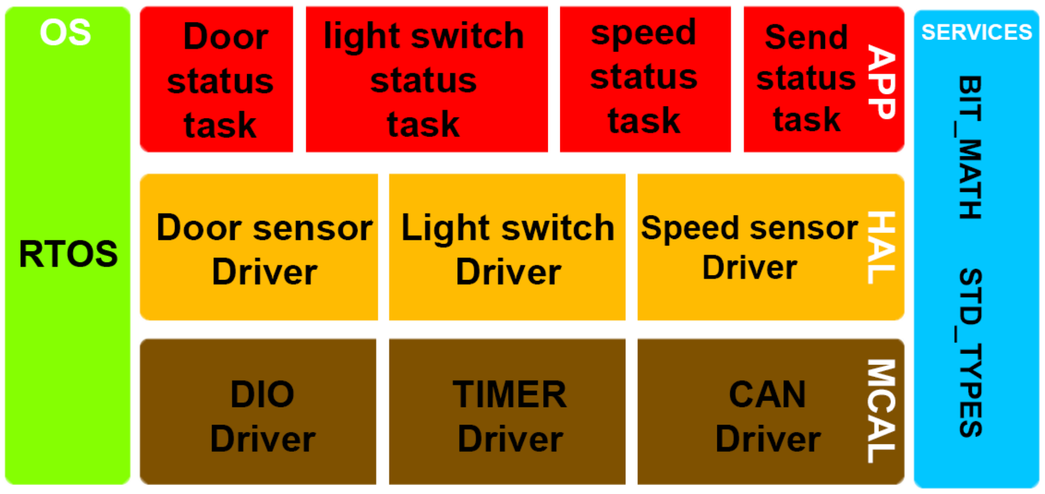
Layered Architecture for MCU1



ECU1 Components

1. MCAL

- **DIO Driver:**
  Driver interface with ECU DIO peripheral which interduce DIO_Set, DIO_Reset, DIO_Toggle, DIO_Read APIs to upper layers.

- **TIMER Driver:**
  Driver interface with ECU TIMER peripheral which interduce TIMER_Init, TIMERDelay_ms, Interrupts set call back APIs to upper layers.

- **CAN Driver:**
  Driver interface with ECU CAN peripheral which interduce CAN_Init, CAN_Read, CAN_Write APIs to upper layers

2. HAL

- **Door sensor Driver:**
  Module responsible for interface with speed sensor providing API DoorSensorRead
  to get Door status.

- **Light switch Driver:**
  Module responsible for interface with Light switch providing API LightSwitchStatus
  to get switch status.

- **Speed sensor Driver**:
  Module responsible for interface with speed sensor providing API SpeedSensorRead
  to get car speed status.

3. SERVICES LAYER

- **BIT_MATH:**
  A header file contains functions like macros which perform bit manipulation SET_BIT, RESET_BIT, TOGGLE_BIT, GETVAL_BIT.

- **STD_TYPES:**
  A header file contains standard types u8, u16, u32, u64, s8, s16, s32, s64, f32, f64.

4. APPLICATION LAYER

The system has 4 tasks and 1 queue

- **Door status task:**
  The task is responsible for getting the current door status then adding the current door status to status queue.

- **Light switch status task:**
  The task is responsible for getting the current switch status then adding the current switch status to status queue.

- **Speed status task**:
  The task is responsible for getting the current speed status then adding the current speed status to status queue.

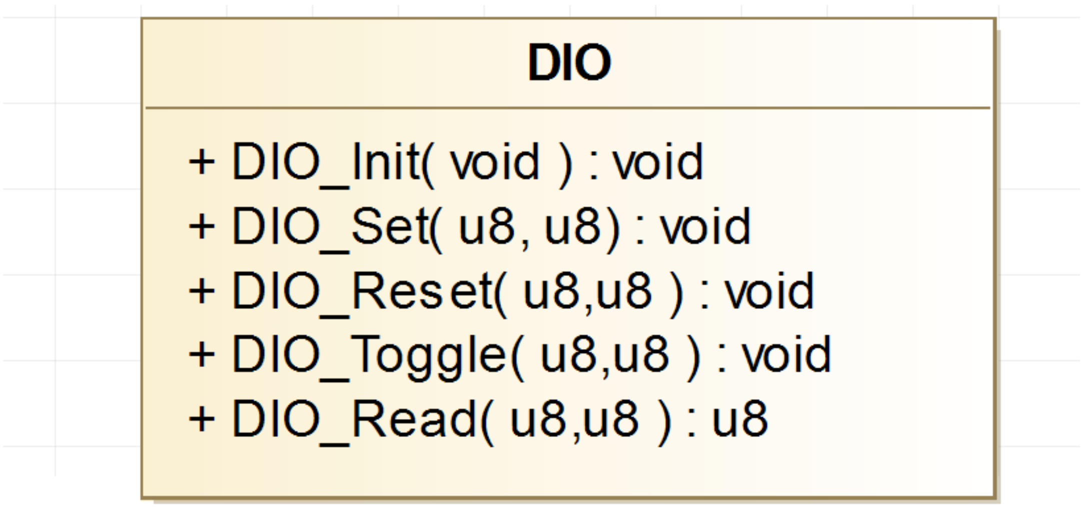
- **Send status task**:
  The task is responsible for sending the system status stored in status queue to MCU2 through CAN bus.

- **Status queue**:
  Intercommunication to sync status messages form the tasks to Send status task**.**

ECU1 APIs Discerption

1. MCAL

- **DIO Driver:**


DIO
+ DIO_Init(void) : void
+ DIO_Set(u8, u8) : void
+ DIO_Reset(u8,u8) : void
+ DIO_Toggle(u8,u8) : void
+ DIO_Read(u8,u8) : u8

DIO_Init

Description: This API use to initialize DIO ports directions

Arguments: void

Return: void

DIO_Set

Description: This API use to set the value of a DIO bin

Arguments: DIO_PORT,BIN_NUM

Return: void

DIO_Reset

Description: This API use to Reset the value of a DIO bin

Arguments: DIO_PORT,BIN_NUM

Return: void

DIO_Toggle

Description: This API use to Toggle the value of a DIO bin

Arguments: DIO_PORT,BIN_NUM

Return: void

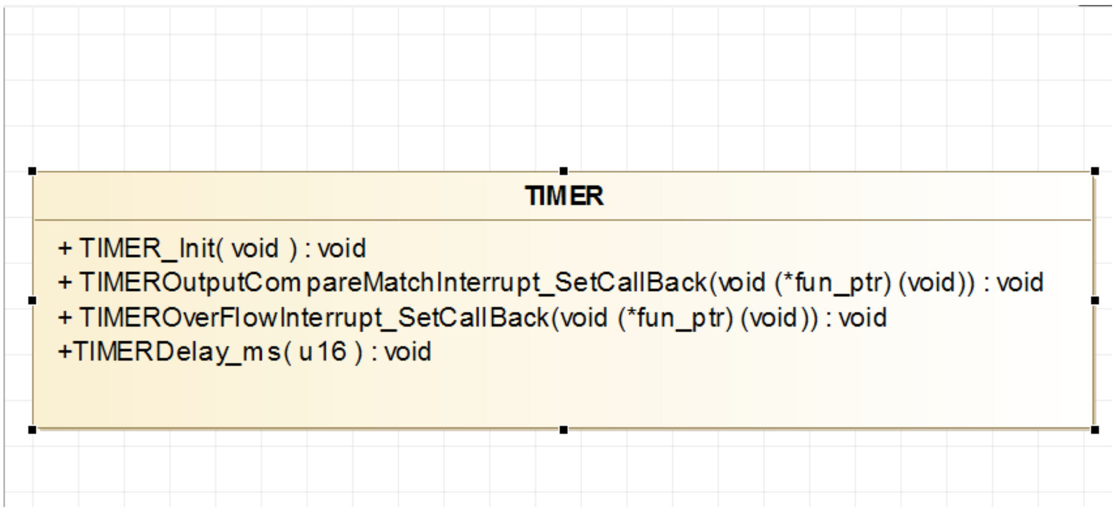
DIO_Read

Description: This API use to Read the value of a DIO bin

Arguments: DIO_PORT,BIN_NUM

Return: Boolean 0 or 1

- **TIMER Driver:**



```
+ TIMER_Init( void ) : void
+ TIMEROutputCompareMatchInterrupt_SetCallBack(void (*fun_ptr) (void)) : void
+ TIMEROverFlowInterrupt_SetCallBack(void (*fun_ptr) (void)) : void
+TIMERDelay_ms( u16 ) : void
```

TIMER_Init

Description: This API use to initialize TIMER tick time

Arguments: void

Return: void

TIMEROutpoutCompareMatchInterrupt_SetCallBack

Description: This API use to assign the address of timer compare match interrupt handler in his right place in vector table

Arguments: pointer to timer compare match interrupt handler

Return: void

TIMEROverFlowInterrupt_SetCallBack

Description: This API use to assign the address of timer over flow interrupt handler in his right place in vector table

Arguments: pointer to timer over flow interrupt handler

Return: void

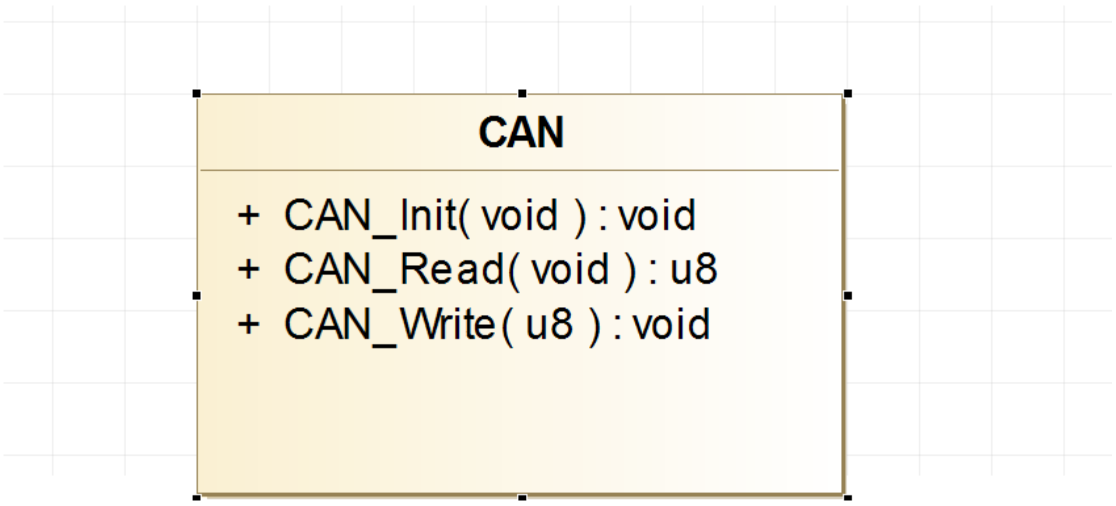
TIMERDelay_ms

Description: This API use to a delay using timer in milliseconds

Arguments: u16 which represent the desired delay in milliseconds

Return: void

- **CAN Driver:**



CAN_Init

Description: This API use to initialize CAN

Arguments: void

Return: void

CAN_Read

Description: This API use to read data sent through CAN bus

Arguments: void

Return: u8 data

CAN_Write

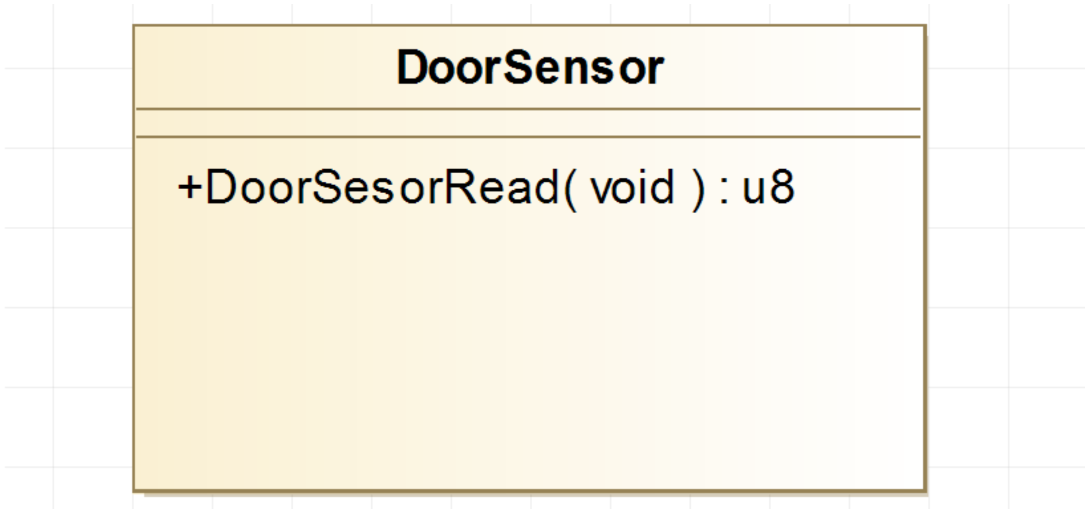
Description: This API use to write data through CAN bus

Arguments: u8 data

Return: void

2. HAL

- **Door sensor Driver:**



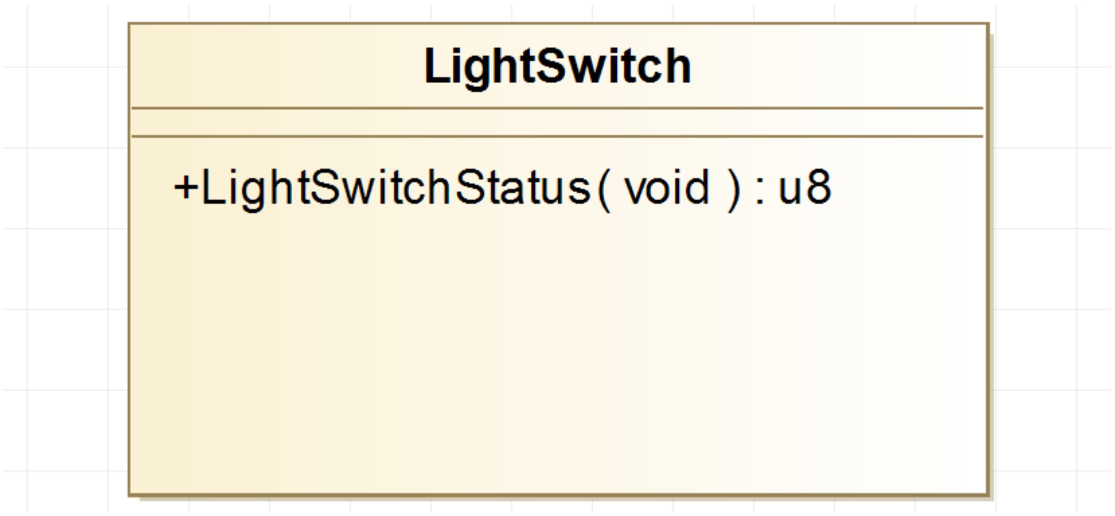
DoorSensorRead

Description: This API use to get the read of door sensor open or close

Arguments: void

Return: u8 which represent the read of the door sensor

- **Light switch Driver:**



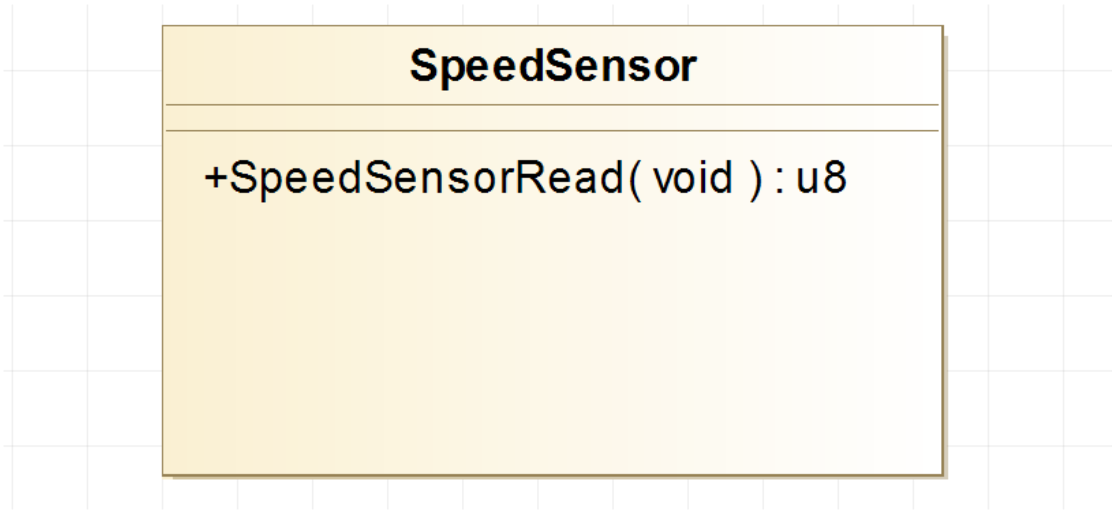
LightSwitchStatus

Description: This API use to get status of light switch pressed or not pressed

Arguments: void

Return: u8 which represent the status of light switch

Speed sensor Driver:



SpeedSensorRead

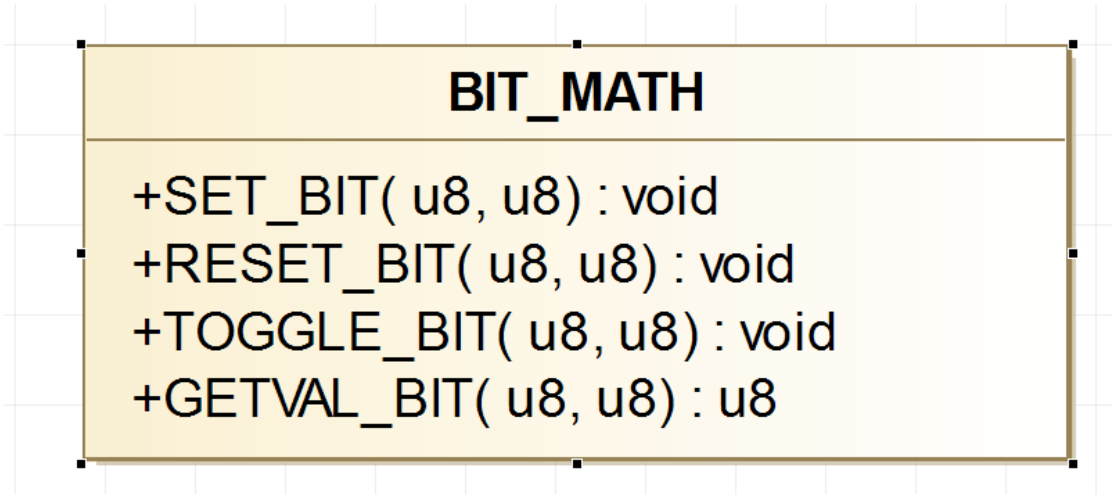
Description: This API use to get the read of speed sensor move or not move

Arguments: void

Return: u8 which represent the read of the speed sensor

3. SERVICES Layer

- **BIT_MATH:**


BIT_MATH
+SET_BIT(u8, u8) : void
+RESET_BIT(u8, u8) : void
+TOGGLE_BIT(u8, u8) : void
+GETVAL_BIT(u8, u8) : u8

SET_BIT

Description: This is a function like macro which be replaced with reg_name|=(1<<bit_num) to set the value of a DIO bin

Arguments: reg_name which represent a pointer to the address of the register
bit_num which represent the bit number in the same register

Return: void

RESET_BIT

Description: This is a function like macro which be replaced with reg_name&=~(1<<bit_num) to reset the value of a DIO bin

Arguments: reg_name which represent a pointer to the address of the register bit_num which represent the bit number in the same register

Return: void

TOGGLE_BIT

Description: This is a function like macro which be replaced with reg_name^=(1<<bit_num) to toggle the value of a DIO bin

Arguments: reg_name which represent a pointer to the address of the register

bit_num which represent the bit number in the same register

Return: void

GITVAL_BIT

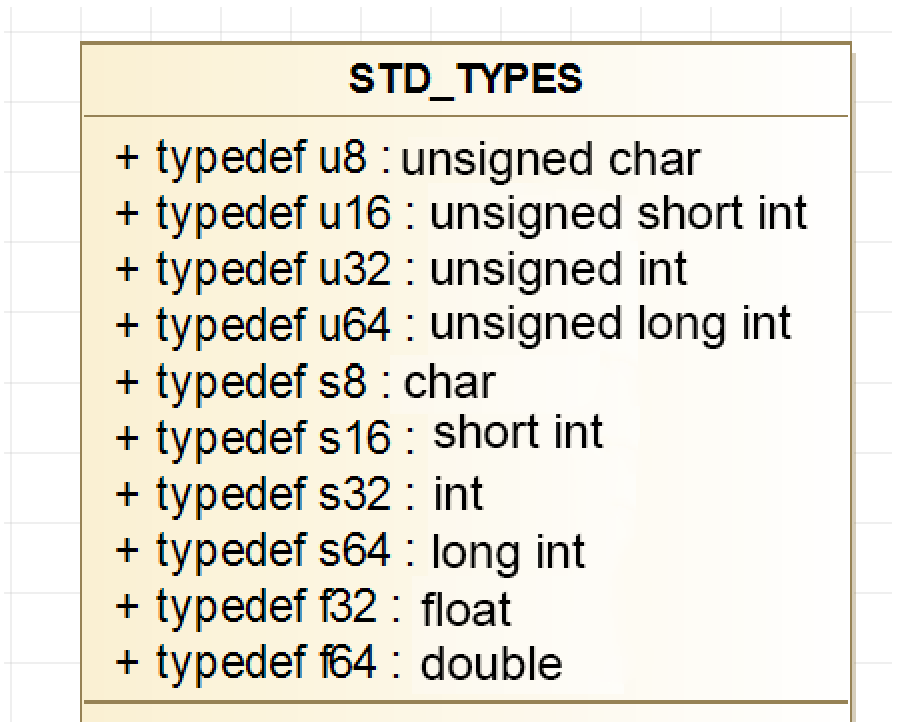
Description: This is a function like macro which be replaced with (reg_name>>bit_num)&1 to get the value of a DIO bin

Arguments: reg_name which represent a pointer to the address of the register

bit_num which represent the bit number in the same register

Return: u8 which represent the bin value

- **STD_TYPES:**



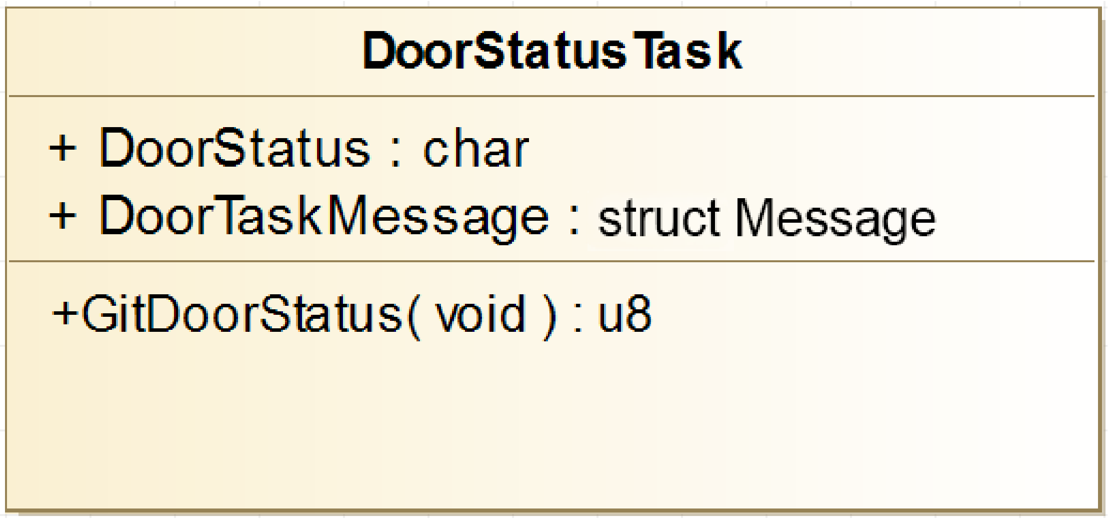
STD_TYPES

```
+ typedef u8 : unsigned char
+ typedef u16 : unsigned short int
+ typedef u32 : unsigned int
+ typedef u64 : unsigned long int
+ typedef s8 : char
+ typedef s16 : short int
+ typedef s32 : int
+ typedef s64 : long int
+ typedef f32 : float
+ typedef f64 : double
```

Description: A header file contains standard types u8, u16, u32, u64, s8, s16, s32, s64, f32, f64.

4. APPLICATION Layer

- **DoorStatusTask:**


DoorStatus Task
+ DoorStatus : char
+ DoorTaskMessage : struct Message
+GitDoorStatus(void) : u8

GitDoorStatus

Description: This is a function use to get the status of door open or close

Arguments: void

Return: u8 which is a variable contain the door status

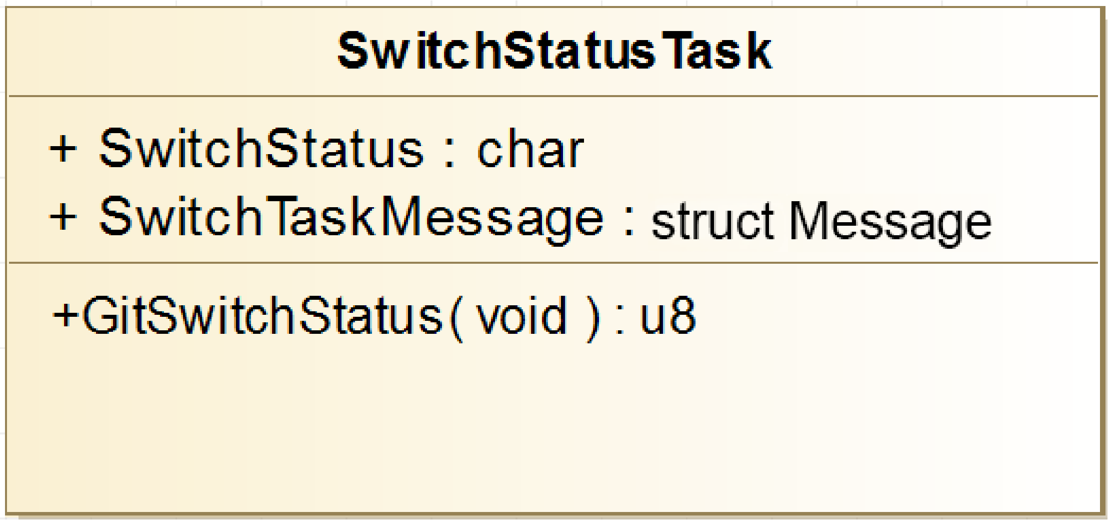
DoorStatus

Description: a variable contains the door status

DoorTaskMessage

Description: a structure contains the message which represent the door status also contain the task id

- **SwitchStatusTask:**


SwitchStatusTask
+ SwitchStatus : char
+ SwitchTaskMessage : struct Message
+GitSwitchStatus(void) : u8

GitSwitchStatus

Description: This is a function use to get the status of light switch pressed or not pressed

Arguments: void

Return: u8 which is a variable contain the light switch status

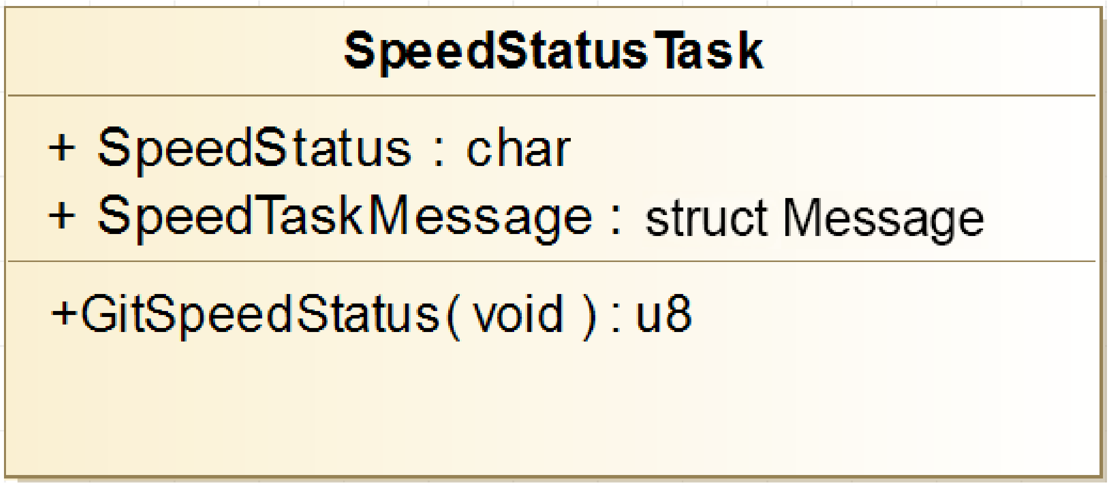
SwitchStatus

Description: a variable contains the light switch status

SwitchTaskMessage

Description: a structure contains the message which represent the switch status also contain the task id

- **SpeedStatusTask:**


SpeedStatus Task
+ SpeedStatus : char
+ SpeedTaskMessage : struct Message
+GitSpeedStatus(void) : u8

GitSpeedStatus

Description: This is a function use to get the status of the car move or not move

Arguments: void

Return: u8 which is a variable contain status of car speed

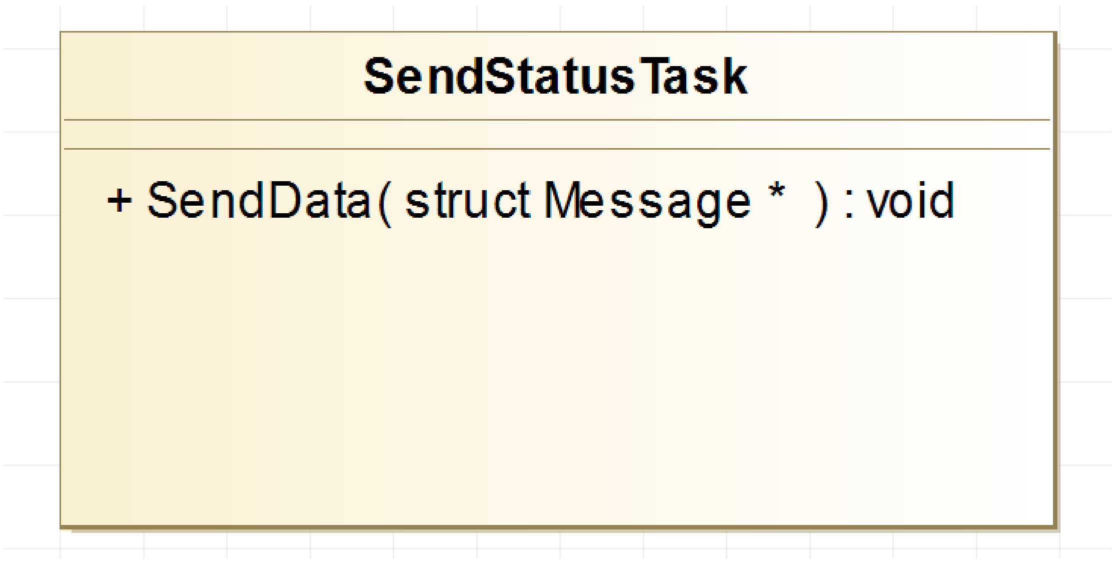
SpeedStatus

Description: a variable contains the status of car speed

DoorTaskMessage

Description: a structure contains the message which represent the speed status also contain the task id

- **SendStatusTask:**



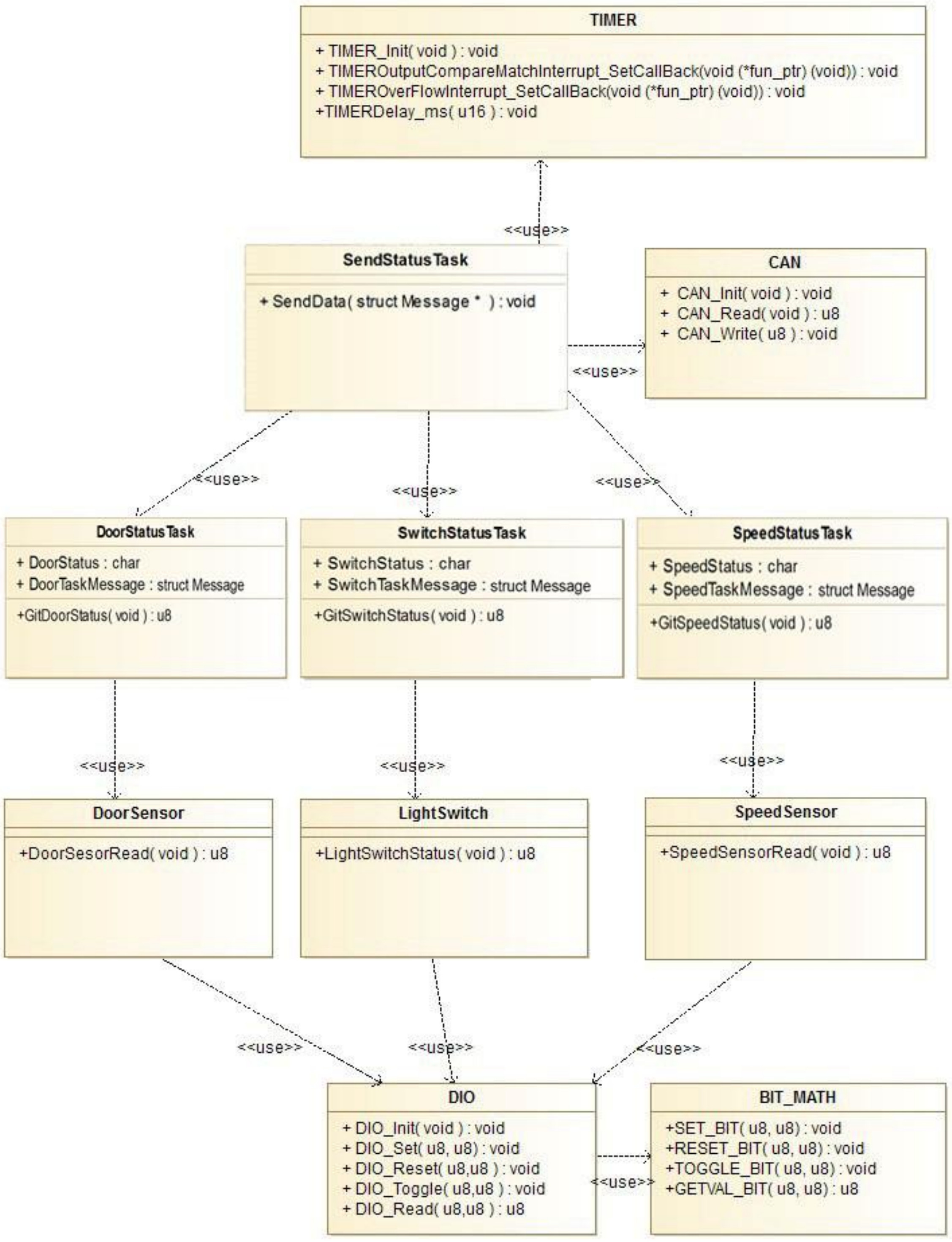
SendData

Description: This is a function use to send the reads of car sensors through CAN bus

Arguments: struct Message * which is a pointer point to the address of the struct Message

Return: void

ECU1 Class diagram


TIMER
+ TIMER_Init(void) : void
+ TIMEROutputCompareMatchInterrupt_SetCallBack(void (*fun_ptr) (void)) : void
+ TIMEROverFlowInterrupt_SetCallBack(void (*fun_ptr) (void)) : void
+TIMERDelay_ms(u16) : void
<<use>>
SendStatusTask
+ SendData(struct Message *) : void
<<use>>
CAN
+ CAN_Init(void) : void
+ CAN_Read(void) : u8
+ CAN_Write(u8) : void
<<use>>
<<use>>
<<use>>
DoorStatusTask
+ DoorStatus : char
+ DoorTaskMessage : struct Message
+GitDoorStatus(void) : u8
SwitchStatusTask
+ SwitchStatus : char
+ SwitchTaskMessage : struct Message
+GitSwitchStatus(void) : u8
SpeedStatusTask
+ SpeedStatus : char
+ SpeedTaskMessage : struct Message
+GitSpeedStatus(void) : u8
<<use>>
<<use>>
<<use>>
DoorSensor
+DoorSesorRead(void) : u8
LightSwitch
+LightSwitchStatus(void) : u8
SpeedSensor
+SpeedSensorRead(void) : u8
<<use>>
<<use>>
<<use>>
DIO
+ DIO_Init(void) : void
+ DIO_Set(u8, u8) : void
+ DIO_Reset(u8,u8) : void
+ DIO_Toggle(u8,u8) : void
+ DIO_Read(u8,u8) : u8
<<use>>
BIT_MATH
+SET_BIT(u8, u8) : void
+RESET_BIT(u8, u8) : void
+TOGGLE_BIT(u8, u8) : void
+GETVAL_BIT(u8, u8) : u8