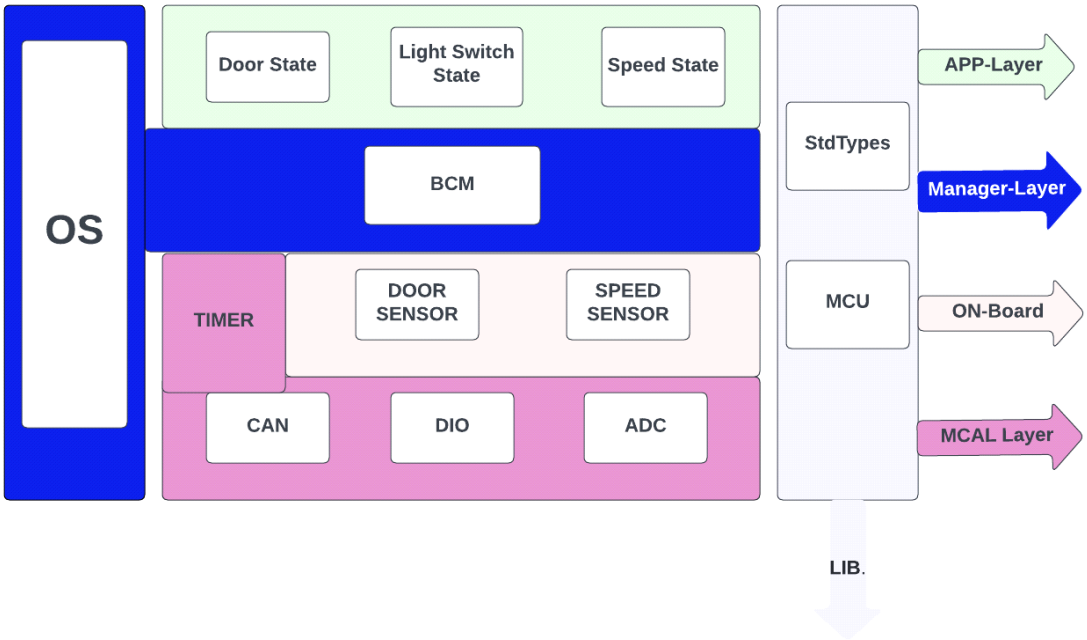
Static Design

1-layered architecture

a)ECU1



2-Provide full detailed APIs for each module as well as a detailed description for the used typedefs

a)TIMER

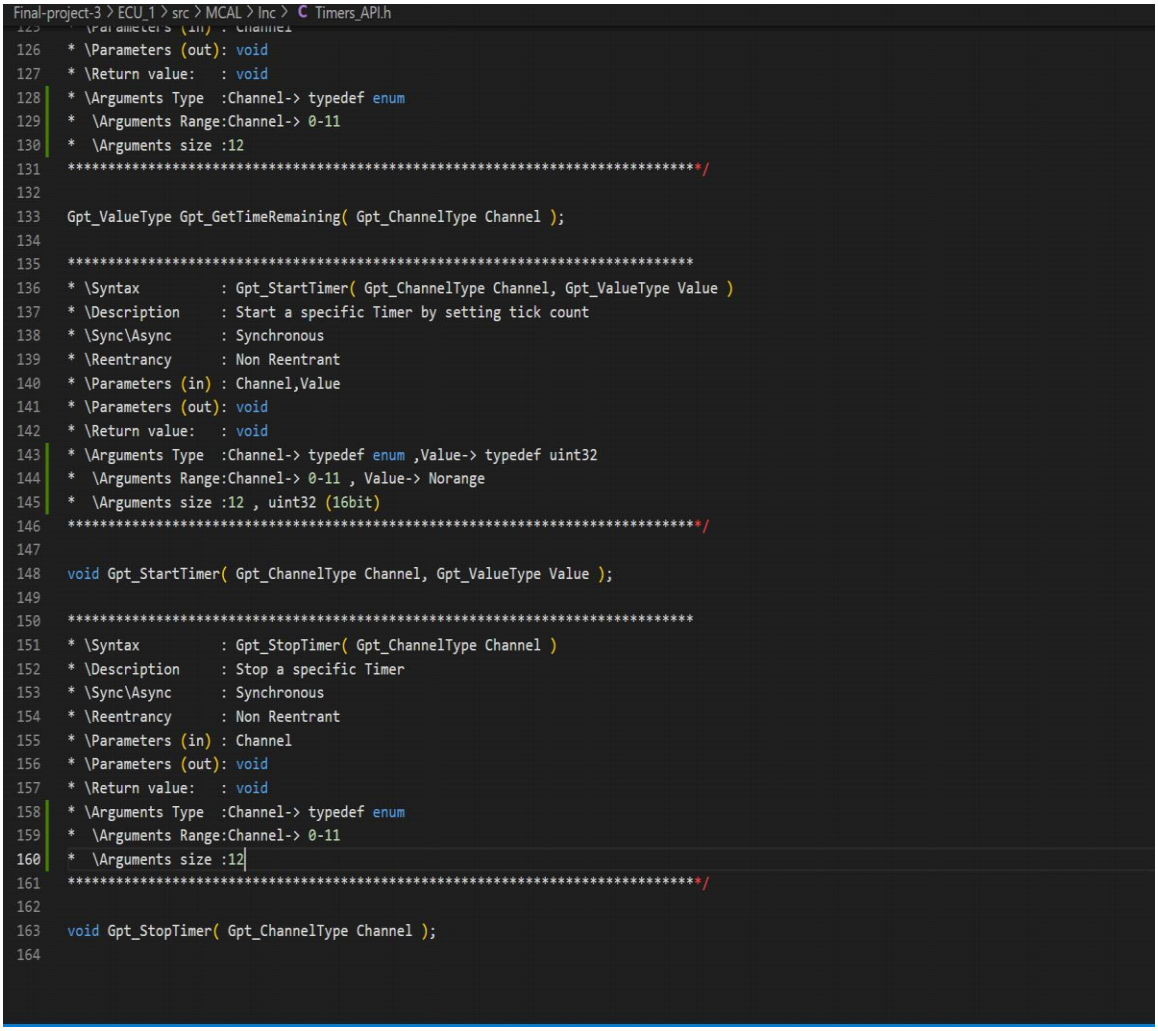
Final-project-3 > ECU_1 > src > MCAL > Inc > C Timers_API.h

```c
typedef enum {

    GPT_16_32_bit_Timer_0         ,
    GPT_16_32_bit_Timer_1         ,
    GPT_16_32_bit_Timer_2         ,
    GPT_16_32_bit_Timer_3         ,
    GPT_16_32_bit_Timer_4         ,
    GPT_16_32_bit_Timer_5         ,
    GPT_32_64_bit_Wide_Timer_0    ,
    GPT_32_64_bit_Wide_Timer_1    ,
    GPT_32_64_bit_Wide_Timer_2    ,
    GPT_32_64_bit_Wide_Timer_3    ,
    GPT_32_64_bit_Wide_Timer_4    ,
    GPT_32_64_bit_Wide_Timer_5

}Gpt_ChannelType;

typedef uint32 Gpt_ValueType;

typedef enum
{
    GPT_MODE_NORMAL,
    GPT_MODE_SLEEP

}Gpt_ModeType;

typedef enum
{
    GPT_PREDEF_TIMER_1US_16BIT,
    GPT_PREDEF_TIMER_1US_24BIT,
    GPT_PREDEF_TIMER_1US_32BIT,
    GPT_PREDEF_TIMER_100US_32BIT
}Gpt_PredefTimerType;

typedef uint32 Gpt_ChannelTickFrequency;
typedef uint32 GptChannelTickValueMax;

typedef enum
{
    GPT_CH_MODE_PERIODIC,
    GPT_CH_MODE_ONESHOT
}ChannelMode;
```

```c
typedef void(*GptNotification)(void);

typedef struct
{
    Gpt_ChannelType                    channel;
    Gpt_ValueType                 channelTickFreq;
    GptChannelTickValueMax      channelTickMaxValue;
    ChannelMode                          channelMode;
    GptNotification                   gptNotification;
}Gpt_ConfigType;

extern const Gpt_ConfigType Gpt_Config[];

extern DIO_LevelType Timer_Flag ;


/*******************************************************************************
* \Syntax          : Gpt_Init( const Gpt_ConfigType* ConfigPtr)
* \Description     : Intialization of Timer
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  :None
*  \Arguments Range:None
*  \Arguments size :None
*******************************************************************************/

void Gpt_Init( const Gpt_ConfigType* ConfigPtr);

/*******************************************************************************
* \Syntax          : Gpt_DisableNotification( Gpt_ChannelType Channel )
* \Description     : Disable Interupt for a specific timer
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Channel
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  :Channel-> typedef enum
*  \Arguments Range:Channel-> 0-11
*  \Arguments size :12
```

Final-project-3 > ECU_1 > src > MCAL > Inc > C Timers_API.h

```
86  ********************************************************************************/
87
88  void Gpt_DisableNotification( Gpt_ChannelType Channel );
89
90  ********************************************************************************
91  * \Syntax          : Gpt_EnableNotification( Gpt_ChannelType Channel )
92  * \Description     : Enable Interupt for a specific timer
93  * \Sync\Async      : Synchronous
94  * \Reentrancy      : Non Reentrant
95  * \Parameters (in) : Channel
96  * \Parameters (out): void
97  * \Return value:   : void
98  * \Arguments Type  :Channel-> typedef enum
99  *  \Arguments Range:Channel-> 0-11
100 *  \Arguments size :12
101 ********************************************************************************/
102
103 void Gpt_EnableNotification( Gpt_ChannelType Channel );
104
105 ********************************************************************************
106 * \Syntax          : Gpt_GetTimeElapsed( Gpt_ChannelType Channel )
107 * \Description     : GetTimeElapsed of specific timer
108 * \Sync\Async      : Synchronous
109 * \Reentrancy      : Non Reentrant
110 * \Parameters (in) : Channel
111 * \Parameters (out): void
112 * \Return value:   : void
113 * \Arguments Type  :Channel-> typedef enum
114 *  \Arguments Range:Channel-> 0-11
115 *  \Arguments size :12
116 ********************************************************************************/
117
118 Gpt_ValueType Gpt_GetTimeElapsed( Gpt_ChannelType Channel );
119
120 ********************************************************************************
121 * \Syntax          : Gpt_GetTimeRemaining( Gpt_ChannelType Channel )
122 * \Description     : GetTimeRemaining of specific timer
123 * \Sync\Async      : Synchronous
124 * \Reentrancy      : Non Reentrant
125 * \Parameters (in) : Channel
126 * \Parameters (out): void
127 * \Return value:   : void
```


Final-project-3 > ECU_1 > src > MCAL > Inc > C Timers_API.h
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :Channel-> typedef enum
* \Arguments Range:Channel-> 0-11
* \Arguments size :12
Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType Channel);
* \Syntax : Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value)
* \Description : Start a specific Timer by setting tick count
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : Channel,Value
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :Channel-> typedef enum ,Value-> typedef uint32
* \Arguments Range:Channel-> 0-11 , Value-> Norange
* \Arguments size :12 , uint32 (16bit)
void Gpt_StartTimer(Gpt_ChannelType Channel, Gpt_ValueType Value);
* \Syntax : Gpt_StopTimer(Gpt_ChannelType Channel)
* \Description : Stop a specific Timer
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : Channel
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :Channel-> typedef enum
* \Arguments Range:Channel-> 0-11
* \Arguments size :12
void Gpt_StopTimer(Gpt_ChannelType Channel);

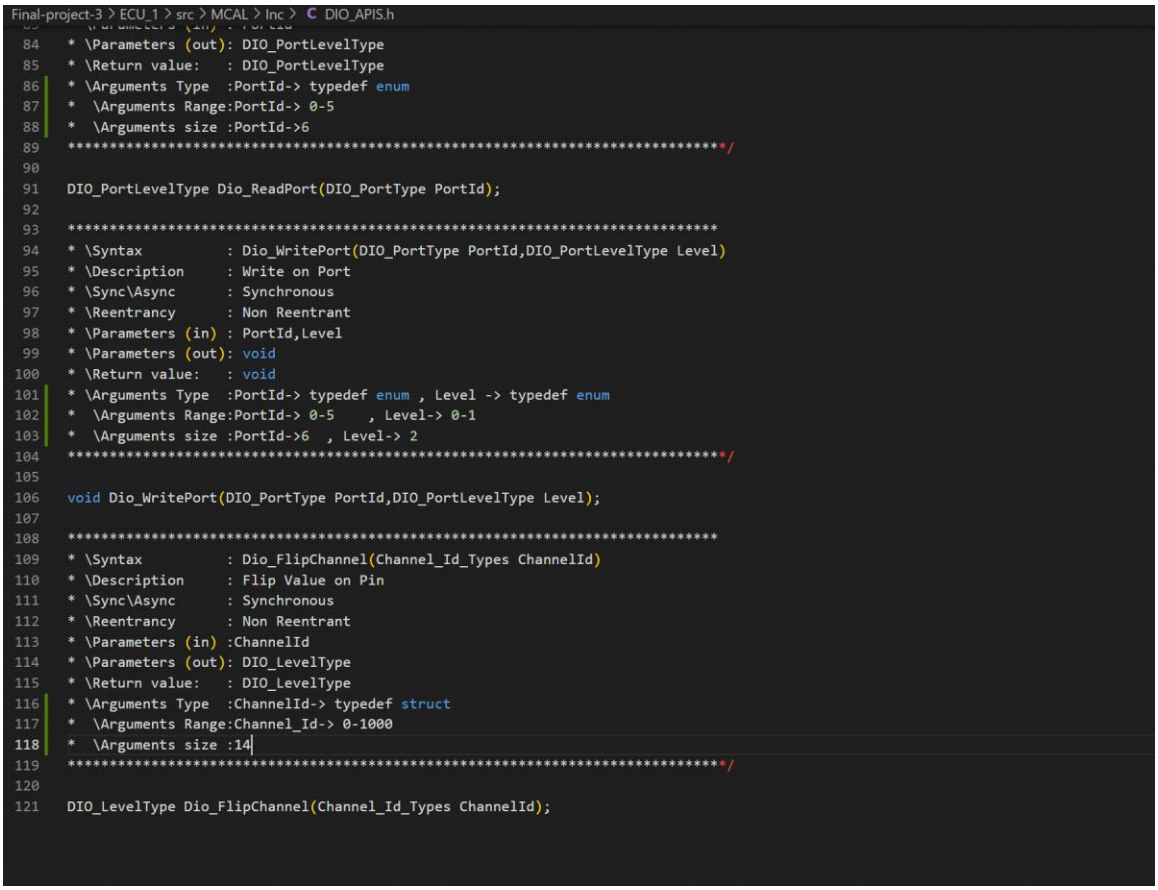
b)DIO

Final-project-3 > ECU_1 > src > MCAL > Inc > C DIO_APIS.h

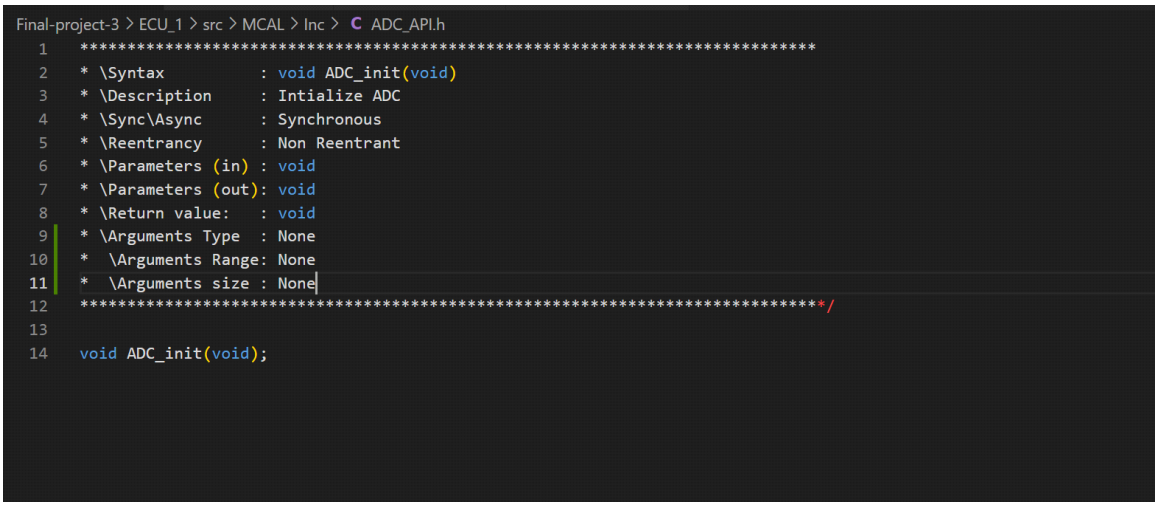
```c

typedef enum {

 PIN0                                    ,
 PIN1                                    ,
 PIN2                                    ,
 PIN3                                    ,
 PIN4                                    ,
 PIN5                                    ,
 PIN6                                    ,
 PIN7

}DIO_ChannelType;

typedef enum {

 PortA                                   ,
 PortB                                   ,
 PortC                                   ,
 PortD                                   ,
 PortE                                   ,
 PortF


}DIO_PortType;

typedef enum {

 LOW                                      ,
 HIGH

}DIO_LevelType;


typedef struct  {


 DIO_PortType  Port_num                  ;
 DIO_ChannelType Chann_n                 ;

}Channel_Id_Types;

```

Final-project-3 > ECU_1 > src > MCAL > Inc > C DIO_APIS.h

```c

typedef uint32  DIO_PortLevelType ;



/*******************************************************************************
* \Syntax          : DIO_LevelType Dio_ReadChannel(Channel_Id_Types ChannelId)
* \Description     : Read PIN by its Pin number and return its value
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ChannelId
* \Parameters (out): DIO_LevelType
* \Return value:   : DIO_LevelType
* \Arguments Type  :ChannelId-> typedef struct
*  \Arguments Range:Channel_Id-> 0-1000
*  \Arguments size :14
*******************************************************************************/

DIO_LevelType Dio_ReadChannel(Channel_Id_Types ChannelId);

/*******************************************************************************
* \Syntax          : void Dio_WriteChannel(Channel_Id_Types ChannelId,DIO_LevelType Level)
* \Description     : Write on PIN High or Low
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ChannelId,Level
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  :ChannelId-> typedef struct ,Level -> typedef enum
*  \Arguments Range:Channel_Id-> 0-1000  , Level-> 0-1
*  \Arguments size :Channel_Id->14 ,Level->2
*******************************************************************************/

void Dio_WriteChannel(Channel_Id_Types ChannelId,DIO_LevelType Level);

/*******************************************************************************
* \Syntax          : DIO_PortLevelType Dio_ReadPort(DIO_PortType PortId)
* \Description     : Read Port and return Port
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : PortId
* \Parameters (out): DIO_PortLevelType
```


Final-project-3 > ECU_1 > src > MCAL > Inc > C DIO_APIS.h
* \Parameters (out): DIO_PortLevelType
* \Return value: : DIO_PortLevelType
* \Arguments Type :PortId-> typedef enum
* \Arguments Range:PortId-> 0-5
* \Arguments size :PortId->6
DIO_PortLevelType Dio_ReadPort(DIO_PortType PortId);
* \Syntax : Dio_WritePort(DIO_PortType PortId,DIO_PortLevelType Level)
* \Description : Write on Port
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : PortId,Level
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :PortId-> typedef enum , Level -> typedef enum
* \Arguments Range:PortId-> 0-5 , Level-> 0-1
* \Arguments size :PortId->6 , Level-> 2
void Dio_WritePort(DIO_PortType PortId,DIO_PortLevelType Level);
* \Syntax : Dio_FlipChannel(Channel_Id_Types ChannelId)
* \Description : Flip Value on Pin
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) :ChannelId
* \Parameters (out): DIO_LevelType
* \Return value: : DIO_LevelType
* \Arguments Type :ChannelId-> typedef struct
* \Arguments Range:Channel_Id-> 0-1000
* \Arguments size :14
DIO_LevelType Dio_FlipChannel(Channel_Id_Types ChannelId);

c)ADC


Final-project-3 > ECU_1 > src > MCAL > Inc > C ADC_API.h
* \Syntax : void ADC_init(void)
* \Description : Intialize ADC
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value: : void
* \Arguments Type : None
* \Arguments Range: None
* \Arguments size : None
void ADC_init(void);

d)CAN

```
1
2   ********************************************************************************
3   * \Syntax          : CAN_Initialize(uint32 Channel_Id ,uint32 speed, uint32 linkingPort,uint32 interrupt)
4   * \Description     : Initialisation of the channel, setting the speed, linking port and interrupt for non PnP devices
5   * \Sync\Async      : Synchronous
6   * \Reentrancy      : Non Reentrant
7   * \Parameters (in) : Channel_Id,speed,linkingPort,interrupt
8   * \Parameters (out): void
9   * \Return value:   : void
10  * \Arguments Type  : uint32,uint32,uint32,uint32
11  *  \Arguments Range:Channel_Id-> 0-1000  , speed-> 0-200 , linkingPort ->0-100 , interrupt -> 0-100
12  *  \Arguments size :uint32,uint32,uint32,uint32
13  ********************************************************************************/
14
15  void CAN_Initialize(uint32 Channel_Id ,uint32 speed, uint32 linkingPort,uint32 interrupt);
16
17  ********************************************************************************
18  * \Syntax          : CAN_Write(uint32 Channel_Id,uint32 CAN_data)
19  * \Description     : Sending a CAN message
20  * \Sync\Async      : Synchronous
21  * \Reentrancy      : Non Reentrant
22  * \Parameters (in) : Channel_Id,CAN_data
23  * \Parameters (out): void
24  * \Return value:   : void
25  * \Arguments Type  : uint32,uint32
26  *  \Arguments Range:Channel_Id-> 0-1000  , CAN_data-> 0-30
27  *  \Arguments size :uint32,uint32
28  ********************************************************************************/
29
30  void CAN_Write(uint32 Channel_Id,uint32 CAN_data);
```

```
void CAN_Write(uint32 Channel_Id,uint32 CAN_data);

********************************************************************************
* \Syntax          : CAN_Read(uint32 Channel_Id)
* \Description     : Reading a CAN message
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Channel_Id
* \Parameters (out): uint32
* \Return value:   : uint32
* \Arguments Type  : uint32
*  \Arguments Range:Channel_Id-> 0-1000
*  \Arguments size :uint32
********************************************************************************/

uint32 CAN_Write(uint32 Channel_Id);
```

e) DOOR SENSOR

Final-project-3 > ECU_1 > src > OnBoard > Inc > C DoorSensor_API.h

```h


typedef enum {

  DoorSensor_0,
  DoorSensor_1,
  DoorSensor_2,

}DoorSensor_Type;


typedef uint32  Sensor_Read ;



/*******************************************************************************
* \Syntax          : DoorSensor_Init(void)
* \Description     : DoorSensor_Initilization
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : None
*  \Arguments Range: None
*  \Arguments size : None
*******************************************************************************/

void DoorSensor_Init(void);


/*******************************************************************************
* \Syntax          : DoorSensor_ReadStatus( DoorSensor_Type Sensor_Num)
* \Description     : Read the sensor readings every 10 ms
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): Sensor_Read
* \Return value:   : Sensor_Read
* \Arguments Type  : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*  \Arguments Range: Sensor_Num->0-2 , Sensor_Read-> no range (Integer uint32)
*  \Arguments size : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
```

Ln 42, Col 76 (232 sele

```h
/*******************************************************************************
* \Syntax          : DoorSensor_ReadStatus( DoorSensor_Type Sensor_Num)
* \Description     : Read the sensor readings every 10 ms
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): Sensor_Read
* \Return value:   : Sensor_Read
* \Arguments Type  : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*  \Arguments Range: Sensor_Num->0-2 , Sensor_Read-> no range (Integer uint32)
*  \Arguments size : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*******************************************************************************/

Sensor_Read DoorSensor_ReadStatus( DoorSensor_Type Sensor_Num);
```

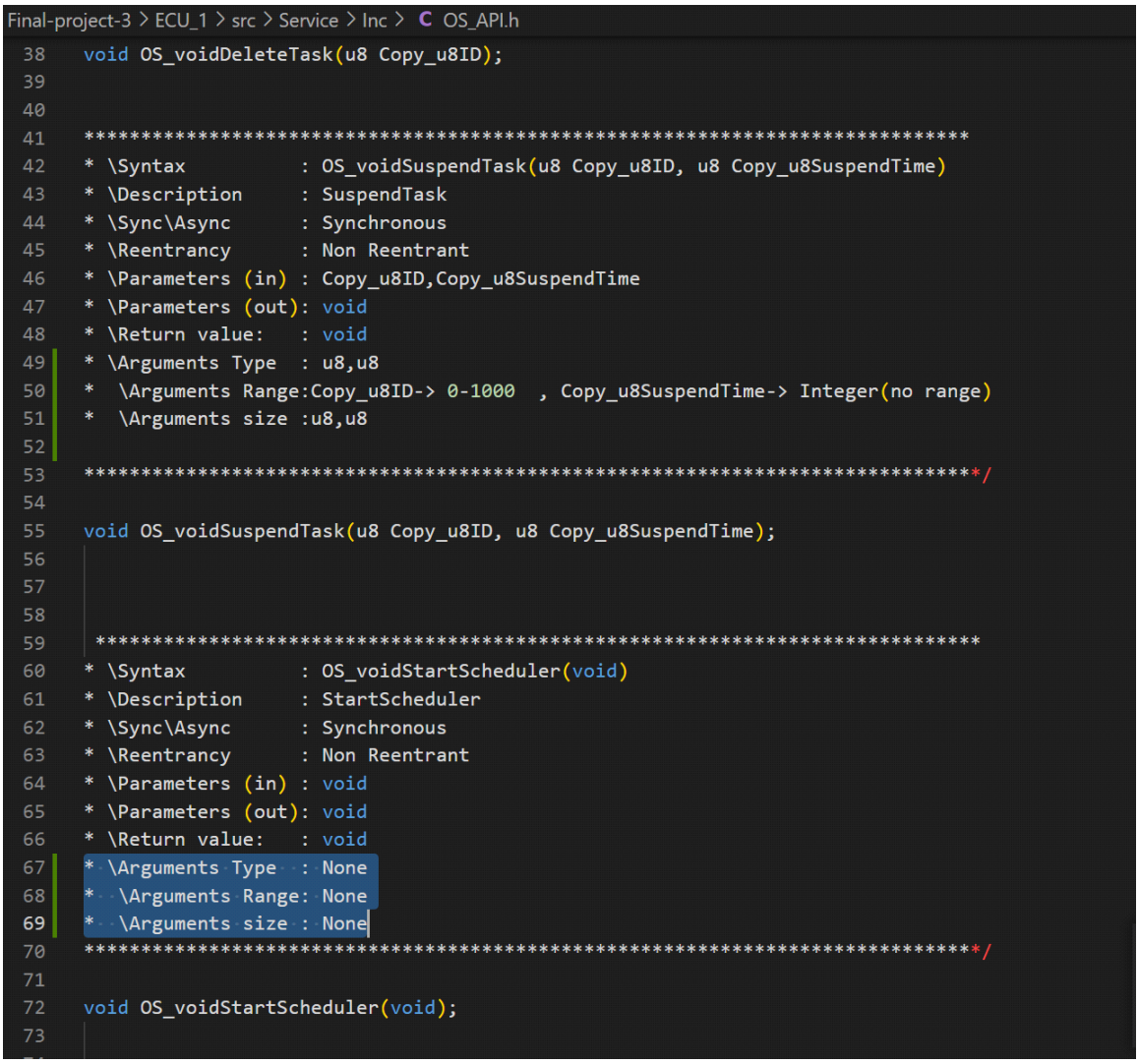
f)SPEED SENSOR

Final-project-3 > ECU_1 > src > OnBoard > Inc > C SpeedSensor.h

```c
typedef enum {

  SpeedSensor_0,
  SpeedSensor_1,
  SpeedSensor_2,

}SpeedSensor_Type;


typedef uint32  Sensor_Read ;




/*******************************************************************************
* \Syntax          : SpeedSensor_Init(void)
* \Description     : SpeedSensor Initilization
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : None
*  \Arguments Range: None
*  \Arguments size : None
*******************************************************************************/

void SpeedSensor_Init(void);

/*******************************************************************************
* \Syntax          : SpeedSensor_ReadStatus( DoorSensor_Type Sensor_Num)
* \Description     : Read the sensor readings every 5 ms
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): Sensor_Read
* \Return value:   : Sensor_Read
* \Arguments Type  : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*  \Arguments Range: Sensor_Num->0-2 , Sensor_Read-> no range (Integer uint32)
*  \Arguments size : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*******************************************************************************/

```

```c
/*******************************************************************************
* \Syntax          : SpeedSensor_ReadStatus( DoorSensor_Type Sensor_Num)
* \Description     : Read the sensor readings every 5 ms
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): Sensor_Read
* \Return value:   : Sensor_Read
* \Arguments Type  : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*  \Arguments Range: Sensor_Num->0-2 , Sensor_Read-> no range (Integer uint32)
*  \Arguments size : Sensor_Num->typedef enum , Sensor_Read->typedef uint32
*******************************************************************************/

Sensor_Read SpeedSensor_ReadStatus( DoorSensor_Type Sensor_Num);
```

g)OS


Final-project-3 > ECU_1 > src > Service > Inc > C OS_API.h
38 void OS_voidDeleteTask(u8 Copy_u8ID);
39
40
41 **
42 * \Syntax : OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime)
43 * \Description : SuspendTask
44 * \Sync\Async : Synchronous
45 * \Reentrancy : Non Reentrant
46 * \Parameters (in) : Copy_u8ID,Copy_u8SuspendTime
47 * \Parameters (out): void
48 * \Return value: : void
49 * \Arguments Type : u8,u8
50 * \Arguments Range:Copy_u8ID-> 0-1000 , Copy_u8SuspendTime-> Integer(no range)
51 * \Arguments size :u8,u8
52
53 **/
54
55 void OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime);
56
57
58
59 **
60 * \Syntax : OS_voidStartScheduler(void)
61 * \Description : StartScheduler
62 * \Sync\Async : Synchronous
63 * \Reentrancy : Non Reentrant
64 * \Parameters (in) : void
65 * \Parameters (out): void
66 * \Return value: : void
67 * \Arguments Type : None
68 * \Arguments Range: None
69 * \Arguments size : None
70 **/
71
72 void OS_voidStartScheduler(void);
73

```
41  ********************************************************************************
42  * \Syntax          : OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime)
43  * \Description     : SuspendTask
44  * \Sync\Async      : Synchronous
45  * \Reentrancy      : Non Reentrant
46  * \Parameters (in) : Copy_u8ID,Copy_u8SuspendTime
47  * \Parameters (out): void
48  * \Return value:   : void
49  * \Arguments Type  : u8,u8
50  *  \Arguments Range:Copy_u8ID-> 0-1000  , Copy_u8SuspendTime-> Integer(no range)
51  *  \Arguments size :u8,u8
52
53  ********************************************************************************/
54
55  void OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime);
56
57
58
59   ********************************************************************************
60  * \Syntax          : OS_voidStartScheduler(void)
61  * \Description     : StartScheduler
62  * \Sync\Async      : Synchronous
63  * \Reentrancy      : Non Reentrant
64  * \Parameters (in) : void
65  * \Parameters (out): void
66  * \Return value:   : void
67  * \Arguments Type  : None
68  *  \Arguments Range: None
69  *  \Arguments size : None
70  ********************************************************************************/
71
72  void OS_voidStartScheduler(void);
73
74
```

Final-project-3 > ECU_1 > src > Service > Inc > C OS_API.h

```c

 ********************************************************************************
* \Syntax          : OS_voidResumeTask(u8 Copy_u8ID)
* \Description     : ResumeTask
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Copy_u8ID
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : u8
*  \Arguments Range:Copy_u8ID-> 0-1000
*  \Arguments size :u8
********************************************************************************/

void OS_voidResumeTask(u8 Copy_u8ID);


 ********************************************************************************
* \Syntax          : OS_u8GetTaskState(u8 Copy_u8ID)
* \Description     : GetTaskState
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Copy_u8ID
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : u8
*  \Arguments Range:Copy_u8ID-> 0-1000
*  \Arguments size :u8
********************************************************************************/

u8 OS_u8GetTaskState(u8 Copy_u8ID);

```

h)BCM

```c
/*******************************************************************************
* \Syntax          : BCM_init(void)
* \Description     : Intialize BCM
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : None
*  \Arguments Range: None
*  \Arguments size : None
*******************************************************************************/

void BCM_init(void);
```

I)DOOR STATE

Final-project-3 > ECU_1 > src > APP > Inc > C DoorState_API.h

```c
/*******************************************************************************
* \Syntax          : DoorSensor_SentState_10ms( DoorSensor_Type Sensor_Num)
* \Description     : Door state message will be sent every 10 ms to ECU 2
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : Sensor_Num->typedef enum
* \Arguments Range: Sensor_Num->0-2
* \Arguments size : Sensor_Num->typedef enum
*******************************************************************************/

void DoorSensor_SendState_10ms( DoorSensor_Type Sensor_Num);
```

J) SPEED SENSOR STATE

Final-project-3 > ECU_1 > src > APP > Inc > C SpeedState_API.h

```c


/*******************************************************************************
* \Syntax          : SpeedSensor_SendState_10ms( DoorSensor_Type Sensor_Num)
* \Description     : Speed state message will be sent every 5 ms to ECU 2
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  : Sensor_Num->typedef enum
*  \Arguments Range: Sensor_Num->0-2
*  \Arguments size : Sensor_Num->typedef enum
*******************************************************************************/

void SpeedSensor_SendState_10ms( DoorSensor_Type Sensor_Num);
```

K)LIGHTSWITCH STATE

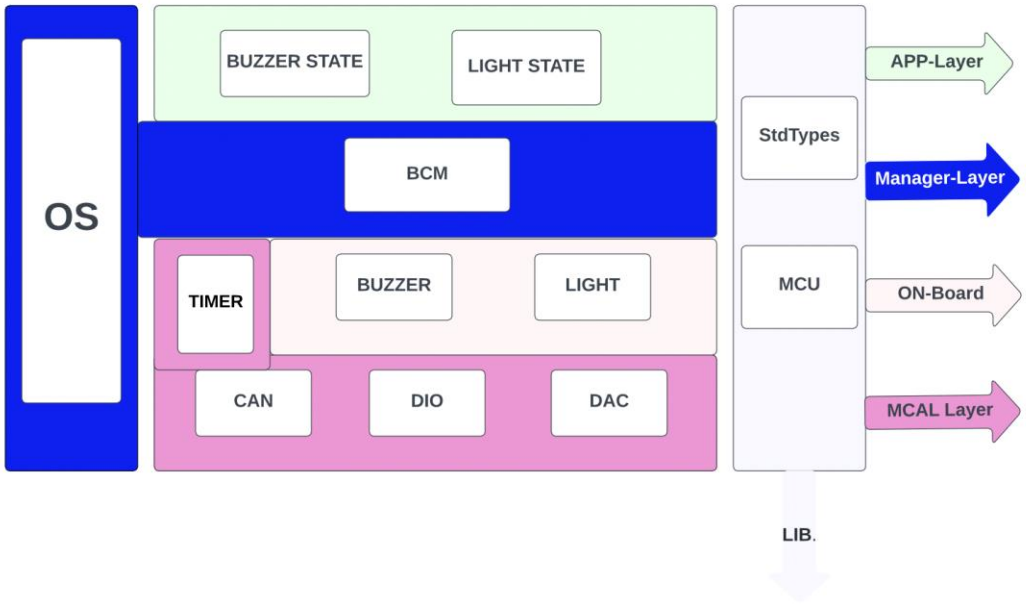
Final-project-3 > ECU_1 > src > APP > Inc > C Switch_API.h

```c


/*******************************************************************************
* \Syntax          : Switch_init(Channel_Id_Types Switch_Id);
* \Description     : Intialize Switch with its port and pin number
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Switch_Id
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  :Switch_Id-> typedef struct
*  \Arguments Range:Switch_Id-> 0-1000
*  \Arguments size :14

*******************************************************************************/

void Switch_init(Channel_Id_Types Switch_Id);

/*******************************************************************************
* \Syntax          : Switch_SendState_20ms( DoorSensor_Type Sensor_Num)
* \Description     : Light switch state message will be sent every 20 ms to ECU 2
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : ConfigPtr
* \Parameters (out): void
* \Return value:   : void
* \Arguments Type  :Switch_Id-> typedef struct
*  \Arguments Range:Switch_Id-> 0-1000
*  \Arguments size :14
*******************************************************************************/

void Switch_SendState_20ms(Channel_Id_Types Switch_Id );
```

L)STD_TYPES

```c
typedef unsigned char uint8;
typedef unsigned short int uint16;
typedef unsigned long int uint32;

typedef signed char suint8;
typedef signed short int suint16;
typedef signed long int suint32;
```

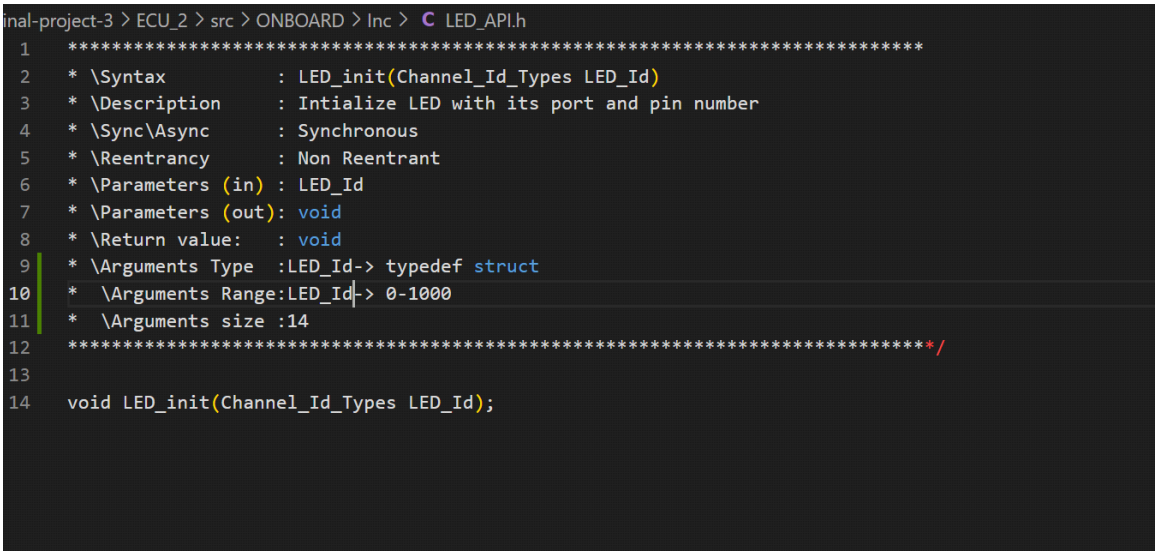
b)ECU2



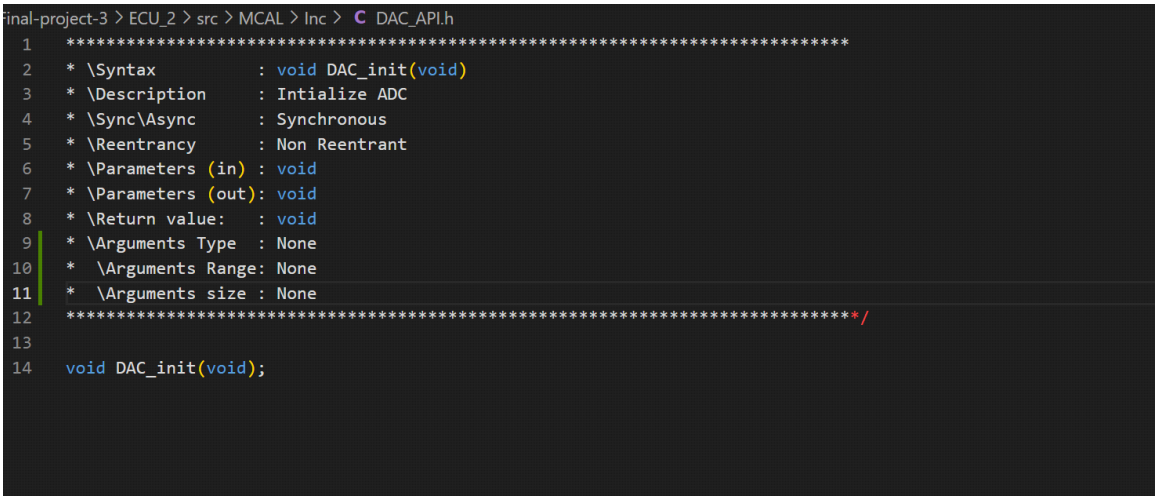
A)BUZZER


Final-project-3 > ECU_2 > src > ONBOARD > Inc > C Buzzer.h
* \Syntax : SBuzzer_init(Channel_Id_Types Buzzer_Id)
* \Description : Intialize BUZZER with its port and pin number
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : Buzzer_Id
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :Buzzer_Id-> typedef struct
* \Arguments Range:Buzzer_Id-> 0-1000
* \Arguments size :14
void Buzzer_init(Channel_Id_Types Buzzer_Id);

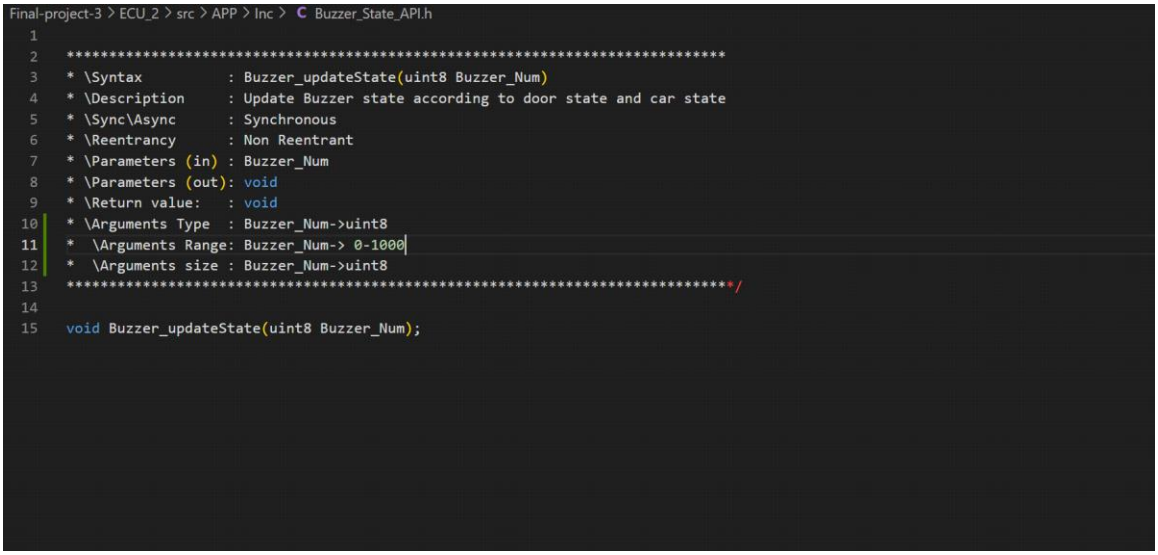
B)LED


inal-project-3 > ECU_2 > src > ONBOARD > Inc > C LED_API.h
* \Syntax : LED_init(Channel_Id_Types LED_Id)
* \Description : Intialize LED with its port and pin number
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : LED_Id
* \Parameters (out): void
* \Return value: : void
* \Arguments Type :LED_Id-> typedef struct
* \Arguments Range:LED_Id-> 0-1000
* \Arguments size :14
void LED_init(Channel_Id_Types LED_Id);

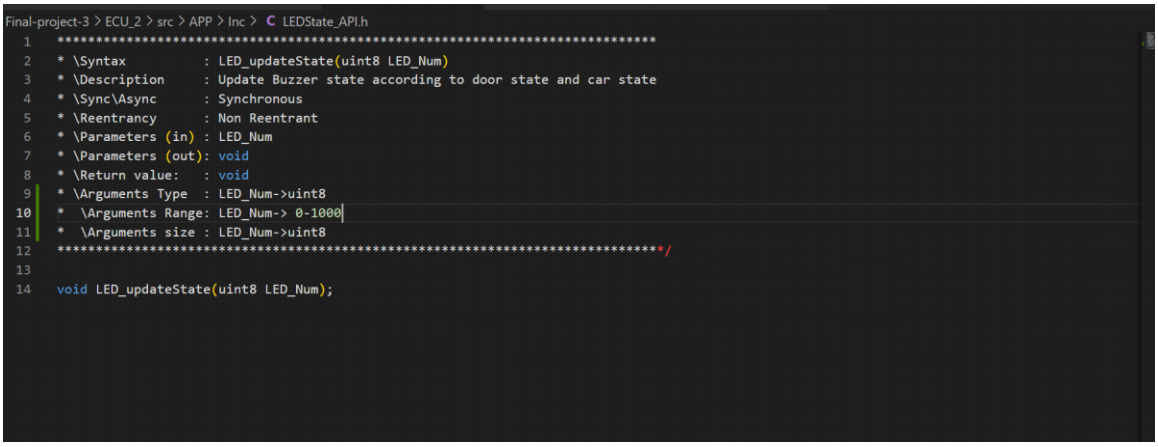
C)DAC



D)BUZZER STATE



E)LED STATE


Final-project-3 > ECU_2 > src > APP > Inc > C LEDState_API.h
/***
* \Syntax : LED_updateState(uint8 LED_Num)
* \Description : Update Buzzer state according to door state and car state
* \Sync\Async : Synchronous
* \Reentrancy : Non Reentrant
* \Parameters (in) : LED_Num
* \Parameters (out): void
* \Return value: : void
* \Arguments Type : LED_Num->uint8
* \Arguments Range: LED_Num-> 0-1000
* \Arguments size : LED_Num->uint8
***/
void LED_updateState(uint8 LED_Num);