

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);
```

	Input	Output	Description
M_SysTick_Void _SysTickInit(u32)	The time interval required	No output	This API initializes the SysTick (setting up its clock source & interrupt) and taking its initial time interval as input
M_SysTick_Void _SysTickStart(void)	No input	No output	This API starts the systick using initial interval given in the initialization
M_SysTick_Void _SysTickStop(void)	No input	No output	This API stops the systick timer
M_SysTick_Void _SetCallBack (void(*)(void))	Pointer to a function	No output	This API selects the function in the APP layer which the SysTick will call in case of interrupt
SysTick_Handler (void)	No input	No output	This is the interrupt API of the SysTick which will be called upon interrupt and then call back the function in the APP layer

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
```

```

#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;

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);

```

	Input	Output	Description
M_GPIO_Void_SetPinDirection(u8, u8)	The number of the Pin as displayed in the #defines and the Pin type (Input or output)	No output	This API sets the required as input or output based on the user input
M_GPIO_Void_SetPinValue(u8, u8)	The number of the Pin as displayed in the #defines and the pin value (Low or high)	No output	This API sets the required as low or high based on the user input
M_GPIO_U8_GetPinValue(u8)	The number of the pin as displayed in the #defines	The value of the required pin (low	This API reads the required pin and returns its state (low or high)

		or high)	
M_GPIO_Void_ SetPinDigital (u8 ,u8)	The number of the pin as displayed in the #defines and its state (Digital or analog)	No output	This API sets the required as digital or analog by setting or clearing the GPIODEN reg respectively
M_GPIO_Void_ SetAlterFunc(u8, u8)	The number of the pin as displayed in the #defines and the number of the required alternate function	No output	This API selects the required alternate function the pin as illustrated in table 10-2 in the datasheet
M_GPIO_Void_ EnableODR(u8 ,u8)	The number of the pin as displayed in the #define and whether the ODR should be enabled	No output	This API enable or disable the Open Drain configuration for the required pin
M_GPIO_Void_ EnablePUR(u8 ,u8)	The number of the pin as displayed in the #defines and whether the pull up resistor should be enabled	No output	This API sets a weak pull up resistor on the corresponding pin
M_GPIO_Void_ EnablePDR (u8 , u8)	The number of the pin as displayed in the #defines and whether the pull down resistor should be enabled	No output	This API sets a weak pull down resistor on the corresponding pin

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)
```

	Input	Output	Description
M_CAN_CAN Status_CANInit (CAN_Init*)	Pointer to the initialization struct	Typedef enum with the required initialization parameters	This API initializes the CAN communication module with the required parameters
M_CAN_CAN_ Status_CANDeInit (void)	No input	No output	This API deinitializes the CAN communication module
CANBitRateSet (u32 ui32Base, u32 ui32SourceClock, u32 ui32BitRate)	The base, source clock and the required bitrate of the can bus	No output	This API sets the rate of the CAN bus based on the user input
CANEnable (u32 ui32Base)	Logic zero or one	No output	This API enables the CAN module
CANDisable (u32 ui32Base)	Logic zero or one	No output	This API disables the CAN module
CANMessageSet (u32 ui32Base, u32 ui32 ObjID,tCANMsgObject *psMsgObject,tMsg ObjType eMsgType)	The base, ID for the message, pointer to the message and message type	No output	This API sends a message on the CAN bus
CANMessageClear (u32 ui32Base, u32 ui32ObjID)	The base and ID of message	No output	This API clears a message on the CAN bus

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));
```

	Input	Output	Description
H_DoorSensor_Void_DoorSensorInit(void)	No input	No output	This API initialize the door sensor
H_DoorSensor_Void_GetDoorStatus(void)	No input	Typedef enum object	This API returns the reading of the door sensor
H_DoorSensor_Void_DoorSensor_OpenCallback(void(*openStatus_Callback)(void))	Pointer to a function	No output	This call back function selects the API which will execute if the door open
H_DoorSensor_Void_DoorSensor_CloseCallback(void(*closeStatus_Callback)(void))	Pointer to a function	No output	This call back function selects the API which will execute if the door closed

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));
```

	Input	Output	Description
H_LightSwitch_Void_LightSwitchInit(void)	No input	No output	This API initialize the light switch
H_LightSwitch_Void_GetSwitchStatus(void)	No input	Typedef enum object	This API returns the reading of the light switch
H_LightSwitch_Void_LightSwitch_OpenCallback(void(*OnStatus_CallBack)(void))	Pointer to a function	No output	This call back function selects the API which will execute if the switch is pressed
H_LightSwitch_Void_LightSwitch_CloseCallback(void(*OffStatus_CallBack)(void))	Pointer to a function	No output	This call back function selects the API which will execute if the switch released

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);
```

	Input	Output	Description
H_SpeedSensor_Void_SpeedSensorInit(void)	No input	No output	This API initialize the speed sensor
H_SpeedSensor_Void_GetCarStatus(void)	No input	Typedef enum object	This API returns the reading of the speed sensor

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:

```
/*-----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);
```

	Input	Output	Description
H_LightsControl_Void_LightsControlInit(void)	No input	No output	This API initialize the lights control module
H_LightsControl_Switch_Status_LightOn(Light_Switch)	Typedef enum of the lights type	Typedef enum object	This API turns the selected lights on
H_LightsControl_Switch_Status_LightOff(Light_Switch)	Typedef enum of the lights type	Typedef enum object	This API turn the selected lights off

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);
```

	Input	Output	Description
H_BuzzerControl_Void_BuzzerControlInit(void)	No input	No output	This API initialize the buzzer control module
H_BuzzerControl_BuzzerStatus_BuzzerOn(Light_Switch)	Typedef enum of the lights type	Typedef enum object	This API turns the buzzer on
H_BuzzerControl_BuzzerStatus_BuzzerOff(Light_Switch)	Typedef enum of the lights type	Typedef enum object	This API turn the buzzer off