AUTOMOTIVE DOOR CONTROL SYSTEM DESIGN

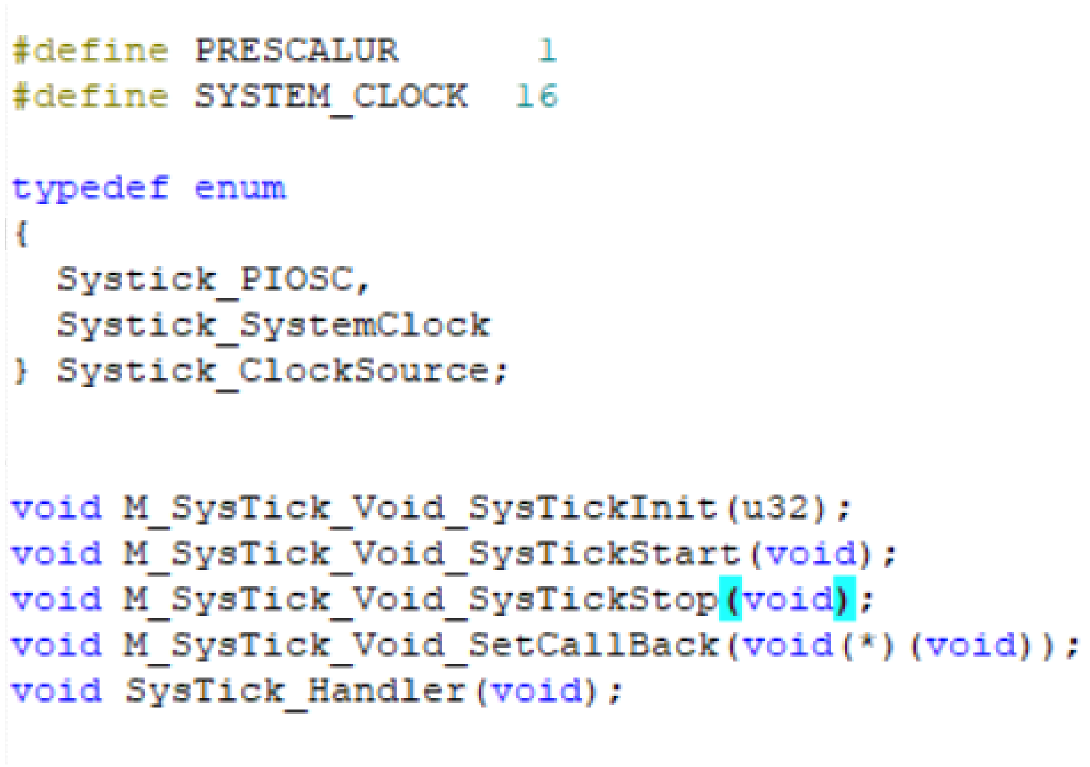
API DESCRIPTIONS

I- MCAL layer

1- Systick

The system timer module is used as the beating heart of the operating system, as well as to synchronize the communication between the two ECUs.

The following typedefs are been used:

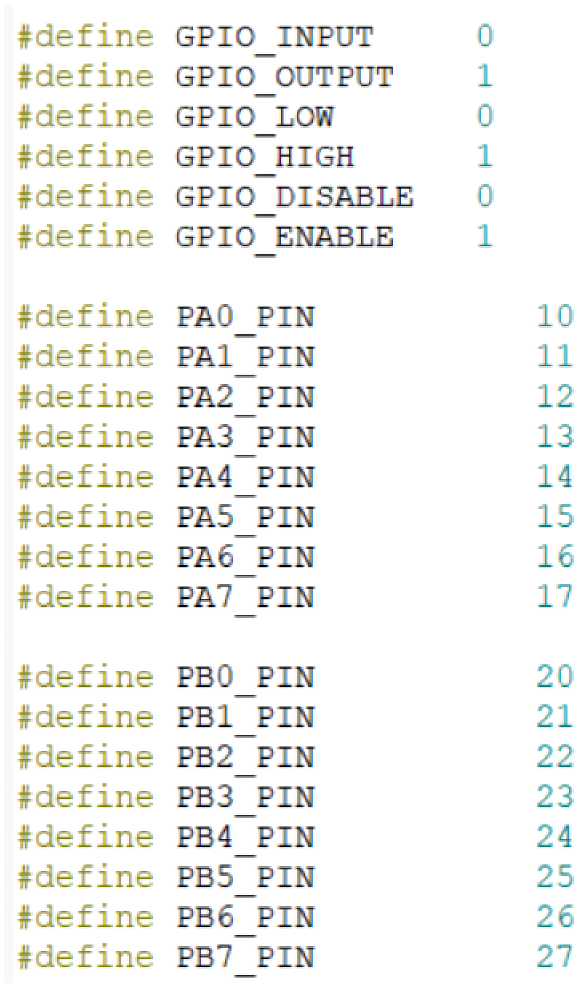


```
#define PRESCALUR       1
#define SYSTEM_CLOCK   16

typedef enum
{
   Systick_PIOSC,
   Systick_SystemClock
} Systick_ClockSource;


void M_SysTick_Void_SysTickInit(u32);
void M_SysTick_Void_SysTickStart(void);
void M_SysTick_Void_SysTickStop(void);
void M_SysTick_Void_SetCallBack(void(*)(void));
void SysTick_Handler(void);
```

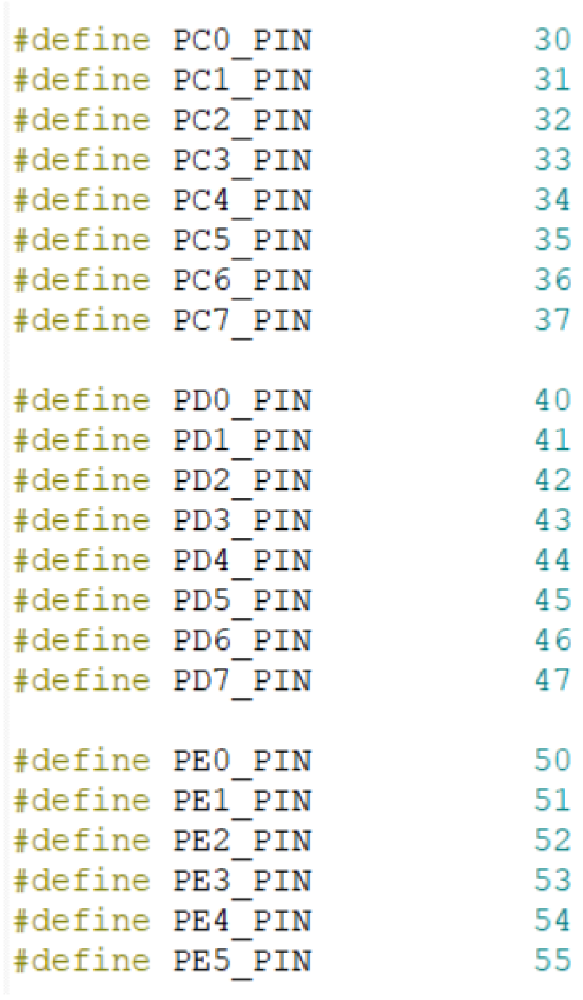
2-GPIO module

Input/Output ports will be used by ECU1 to control the sensors and send data to ECU2 by CAN. The APIs in this driver ensures the reading and writing of data from and to the digital pins and also controlling the external interrupts.

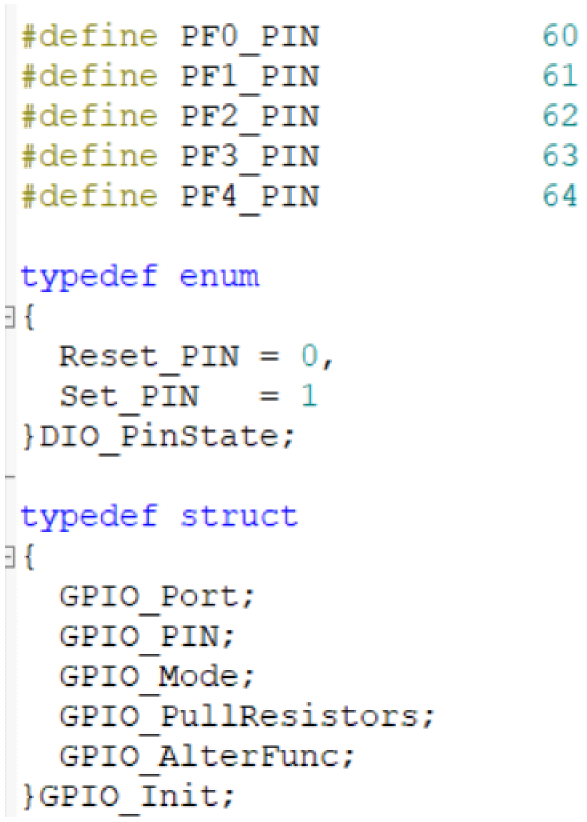
The typedefs and APIs as follow:



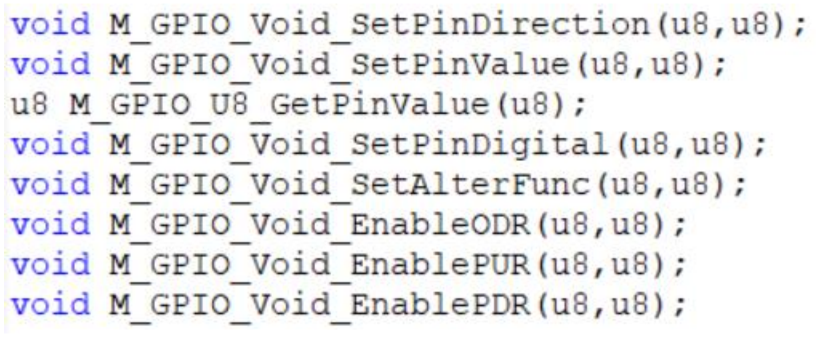
```
#define GPIO_INPUT      0
#define GPIO_OUTPUT     1
#define GPIO_LOW        0
#define GPIO_HIGH       1
#define GPIO_DISABLE    0
#define GPIO_ENABLE     1

#define PA0_PIN            10
#define PA1_PIN            11
#define PA2_PIN            12
#define PA3_PIN            13
#define PA4_PIN            14
#define PA5_PIN            15
#define PA6_PIN            16
#define PA7_PIN            17

#define PB0_PIN            20
#define PB1_PIN            21
#define PB2_PIN            22
#define PB3_PIN            23
#define PB4_PIN            24
#define PB5_PIN            25
#define PB6_PIN            26
#define PB7_PIN            27
```



```
#define PC0_PIN            30
#define PC1_PIN            31
#define PC2_PIN            32
#define PC3_PIN            33
#define PC4_PIN            34
#define PC5_PIN            35
#define PC6_PIN            36
#define PC7_PIN            37

#define PD0_PIN            40
#define PD1_PIN            41
#define PD2_PIN            42
#define PD3_PIN            43
#define PD4_PIN            44
#define PD5_PIN            45
#define PD6_PIN            46
#define PD7_PIN            47

#define PE0_PIN            50
#define PE1_PIN            51
#define PE2_PIN            52
#define PE3_PIN            53
#define PE4_PIN            54
#define PE5_PIN            55
```



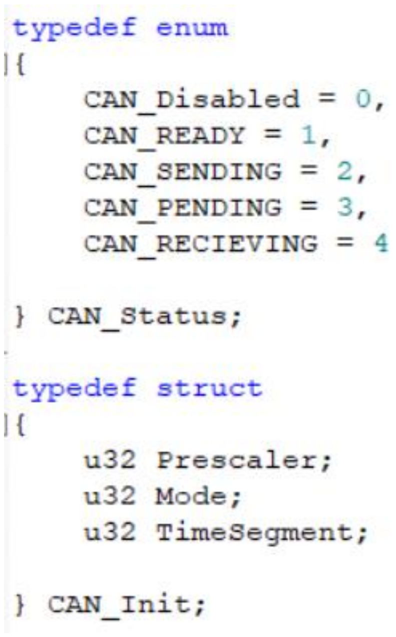
```c
#define PF0_PIN          60
#define PF1_PIN          61
#define PF2_PIN          62
#define PF3_PIN          63
#define PF4_PIN          64

typedef enum
{
  Reset_PIN = 0,
  Set_PIN   = 1
}DIO_PinState;

typedef struct
{
  GPIO_Port;
  GPIO_PIN;
  GPIO_Mode;
  GPIO_PullResistors;
  GPIO_AlterFunc;
}GPIO_Init;
```



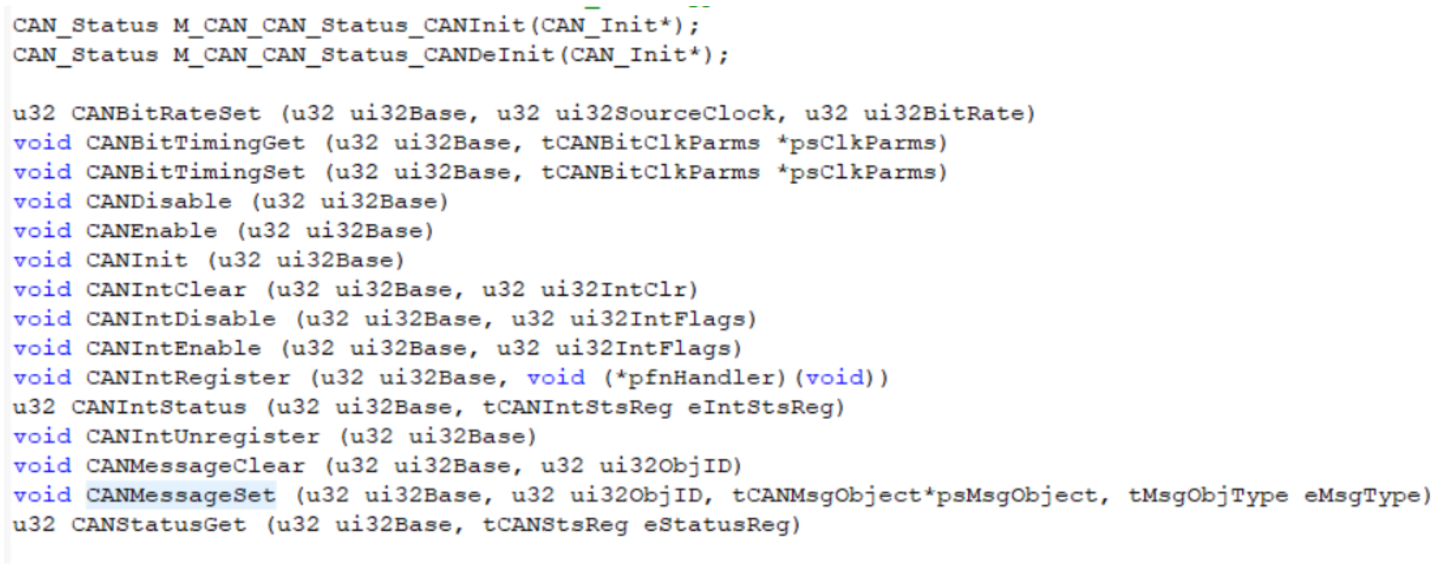
```c
void M_GPIO_Void_SetPinDirection(u8,u8);
void M_GPIO_Void_SetPinValue(u8,u8);
u8 M_GPIO_U8_GetPinValue(u8);
void M_GPIO_Void_SetPinDigital(u8,u8);
void M_GPIO_Void_SetAlterFunc(u8,u8);
void M_GPIO_Void_EnableODR(u8,u8);
void M_GPIO_Void_EnablePUR(u8,u8);
void M_GPIO_Void_EnablePDR(u8,u8);
```

3-CAN driver

The CAN communication protocol is the one used to transfer the data between the two ECUs. Its typedefs and APIs as follow:



```
typedef enum
{
    CAN_Disabled = 0,
    CAN_READY = 1,
    CAN_SENDING = 2,
    CAN_PENDING = 3,
    CAN_RECIEVING = 4

} CAN_Status;

typedef struct
{
    u32 Prescaler;
    u32 Mode;
    u32 TimeSegment;

} CAN_Init;
```

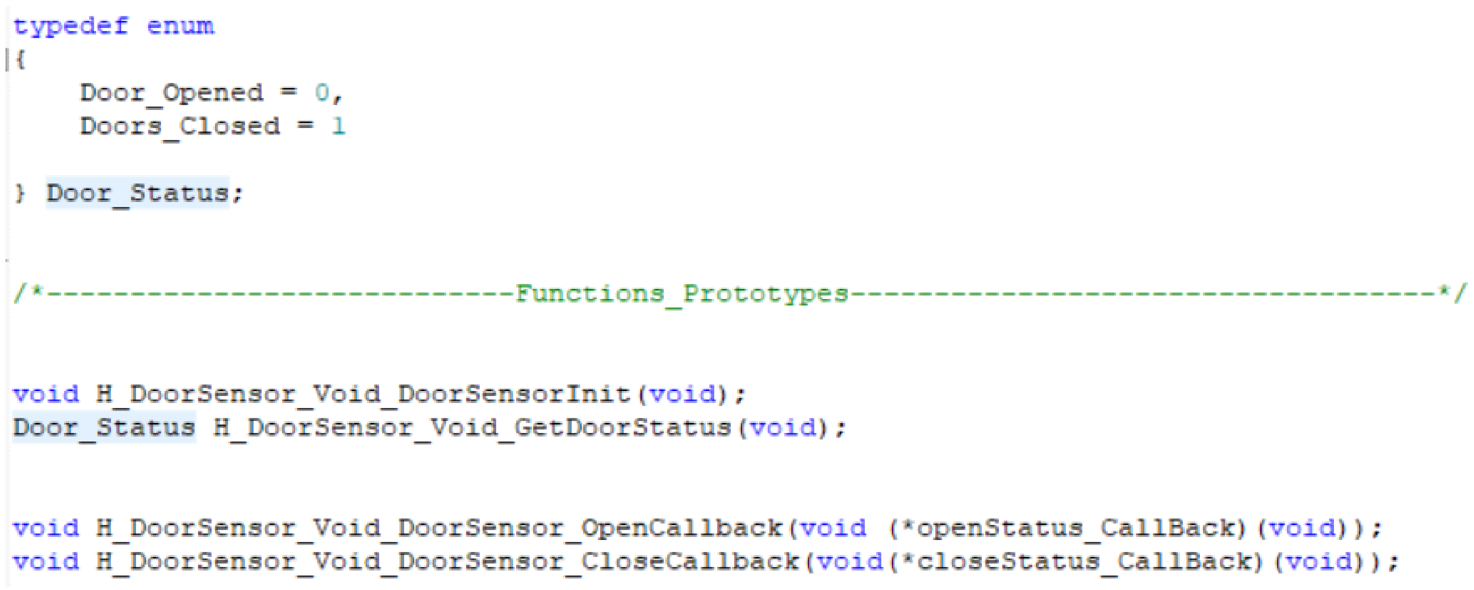


```
CAN_Status M_CAN_CAN_Status_CANInit(CAN_Init*);
CAN_Status M_CAN_CAN_Status_CANDeInit(CAN_Init*);

u32 CANBitRateSet (u32 ui32Base, u32 ui32SourceClock, u32 ui32BitRate)
void CANBitTimingGet (u32 ui32Base, tCANBitClkParms *psClkParms)
void CANBitTimingSet (u32 ui32Base, tCANBitClkParms *psClkParms)
void CANDisable (u32 ui32Base)
void CANEnable (u32 ui32Base)
void CANInit (u32 ui32Base)
void CANIntClear (u32 ui32Base, u32 ui32IntClr)
void CANIntDisable (u32 ui32Base, u32 ui32IntFlags)
void CANIntEnable (u32 ui32Base, u32 ui32IntFlags)
void CANIntRegister (u32 ui32Base, void (*pfnHandler)(void))
u32 CANIntStatus (u32 ui32Base, tCANIntStsReg eIntStsReg)
void CANIntUnregister (u32 ui32Base)
void CANMessageClear (u32 ui32Base, u32 ui32ObjID)
void CANMessageSet (u32 ui32Base, u32 ui32ObjID, tCANMsgObject*psMsgObject, tMsgObjType eMsgType)
u32 CANStatusGet (u32 ui32Base, tCANStsReg eStatusReg)
```

II- HAL layer

1-Door sensor (ECU1)

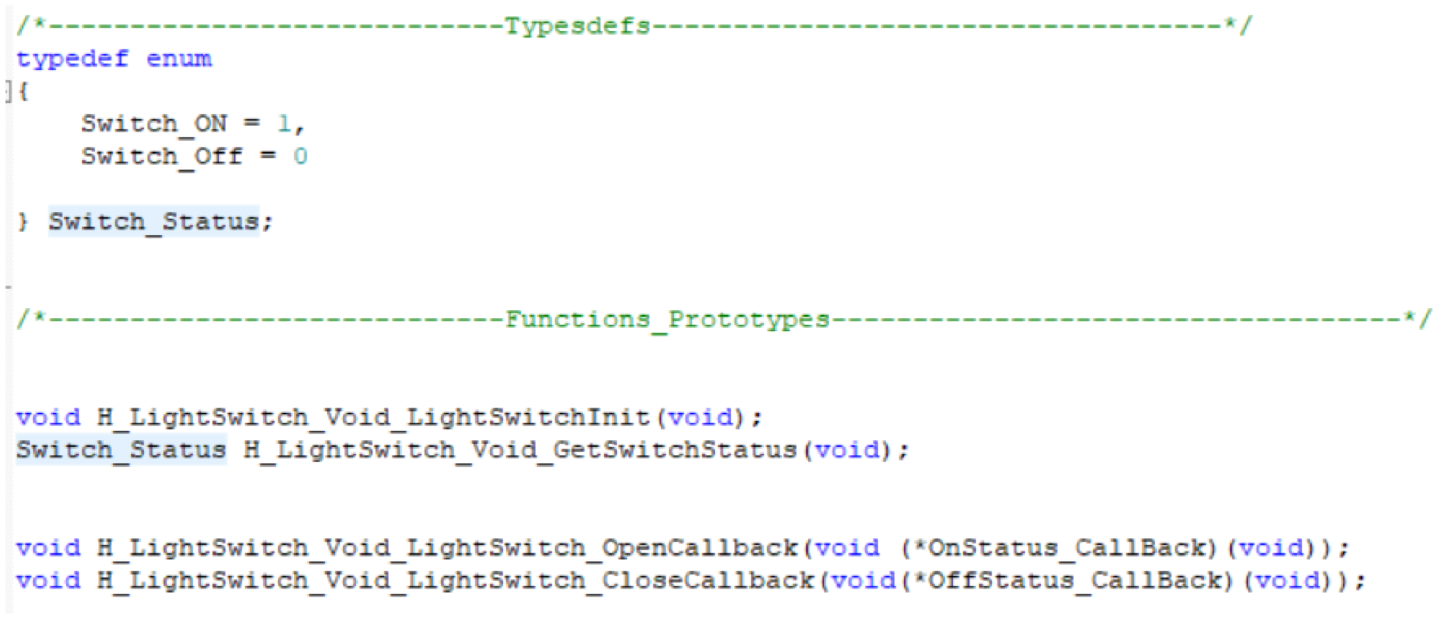
The door sensor driver is the one handling the readings of the door sensor at a periodic rate and ensuring no loss of information. Its typedefs and APIs as follow:



```
typedef enum
{
    Door_Opened = 0,
    Doors_Closed = 1

} Door_Status;


/*----------------------------Functions_Prototypes-----------------------------------*/


void H_DoorSensor_Void_DoorSensorInit(void);
Door_Status H_DoorSensor_Void_GetDoorStatus(void);


void H_DoorSensor_Void_DoorSensor_OpenCallback(void (*openStatus_CallBack)(void));
void H_DoorSensor_Void_DoorSensor_CloseCallback(void(*closeStatus_CallBack)(void));
```

2-Light Switch driver (ECU1)

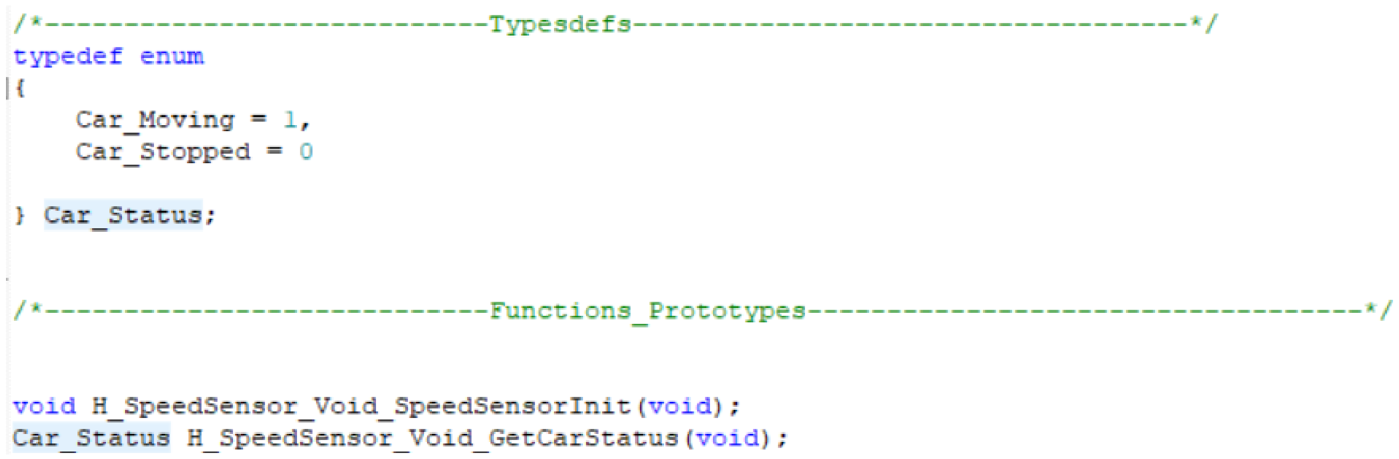
The light switch driver is the one responsible for the external interrupt of the light switch and handling its reading once this interrupt is fired. Its typedefs and APIs as follow:



```
/*-----------------------------Typesdefs------------------------------------*/
typedef enum
{
    Switch_ON = 1,
    Switch_Off = 0

} Switch_Status;


/*-----------------------------Functions_Prototypes------------------------------------*/


void H_LightSwitch_Void_LightSwitchInit(void);
Switch_Status H_LightSwitch_Void_GetSwitchStatus(void);


void H_LightSwitch_Void_LightSwitch_OpenCallback(void (*OnStatus_CallBack)(void));
void H_LightSwitch_Void_LightSwitch_CloseCallback(void(*OffStatus_CallBack)(void));
```

3- Speed sensor (ECU1)

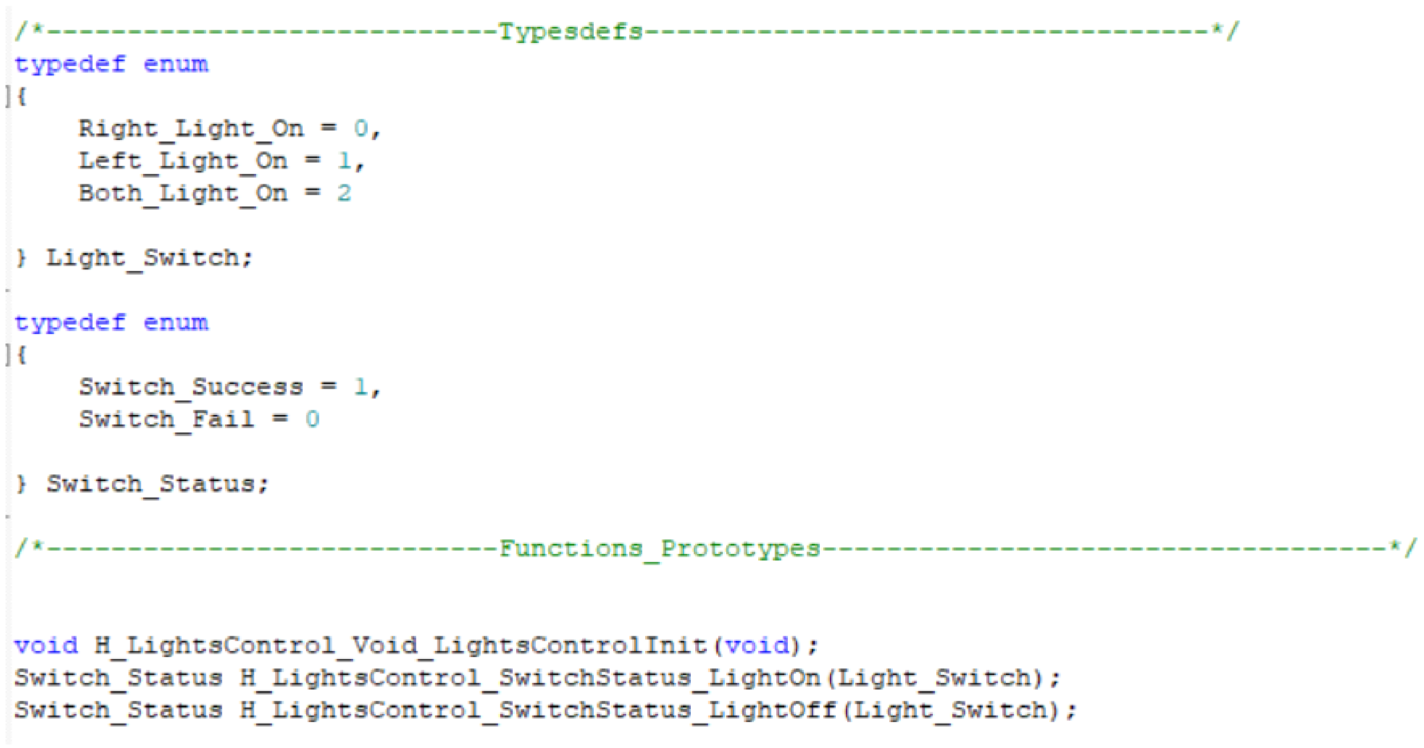
The speed sensor driver is the one reading the speed sensor data and sending it to the communication manager at a periodic rate. Its typedefs and APIs as follow:



```
/*----------------------------Typesdefs-----------------------------------*/
typedef enum
{
    Car_Moving = 1,
    Car_Stopped = 0

} Car_Status;


/*----------------------------Functions_Prototypes-----------------------------------*/


void H_SpeedSensor_Void_SpeedSensorInit(void);
Car_Status H_SpeedSensor_Void_GetCarStatus(void);
```

4-Lights control (ECU2)

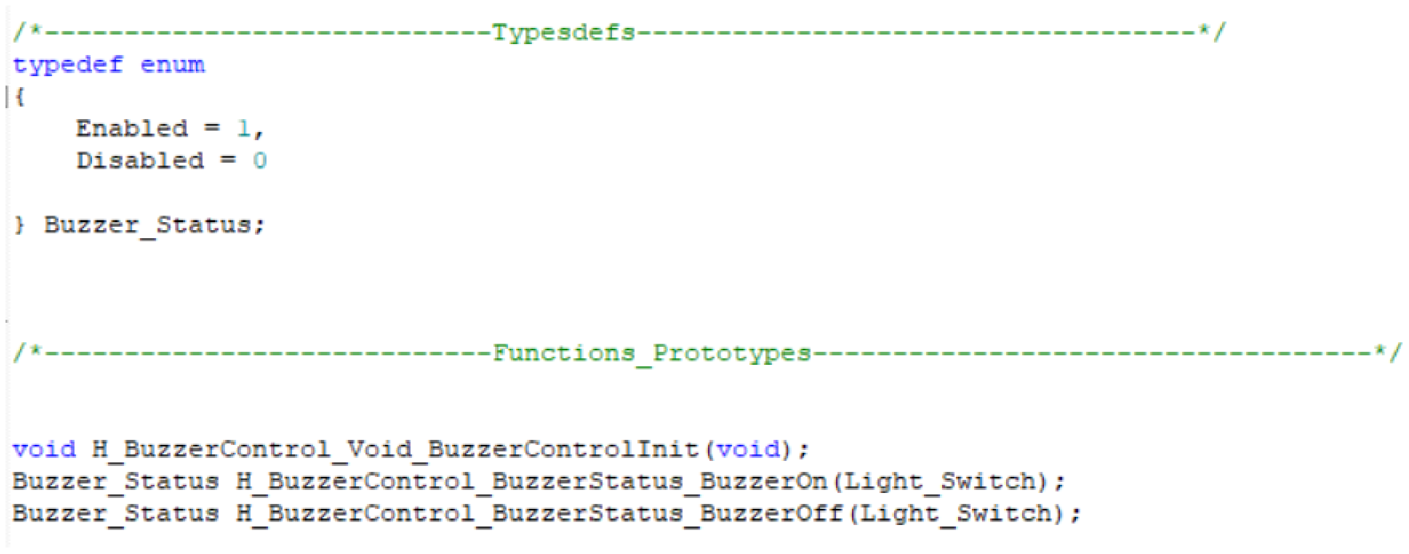
The lights control driver is the one responsible for turning the car lights on or off based on data transferred from ECU1 and processed by the data handler. Its typedefs and APIs as follow:



```c
/*----------------------------Typesdefs------------------------------------*/
typedef enum
{
    Right_Light_On = 0,
    Left_Light_On = 1,
    Both_Light_On = 2

} Light_Switch;

typedef enum
{
    Switch_Success = 1,
    Switch_Fail = 0

} Switch_Status;

/*----------------------------Functions_Prototypes------------------------------------*/


void H_LightsControl_Void_LightsControlInit(void);
Switch_Status H_LightsControl_SwitchStatus_LightOn(Light_Switch);
Switch_Status H_LightsControl_SwitchStatus_LightOff(Light_Switch);
```

5-Buzzer Control (ECU2)

The buzzer control driver is the one responsible for the alarm system in the car and it is triggered by the data handler based on information from ECU1. Its typedefs and APIs as follow:



```
/*----------------------------Typesdefs-----------------------------------*/
typedef enum
{
    Enabled = 1,
    Disabled = 0

} Buzzer_Status;



/*----------------------------Functions_Prototypes-----------------------------------*/


void H_BuzzerControl_Void_BuzzerControlInit(void);
Buzzer_Status H_BuzzerControl_BuzzerStatus_BuzzerOn(Light_Switch);
Buzzer_Status H_BuzzerControl_BuzzerStatus_BuzzerOff(Light_Switch);
```