select the next module
Key4	Next SCR	Short Press	This key is used to go to the next Screen

8.2.3 Demo App Screen 1

The below image shows the integrated UI screen from which the user can verify the following software modules.

1) J1939
2) RTC

8.2.3.1 Screen 1 Description:

Screen-1 contains Gauge and RTC. The gauge has a pointer value that will vary based on engine speed and engine speed will be updated based on CAN and J1939.

The RTC has a digital Clock to display the data.

In the above image all the sections of the screen are shown using the arrows, as each section is functioning for different test case such as RTC is for time, J1939 for Engine Speed.

8.2.3.2 Screen 1 Test procedure:

Module Name	Screen Sections	Description	Test Procedure	Results	Range
RTC	Time	This is the real time clock value which will be displayed on the top right corner of screen.	There is no specific test, rather the time will be updated based on the real time data	The current time will be displayed on the Digital clock	-

J1939	Engine Speed	This functionality is used to update the engine Speed based on J1939 which gets updated through CAN	When the system is connected to an external device via CAN, the CAN channel will send the speed details to J1939 and that value will be updated on the UI without any intervention from the user.	The gauge value varies based on the speed of the system.	-

8.2.4 Demo App Screen 2

The below image shows the integrated UI screen from which the user can verify the mentioned software modules.

8.2.4.1 Screen 2 Descriptions:

Screen-2 consists of two modules which are mentioned below:

(1) Digital Output
(2) Ignition Indication

The below illustration provides details of screen 2. The screen has two partitions the first partition contains Digital Output. The other partition contains the Ignition. The digital output comprises of two pin values which is HIGH side and LOW side and one open drive, these pins decide the value of the output. The value 1 indicates the HIGH side and value 0 indicates the LOW side and open drive indicates open. The power monitor comprises the ignition indicator which indicates whether the ignition is ON/OFF. When the indicator is turned ON the power monitor is enabled and when it is turned OFF the power monitor is disabled. When both the high side and Low side pins are OFF (value is 0), the state of the Digital Output is Open drive.

In the above screen, the user can see Output block which has two functionality one is for High Side and the other is for Low Side, in this both we have ON and OFF options. Next is the LCD brightness which has a slider that is used to show the brightness level. Next to LDC brightness is Ignition which has OFF/ON option. On to the other side of the Screen there is Input block available which has four configuration properties and their respective counter values. The user can verify the below functionality on Screen 2 of the sample app.

8.2.4.2 Screen 2 Test Procedure:

Module Name	Screen Sessions	Description	Test Procedure	Expected Result	Units
Digital Output	Output	The digital output comprises of two pin values which is HIGH side, LOW side and Open drive these pins decide the value of the output. The value 1 indicates the HIGH side and value 0 indicates the LOW side. If both the value of HIGH side and LOW side are 0, the digital output is in Open drive	**High Side: -** User can probe the digital output pins by verifying whether the Digital Output high pin should be 1 and Digital Output low Pin should be 0. User can navigate to High side by long pressing Key2 and set it using Key3(long press)	The High Side functionality present in the Output block will be Turned ON	-
			Low Side: - User can probe the digital output pin to by verifying the Digital Output High pin should be 0 and Digital Output low Pin should be 1. User can navigate to Low side by long pressing Key2 and set it using Key3(long press)	The Low Side functionality present in the Output block will be Turned ON	-
			Open Drive: - User can probe the digital output pin to by verifying the Digital Output High pin should be 0 and Digital Output low	The Open drive functionality can be observed by GUI (both high side	

			Pin should be 0. User can navigate to Open Drive (High side and low side OFF) by long pressing Key2 and set it using Key3(long press)	and low side GUI should be OFF) and user can verify by probing the digital io pins state.	
Power Monitor	Ignition Indication	The power monitor comprises the indicator mode which provides two basic functionalities ON/OFF mode. When the indicator is turned ON the power monitor is enabled and when it is turned OFF the power monitor is disabled.	User can navigate the Ignition ON/OFF.	The Ignition will switch modes to ON/OFF based on Power Monitor.	-

8.2.5 Demo App screen 3

Screen-3 consists of only configurable input module.

The configurable input is designed to configure various input channels. It is used to update the latest configurable input values based upon the previously configured channel properties. The different configurable inputs that are currently available are Digital, resistance, voltage and frequency. The blocks in the figure represents different configurable inputs.

8.2.5.1 Screen 3 Test Procedure:

Module Name	Screen Sessions	Description	Test Procedure	Expected Result	Units
Configurable Input	Input	The configurable input is designed to configure various input channels. It is used to update the latest configurable input values based upon the previously configured channel properties. The different configurable inputs that are currently	User can configure each property for a particular channel and then view the data in the adjacent location. **Digital: -** User can configure the Digital property for AI1 to AI5 channel Using Key2(Short press) and	The value of the digital input will be updated on the GUI counter.	0-5

		available are Digital, resistance, current and frequency.	Key3(Short press) and set the specific AI value using Key3		
			Resistance: - User can configure the resistance property for AI1 to AI5 channel Using Key2(Short press) and Key3(Short press) and set the specific AI value using Key3	The value of the Resistance input will be updated on the GUI counter.	Ohm's
			Frequency: - User can configure the frequency property for AI1 to AI5 channel Using Key2(Short press) and Key3(Short press) and set the specific AI value using Key3	The counter value of the frequency input will be updated	Hz
			Voltage: - User can configure the Voltage property for AI1 to AI5 channel Using Key2(Short press) and Key3(Short press) and set the specific AI value using Key3	The value of the Voltage input will be updated on the GUI counter.	V

8.2.6 Demo App Screen 4

The below display is the integrated UI screen from which the user can select the following software modules.

1) SW Timer
2) EEPROM
3) Power mode

8.2.6.1 Screen 4 Descriptions:

Screen 4 is designed using three software module which includes Software timer, EEPROM and Power Mode.

The topmost block which is available on the screen is for Software Timer which has three functionalities:
(1) Increase Timeout
(2) Decrease Timeout
(3) Current timeout

In the later part there are two sections, EEPROM Power mode is available. In the EEPROM block the Timeout counter is read from the EEPROM and displayed and for Power mode the Stop functionality is available.

8.2.6.2 Screen 4 Test procedure:

Module Name	Screen Section	Functionality	Description	Test Procedure	Expected Result
Software Timer	SW Timer	Increase Timeout	The software timer is used to set timeout value for the device software. Where increase timeout is used to increase the time out value	User can increase the timeout value by short pressing of the key-2(Short Press).	Time out value will be increased and displayed under Current timeout
Software Timer	SW Timer	Decrease Timeout	The software timer is used to set timeout value for the device software. Where decrease timeout is used to decrease the timeout value	User can decrease the timeout value by short pressing the key-3.	Time out value will be decreased and displayed under Current timeout
Software Timer	SW Timer	Current Timeout	The software timer is used to set timeout value for the device software. where current timeout is used to show the current timeout value	The current value is auto generated based on the increment or decrement operation performed by the user.	The current timeout value will be updated

EEPROM	EEPROM	Timeout Counter	EEPROM stores the software timer data and the same can be read back.	The timeout counter value will be directly fetched form the EEPROM database which will be dependent on the Software timeout.	The Timeout counter value will be displayed.
Power Mode	Power mode	Stop Mode	This functionality is used to stop all the functions inside the system and it is waiting into the same mode until an interrupt will occur and activate the device.	User can set the power mode as STOP by long pressing of the key-3 the board will switch to the reset mode. User can come back to the wake-up state by using any one of the wake-up sources. They are: • Keypad • RTC • Ignition • CAN	The running functionality will be kept on halt, and the system will be set to Stop mode i.e., system will turn OFF

9 BLE Mobile Test Application

The BLE test application is an android based mobile application which can be used to communicate with the MAXAI280 hardware via BLE for testing/debugging purpose. It supports the below functionalities.

Read/write to all the DB Variables supported by the SDK
Direct Memory location Read.
In this section we will walk you through the BLE App screens and how to use the functionalities of the BLE App.

9.1 Scan Screen

Install the .apk file into your Android Mobile. After installation, open the App. The app home screen is shown as below.
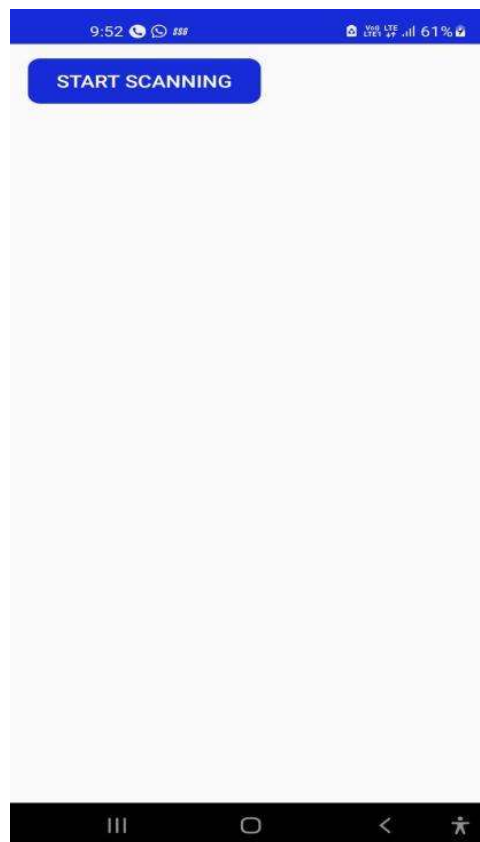
Press the Start Scanning button. The list of Maximatecc AI280 devices available will be displayed on the screen as shown below.

If bluetooth is turned OFF on your mobile phone you will get a notification to turn on Bluetooth and location before using the BLE functionality.

Once the scanning starts you will see the below screen with the list of devices.

9.2   Connect Screen

Press the "**Connect**" button" to connect to the MAXAI280 device. Once the device is connected, "**Connected**" button status will be shown. Press the enter button to go to the next screen. To disconnect the device from BLE communication, the user can press the **"Disconnect"** button.
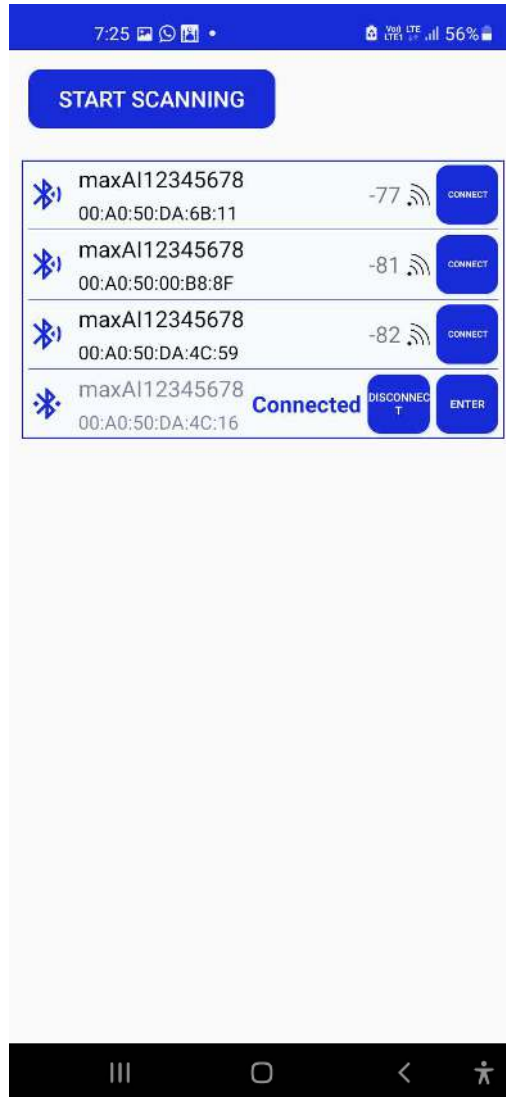
After the connection is successful, the connected device names and services will be shown on the next screen. Press the service UUID on the list and it will move to the next screen.
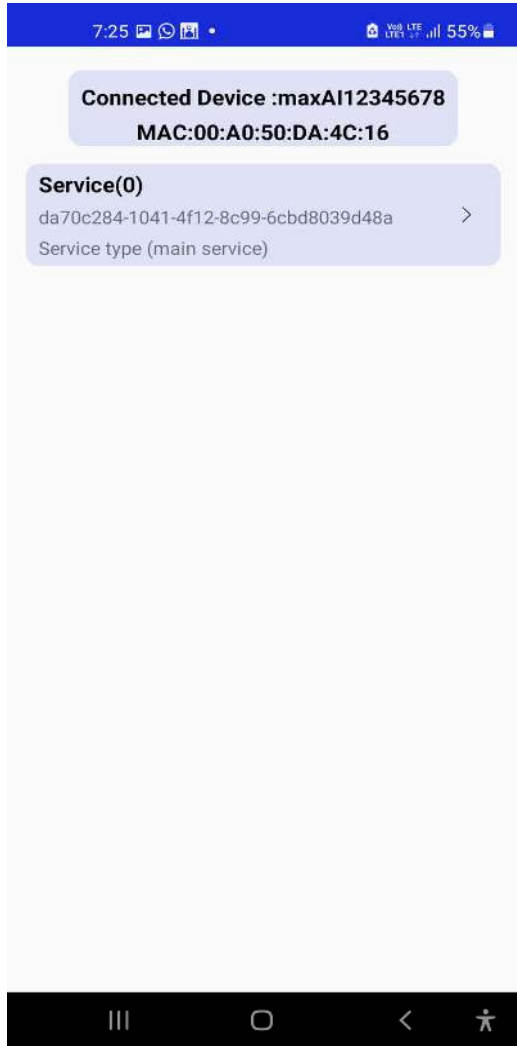
The screen below shows the characteristic UUID on the list. Press the characteristics on the list and it will move to the GUI Screen to communicate with the device.
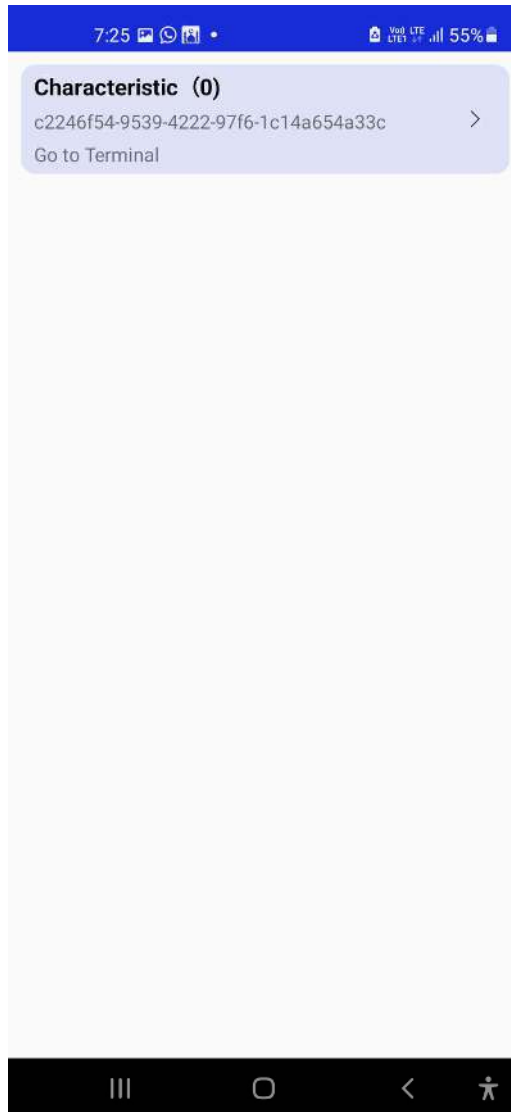
9.3 GUI Screen

The main GUI Screen for the device is shown below.
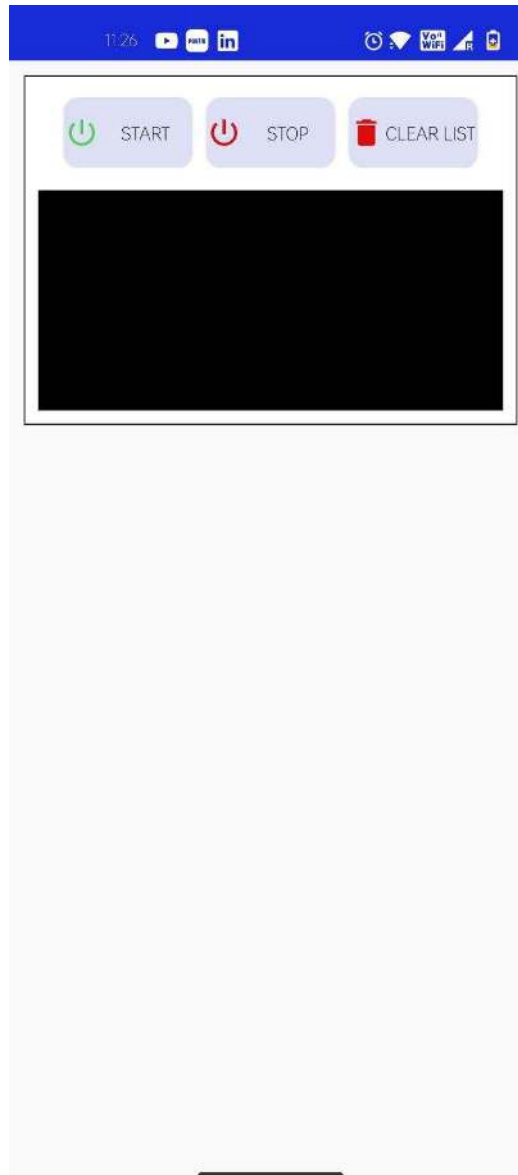
Figure 2

Press the START button to start with the testing. The second half of the screen will be populated with the options once the test has started and the start button is highlighted as shown below.

9.4 Read/Write DB Variable Screen

The screen for Read/Write DB Variable is as below. The screen marked by RED shows the Terminal for RX/TX communications of the DB variables. The screen marked by BLUE shows the Read/Write DB Variable Screen.
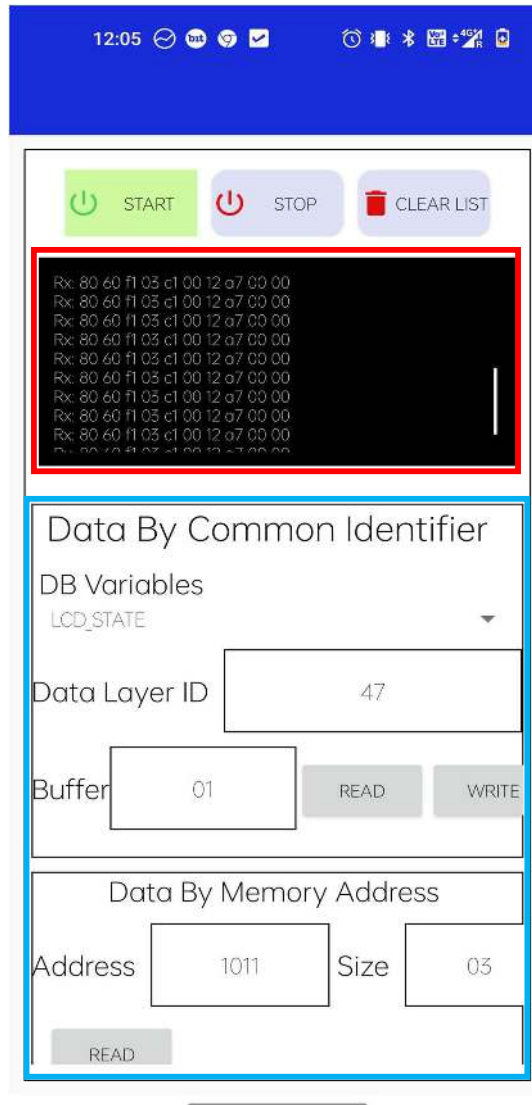
To read the data from the AI280 SDK DB, select the appropriate SDK module for accessing the module's DB variables on the drop-down list. Select the DB variables in the DB drop-down list.

Please click the READ Button for the selected DB variable. The terminal will reflect the communication between the device and the Mobile App. The values present on the device shall also be reflected on the Terminal. To send the updated data to the device, manually type the value

in the buffer text box and then click the WRITE Button. The Terminal will reflect the communication between the device and the Mobile App.
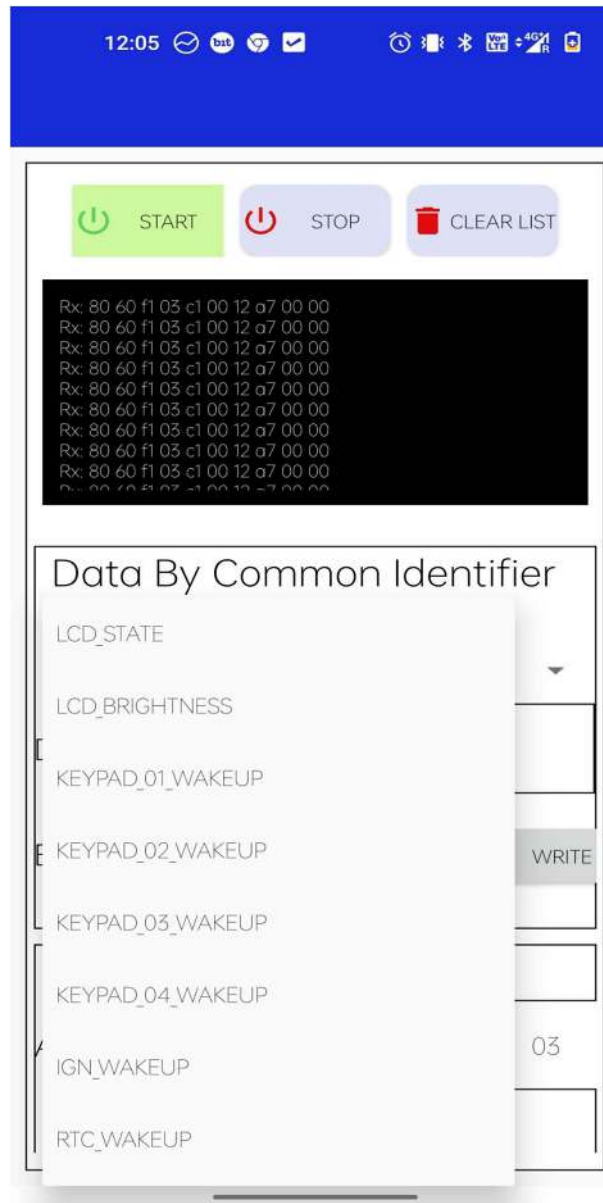
In the LCD module, the LCD state variable is selected. This variable as defined in the section 6.12.4 is used to turn on the LCD.

Once the DB variable is selected, the appropriate DB field ID value will be displayed in the Datalayer ID text box available in the area marked by "Data By Common Identifier".
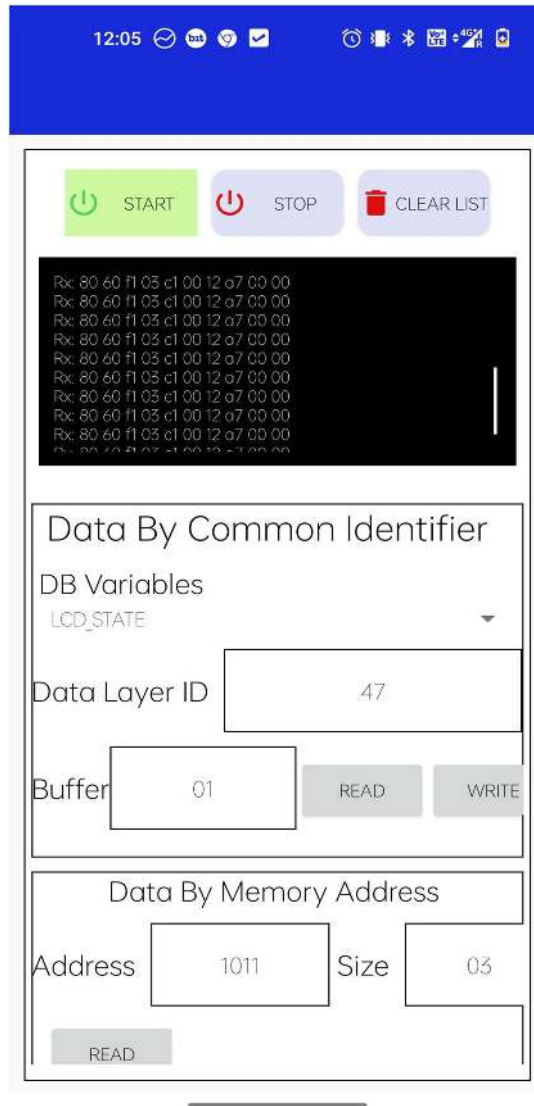
To enable the LCD_STATE in the Buffer text box, hit the write button. The value gets written to the Data base in the AI280 module and the LCD turns ON.

9.5 Read/Write by Memory Address Screen

The screen marked by yellow shows the Memory Access Area. If the user needs to read any memory location, he can directly provide the address in the address field and the size value in the size text field and then click read. The data received from the device would be listed in the TX/RX area. In the above illustration the "Data By Memory address" section has an address and size field.

To read the data of the Memory Location, enter the Address and the Size of the variable under consideration. Click the READ Button for the Selected Memory Address. The terminal will reflect the communication between the device and the Mobile App. The values present on the device shall also be reflected on the Terminal. For example, if we want to read 3 bytes from the address 0x30000000, we will update the address and size as shown below and then click read. The result can be got from the TX/RX area.

The first 4 bytes in the Tx/Rx data carries the header field and the 5th byte in each variable represents the request packet. The next few bytes represent the Data, the last byte denotes the checksum value.

Tx : 80 60 fl 06 23 30 00 00 00 00 03 2d

RX : 80 60 fl 04 63 5f 2a 6b 2c 00

Please see the below screen which shows that the value at memory location 0x30000000 is "5f 2a 6b" as received in our response packet above.
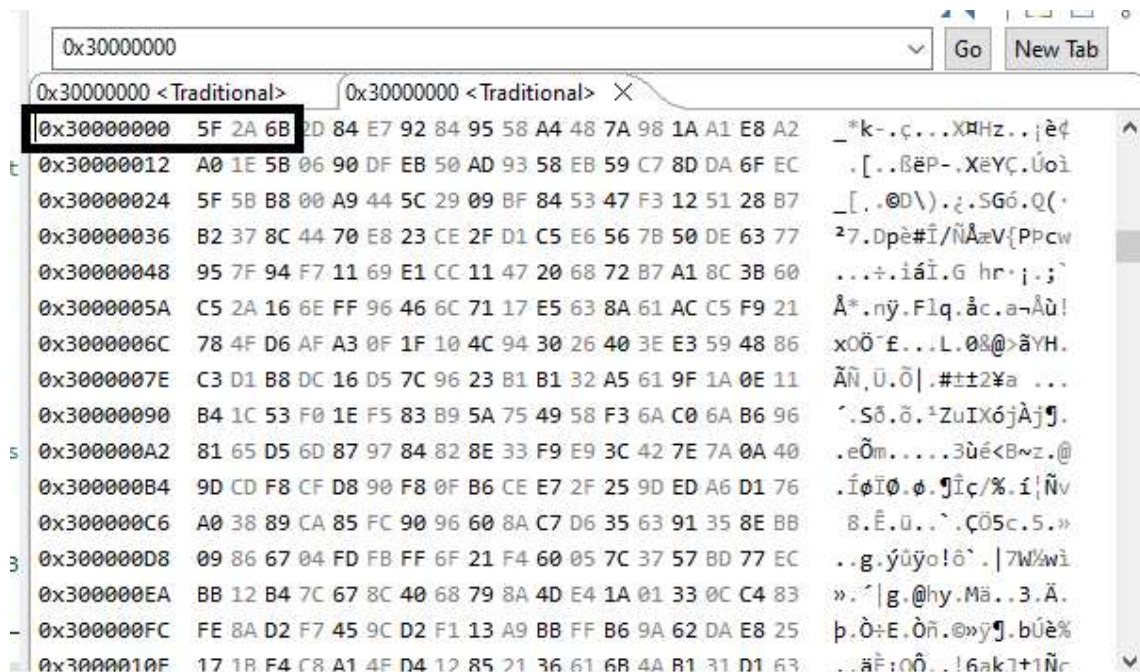
9.6 Generic Data to send

The screen marked by black shows the Generic Data to send section where any generic BLE hex data can be sent to the device.

 For example, we will send a hex data to the BLE device.

TX : 80 60 51 06 50 41 49 34 33 30

The first 4 bytes are header, the next is the request packet followed by the data.

When this data the received by the MAXAI280 board, it is translated to text and displayed in the UI by the Bluetooth App as shown in the below image.

In the below illustration it is understood that the given hexadecimal value is converted into (string value) and the same is been displayed on the device.

9.7 Clear list and Stop testing

Selecting clear list would clear the RX/TX terminal so that the user can see the latest data.

Press the STOP button to stop the testing as shown in the below image.