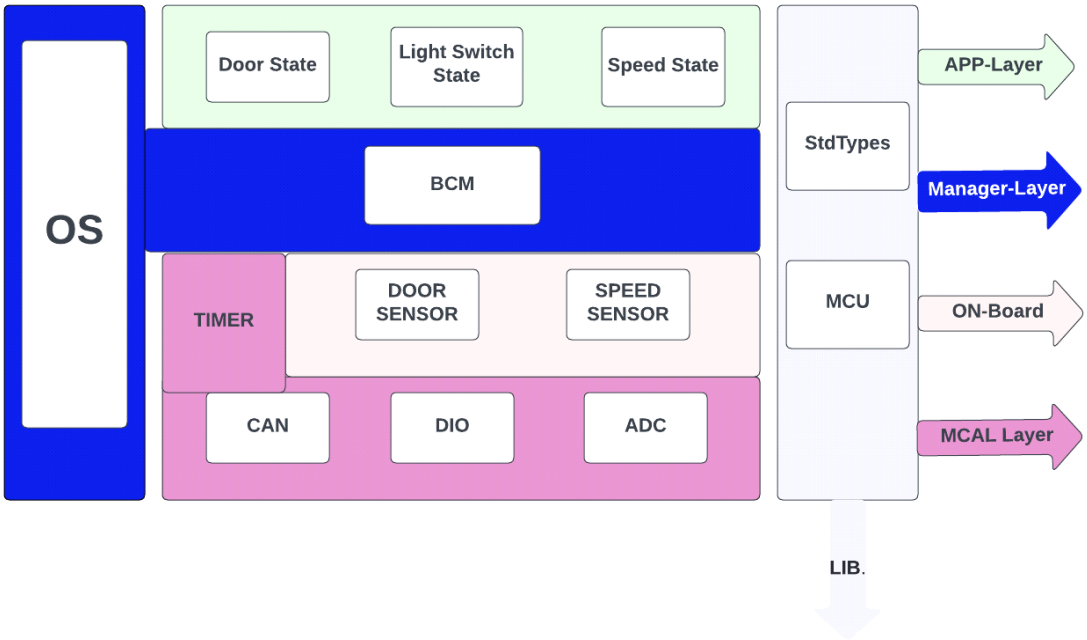
Static Design

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

```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
*************************************************************************/

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

void Gpt_DisableNotification( Gpt_ChannelType Channel );

/*************************************************************************
```

```
84  ********************************************************************************
85  * \Syntax          : Gpt_EnableNotification( Gpt_ChannelType Channel )
86  * \Description     : Enable Interupt for a specific timer
87  * \Sync\Async      : Synchronous
88  * \Reentrancy      : Non Reentrant
89  * \Parameters (in) : Channel
90  * \Parameters (out): void
91  * \Return value:   : void
92  ********************************************************************************/
93
94  void Gpt_EnableNotification( Gpt_ChannelType Channel );
95
96  ********************************************************************************
97  * \Syntax          : Gpt_GetTimeElapsed( Gpt_ChannelType Channel )
98  * \Description     : GetTimeElapsed of specific timer
99  * \Sync\Async      : Synchronous
100 * \Reentrancy      : Non Reentrant
101 * \Parameters (in) : Channel
102 * \Parameters (out): void
103 * \Return value:   : void
104 ********************************************************************************/
105
106 Gpt_ValueType Gpt_GetTimeElapsed( Gpt_ChannelType Channel );
107
108 ********************************************************************************
109 * \Syntax          : Gpt_GetTimeRemaining( Gpt_ChannelType Channel )
110 * \Description     : GetTimeRemaining of specific timer
111 * \Sync\Async      : Synchronous
112 * \Reentrancy      : Non Reentrant
113 * \Parameters (in) : Channel
114 * \Parameters (out): void
115 * \Return value:   : void
116 ********************************************************************************/
117
118 Gpt_ValueType Gpt_GetTimeRemaining( Gpt_ChannelType Channel );
119
120 ********************************************************************************
121 * \Syntax          : Gpt_StartTimer( Gpt_ChannelType Channel, Gpt_ValueType Value )
122 * \Description     : Start a specific Timer by setting tick count
123 * \Sync\Async      : Synchronous
124 * \Reentrancy      : Non Reentrant
125 * \Parameters (in) : Channel,Value
```

```
110   * \Description     : GetTimeRemaining of specific timer
111   * \Sync\Async      : Synchronous
112   * \Reentrancy      : Non Reentrant
113   * \Parameters (in) : Channel
114   * \Parameters (out): void
115   * \Return value:   : void
116   *******************************************************************************/
117
118   Gpt_ValueType Gpt_GetTimeRemaining( Gpt_ChannelType Channel );
119
120   *******************************************************************************
121   * \Syntax          : Gpt_StartTimer( Gpt_ChannelType Channel, Gpt_ValueType Value )
122   * \Description     : Start a specific Timer by setting tick count
123   * \Sync\Async      : Synchronous
124   * \Reentrancy      : Non Reentrant
125   * \Parameters (in) : Channel,Value
126   * \Parameters (out): void
127   * \Return value:   : void
128   *******************************************************************************/
129
130   void Gpt_StartTimer( Gpt_ChannelType Channel, Gpt_ValueType Value );
131
132   *******************************************************************************
133   * \Syntax          : Gpt_StopTimer( Gpt_ChannelType Channel )
134   * \Description     : Stop a specific Timer
135   * \Sync\Async      : Synchronous
136   * \Reentrancy      : Non Reentrant
137   * \Parameters (in) : Channel
138   * \Parameters (out): void
139   * \Return value:   : void
140   *******************************************************************************/
141
142   void Gpt_StopTimer( Gpt_ChannelType Channel );
143
```

b)DIO

```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;

```

```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
*******************************************************************************/

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

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
* \Return value:   : DIO_PortLevelType
*******************************************************************************/

DIO_PortLevelType Dio_ReadPort(DIO_PortType PortId);

/*******************************************************************************
```

```
80   ********************************************************************************/
81
82   DIO_PortLevelType Dio_ReadPort(DIO_PortType PortId);
83
84   ********************************************************************************
85   * \Syntax          : Dio_WritePort(DIO_PortType PortId,DIO_PortLevelType Level)
86   * \Description     : Write on Port
87   * \Sync\Async      : Synchronous
88   * \Reentrancy      : Non Reentrant
89   * \Parameters (in) : PortId,Level
90   * \Parameters (out): void
91   * \Return value:   : void
92   ********************************************************************************/
93
94   void Dio_WritePort(DIO_PortType PortId,DIO_PortLevelType Level);
95
96   ********************************************************************************
97   * \Syntax          : Dio_FlipChannel(Channel_Id_Types ChannelId)
98   * \Description     : Flip Value on Pin
99   * \Sync\Async      : Synchronous
100  * \Reentrancy      : Non Reentrant
101  * \Parameters (in) :ChannelId
102  * \Parameters (out): DIO_LevelType
103  * \Return value:   : DIO_LevelType
104  ********************************************************************************/
105
106  DIO_LevelType Dio_FlipChannel(Channel_Id_Types ChannelId);
```

c)ADC

ADC_API.c

```
1   ********************************************************************************
2   * \Syntax          : void ADC_init(void)
3   * \Description     : Intialize ADC
4   * \Sync\Async      : Synchronous
5   * \Reentrancy      : Non Reentrant
6   * \Parameters (in) : void
7   * \Parameters (out): void
8   * \Return value:   : void
9   ********************************************************************************/
10
11  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  ********************************************************************************/
11
12  void CAN_Initialize(uint32 Channel_Id ,uint32 speed, uint32 linkingPort,uint32 interrupt);
13
14  ********************************************************************************
15  * \Syntax          : CAN_Write(uint32 Channel_Id,uint32 CAN_data)
16  * \Description     : Sending a CAN message
17  * \Sync\Async      : Synchronous
18  * \Reentrancy      : Non Reentrant
19  * \Parameters (in) : Channel_Id,CAN_data
20  * \Parameters (out): void
21  * \Return value:   : void
22  ********************************************************************************/
23
24  void CAN_Write(uint32 Channel_Id,uint32 CAN_data);
25
26  ********************************************************************************
27  * \Syntax          : CAN_Read(uint32 Channel_Id)
28  * \Description     : Reading a CAN message
29  * \Sync\Async      : Synchronous
30  * \Reentrancy      : Non Reentrant
31  * \Parameters (in) : Channel_Id
32  * \Parameters (out): uint32
33  * \Return value:   : uint32
34  ********************************************************************************/
35
36  uint32 CAN_Read(uint32 Channel_Id);
37
```

e) DOOR SENSOR

```c

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

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

Sensor_Read DoorSensor_ReadStatus( DoorSensor_Type Sensor_Num);

```

f)SPEED SENSOR

```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
*******************************************************************************/

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

Sensor_Read SpeedSensor_ReadStatus( DoorSensor_Type Sensor_Num);
```

---

g)OS

```
*******************************************************************************
* \Syntax          : OS_voidCreateTask(u8 Copy_u8ID, u8 Copy_u8Periodicity, u8 Copy_u8InitialDelay,
* \Description     : CreateTask
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) :  Copy_u8ID,  Copy_u8Periodicity,  Copy_u8InitialDelay, void (*ptr)(void)
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/


void OS_voidCreateTask(u8 Copy_u8ID, u8 Copy_u8Periodicity, u8 Copy_u8InitialDelay, void (*ptr)(voi



 *******************************************************************************
* \Syntax          : OS_voidDeleteTask(u8 Copy_u8ID)
* \Description     : DeleteTask
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Copy_u8ID
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/

void OS_voidDeleteTask(u8 Copy_u8ID);


*******************************************************************************
* \Syntax          : OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime)
* \Description     : SuspendTask
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Copy_u8ID,Copy_u8SuspendTime
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/

void OS_voidSuspendTask(u8 Copy_u8ID, u8 Copy_u8SuspendTime);
```

```
/*****************************************************************************
* \Syntax          : OS_voidStartScheduler(void)
* \Description     : StartScheduler
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
*****************************************************************************/

void OS_voidStartScheduler(void);


/*****************************************************************************
* \Syntax          : OS_voidResumeTask(u8 Copy_u8ID)
* \Description     : ResumeTask
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Copy_u8ID
* \Parameters (out): void
* \Return value:   : void
*****************************************************************************/

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

u8 OS_u8GetTaskState(u8 Copy_u8ID);
```

h)BCM

```
*******************************************************************************
* \Syntax          : BCM_init(void)
* \Description     : Intialize BCM
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/

void BCM_init(void);
```

I)DOOR STATE

```
1
2
3   *******************************************************************************
4   * \Syntax          : DoorSensor_SentState_10ms( DoorSensor_Type Sensor_Num)
5   * \Description     : Door state message will be sent every 10 ms to ECU 2
6   * \Sync\Async      : Synchronous
7   * \Reentrancy      : Non Reentrant
8   * \Parameters (in) : ConfigPtr
9   * \Parameters (out): void
10  * \Return value:   : void
11  *******************************************************************************/
12
13  void DoorSensor_SendState_10ms( DoorSensor_Type Sensor_Num);
```

J) SPEED SENSOR STATE

```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
*******************************************************************************/

void SpeedSensor_SendState_10ms( DoorSensor_Type Sensor_Num);
```

K)LIGHTSWITCH STATE

```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
*******************************************************************************/

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

void Switch_SendState_20ms(Channel_Id_Types Switch_Id );
```

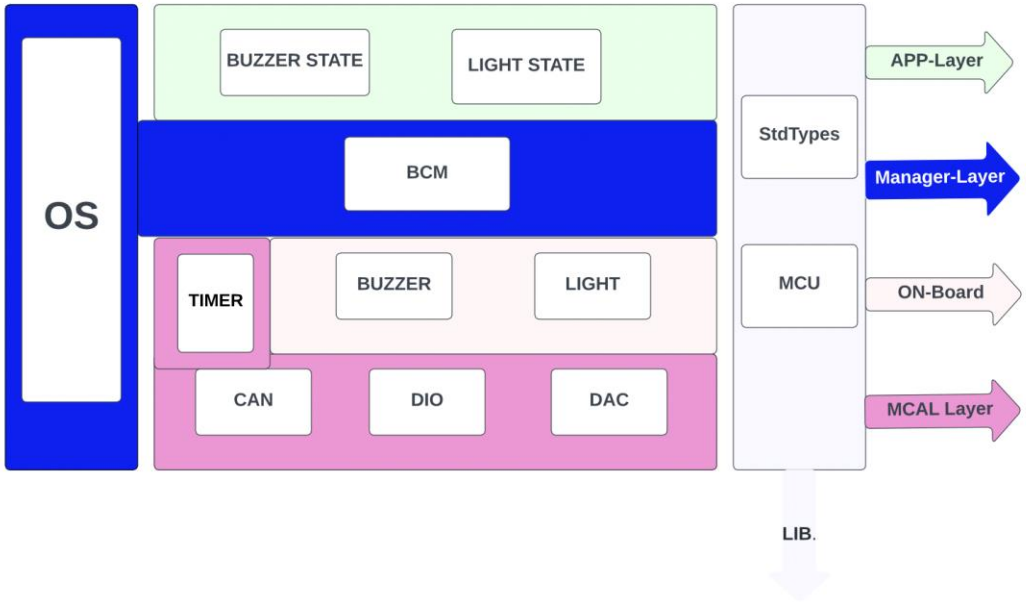
L)STD_TYPES

```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

c > ONBOARD > Inc > C Buzzer.h

```c
/*******************************************************************************
* \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
*******************************************************************************/

void Buzzer_init(Channel_Id_Types Buzzer_Id);
```

B)LED

src > ONBOARD > Inc > C LED_API.h

```c
/*******************************************************************************
* \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
*******************************************************************************/

void LED_init(Channel_Id_Types LED_Id);
```

C)DAC

src > MCAL > Inc > C DAC_API.h

```c
/*******************************************************************************
* \Syntax          : void DAC_init(void)
* \Description     : Intialize ADC
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : void
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/

void DAC_init(void);
```

D)BUZZER STATE

src > APP > Inc > C Buzzer_State_API.h

```c

/*******************************************************************************
* \Syntax          : Buzzer_updateState(uint8 Buzzer_Num)
* \Description     : Update Buzzer state according to door state and car state
* \Sync\Async      : Synchronous
* \Reentrancy      : Non Reentrant
* \Parameters (in) : Buzzer_Num
* \Parameters (out): void
* \Return value:   : void
*******************************************************************************/

void Buzzer_updateState(uint8 Buzzer_Num);
```

E)LED STATE

src > APP > Inc > C LEDState_API.h

```c
/*******************************************************************************
 * \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
 *******************************************************************************/

void LED_updateState(uint8 LED_Num);
```