Diagram

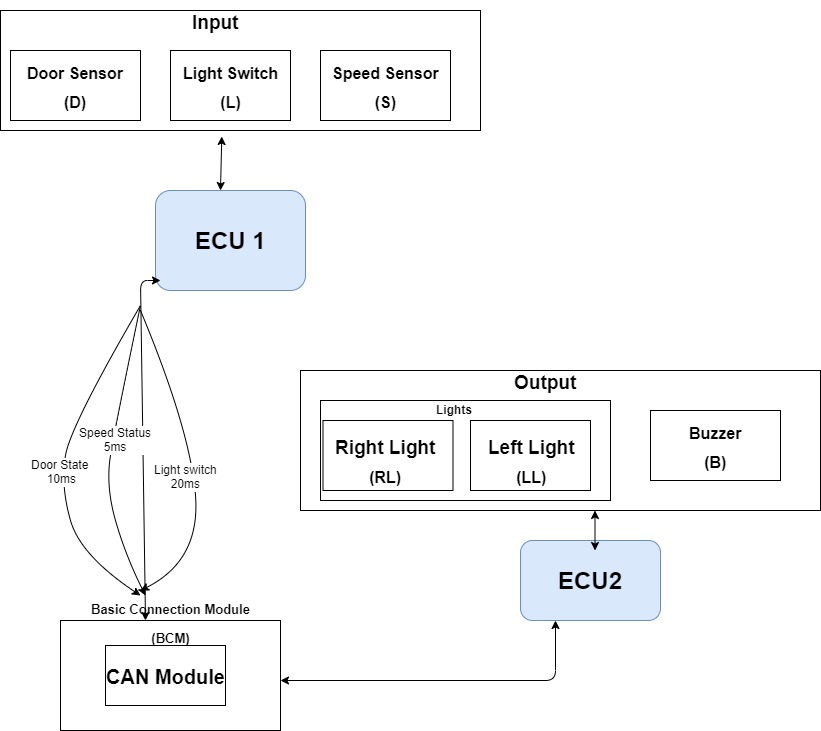
Description automatically generated

Car Door Design

Static and dynamic design

ziad | EGFWD | 9/6/2022

# Requirement block diagram



## Static Design

* ECU 1 Layered Architecture Design

Diagram

Description automatically generated

* APP Layer
* Basic Communication Module (BCM)

APIs :

* void BCM\_initCAN()

Description : initialize CAN module and set its configuration according to CAN config file

* u8 BCM\_u8IsReady()

Description : return 1 if can ready else zero

/\*gets speed from SMM and save it in the struct\*/

* struct SpeedStatusMsg \* BCM\_setSpeedStatusMsg()

Description : gets speed from Speed Monitor Modeule and save it in the struct of type speed status

* struct DoorStateMsg \* BCM\_setDoorStateMsg()

Description: gets door status from Door and Light Monitor Module and save it in the struct of type DoorStateMsg

* struct LightSwitchMsg \* BCM\_setLightSwitchMsg ()

Description : gets lights switch state from Door and Light Monitor Module and save it in the struct of type LightSwitchMsg

* void BCM\_updateCANSpeedStatus()

Description: Send speed on CAN bus

* void BCM\_updateCANDoorStatus()

Description: Send Door state on CAN bus

* void BCM\_updateCANLightSwitchStatus()
* Description: Send Light Switch State on CAN bus

Typedefs: - struct SpeedStatusMsg\_t

- struct DoorStateMsg\_t

- struct LightSwitchMsg\_t

* Speed Monitor Module

APIs :

* void SMM\_initSpeedSensor()

Description: Init SpeedSensor according to Speed sensor driver config file

* SENSOR\_READING\_t SMM\_getSensorReading ()

Description: return sensor reading in a 4 byte variable

Typedefs: - uint32 SENSOR\_READING\_t

* Light Switch Monitor Module

APIs :

* void LSMM\_initSwitchMonitor()

Description: Init Light switch sensor according to light switch sensor driver config file

* SWITCH\_STATE\_t LSMM\_getSwitchState()

Description: return switch state either 0 or 1

Typedefs: - enum SWITCH\_STATE\_t /\*0 or 1 \*/

* Door Monitor Module

APIs :

* void DMM\_initDoorMonitor()

Description: Init door sensor according to Door sensor driver config file

* void DMM\_getDoorState()

returns either 0 or 1

Typedefs: - Enum DOOR\_STATE\_t /\*0 or 1\*/

* OS Layer

APIs :

* void vTask\_periodicSendSpeedState()

Description : calls BCM update speed function to send speed reading periodically on CAN bus

* void vTask\_periodicSendSwitchState()

Description : calls BCM update switch state function to send switch state periodically on CAN bus

* void vTask\_periodicSendDoorState()

Description : calls BCM update door state function to send door state periodically on CAN bus

* void StartSchedular()

Description: enter super loop and start OS scheduling algorithm

* HAL Layer
* Speed Sensor Driver

APIs : -

* void SpeedSensor\_init(

SPEED\_SENSOR\_CHANNEL\_t )

Description: init speed senor with the configyrblr pins

Argument :

-SPEED\_SENSOR\_CHANNEL\_t : enum ,Input , Range from 0 to 10 , decription:- specify the ADC cahnnel connected to the sensor

* u32 SpeedSensor\_getSensorReading ()

Description: return the speed from the ADC

* CAR\_STATE\_t SpeedSensor\_isCarMoving()

Description: returns either one or zero to indicate if the car is moving

Typedefs: - enum SMM\_SENSOR\_CHANNEL\_t/\*from 0 to 10\*/

- enum SMM\_CAR\_STATE\_t /\*zero or one \*/

* Light Switch Driver

APIs :

* void LSD\_initSwitchMonitor(

LSD\_CHANNEL\_t,

LSD\_PORT\_t )

Description: init Switch with the configurble pin that it is connected to

Argument :

- LSD\_CHANNEL\_t,: enum ,Input , Range from 0 to 7 , decription:- specify the DIO Pin connected to the sensor

- LSD\_PORT\_t : enum ,Input , Range from 0 to 4 , decription:- specify the DIO Port connected to the sensor

* LIGHT\_SWITCH\_STATE\_t LSD\_isSwitchClosed()

Description returns either 0 or 1 to indicate if switch is closed

Typedefs: - enum LSD\_CHANNEL\_t/\*from 0 to 7\*/

- enum LSD\_PORT\_t /\*from 0 to 3\*/

- enum LSD\_LIGHT\_SWITCH\_STATE\_t /\*zero or one \*/

Door Lock Driver

APIs : -

* void DLD\_initLockMonitor(

DLD\_CHANNEL\_t,

DLD\_PORT\_t )

Description: init Lock sensor with the configurble pin that it is connected to

Argument :

- DLD\_CHANNEL\_t, enum ,Input , Range from 0 to 7 , decription:- specify the DIO Pin connected to the sensor

- DLD\_PORT\_t: enum ,Input , Range from 0 to 4 , decription:- specify the DIO Port connected to the sensor

* DOOR\_LOCK\_STATE\_t DLD\_isDoorLocked()

Description returns either 0 or 1 to indicate if Door is locked

Typedefs: - enum DLD\_CHANNEL\_t, /\*from 0 to 7\*/

- enum DLD\_PORT\_t/\*from 0 to 3\*/

- enum DOOR\_LOCK\_STATE\_t /\*zero or one\*/

* MCAL Layer
* DIO

APIs :

* void DIO\_vidInit (PIN\_INFO\_t)

Description: init DIO with the configuration for every pin

Argument :

- PIN\_INFO\_t, Array of struct ,Input , Range is defined by a mcro specify the numer of pins used

decription:- specify the DIO Pin configuration like direction Mode Level , PullUP/down etc…

* Void DIO\_vidWritePin(PORT , PIN,LEVEL)

Description: write on the pin the rquired value

Argument :

-PORT: enum, input , range from 0 to 4 , deccription: specify the port

- PIN: enum, input , range from 0 to 7 , deccription: specify the pin

-LEVEL: enum, input , range from 0 to 1 , deccription: specify the logical level

* LEVEL DIO\_vidReadPin(PORT , PIN)

Description: reads the value of the rquired pin, returns either 0 or 1

Argument :

-PORT: enum, input , range from 0 to 4 , deccription: specify the port

- PIN: enum, input , range from 0 to 7 , deccription: specify the pin

Typedefs: - struct PIN\_INFO\_t/\*PORT,PIN ,MODE , LEVEL\*/

- enum PORT/\*0 to 3\*/

- enum PIN/\*0 to 7\*/

- enum LEVEL/\*one or zero\*/

-enum MODE /\*one or zero\*/

* ADC

APIs :

* void ADC\_vidInit (CHANNEL, MODE)

description: init ADC with a specific channel and mode of operation

arguments:

* CHANNEL: enum ,input, range from 0 to 10 ,
* MODE: enum , input , range from 0 to 3 , specify one of the 4 modes of the ADC
* U32 ADC\_u32StartConversion()

Description : starts the conversion of ADC

* void ADC\_vidEnableInt(CHANNEL)

Description : enable the interrupt of ADC to rad value from ISR

Typedefs: - enum CHANNEL/\*0 to 9\*/

- enum MODE/\*0 to 3\*/

* TIMERS

APIs :

* void TIMER\_init ( CHANNEL\_INFO\_t \* )

description: init timer with the used channels and their modes of operation

arguments:

* CHANNEL\_INFO\_t: Array of struct , range is specified by a Macro , input , description contains the configuration of every timer channel /\*timer mode timer Interrupt state continuous or one shot \*/
* Void TIMER\_vidStart(u32 ticks)

Description : start timer and put ticks in the counter register

Argument :

* Ticks : 4bytes variable , input , range from 0 to 232 ,specify the numer of ticks to be counted by the timer
* void TIMER\_vidStop()

Description : stop timer

* u32 TIMER\_u32GetTimeElapsed()

Description : returns the number of ticks counted till now

* u32 TIMER\_u32GetTimeRemaining()

Description : returns the number of ticks remaining till counter finishes

Typedefs: -struct CHANNEL\_INFO\_t/\*Timer Mode , Timer Int sate , continuous or one shot\*/

* CAN

APIs :

* void CAN\_init ( void )

Description : init CAN

* void Can\_SetBaudrate(u16 baudRate)

Description: set the buad rate

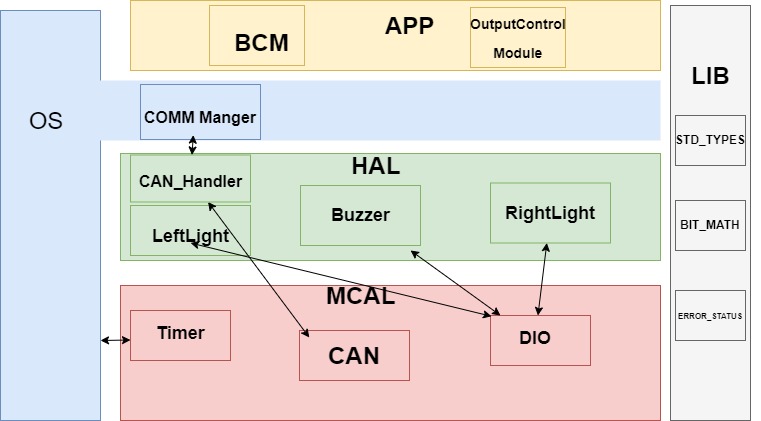
Arguments: two bytes variable , Range: 0 to 216 Description: the new buad rate buadrate

* void CAN\_Write(u16 data)

Description: write on CAN bus the given data

Arguments: data : two bytes variable , Range: 0 to 216 Description: the data to be written on Can bus

* ECU 2 Layered Architecture Design



* APP Layer
* Basic Communication Module (BCM)

APIs :

• void BCM\_initCAN()

Description : initialize CAN module and set its configuration according to CAN config file

•u8 BCM\_u8IsReady()

Description : return 1 if can ready else zero

•struct SpeedStatusMsg \* BCM\_GETSpeedStatusMsg()

Description : gets speed from CAN bus and pass it to Output Control module by the struct pointer

•struct DoorStateMsg \* BCM\_setDoorStateMsg()

Description : gets Door state from CAN bus and pass it to Output Control module by the struct pointer

•struct LightSwitchMsg \* BCM\_setLightSwitchMsg ()

Description : gets switch state from CAN bus and pass it to Output Control module by the struct pointer

Typedefs: - struct SpeedStatusMsg\_t

- struct DoorStateMsg\_t

- struct LightSwitchMsg\_t

* Output Control Module

APIs :

* void OCM\_init()

description: init all hard ware modules

* void OCM\_update()

description called periodically from OS to perform the control logic according to the new states /\*perform the state machine diagram\*/

* void OCM\_enableLight()

description : enable lights by calling Light driver

* void OCM\_disableLight()

description : disable lights by calling Light driver

* void OCM\_asynchLightOff()

description: disable light after 3 seconds

* void OCM\_enableBuzzer()

description : enable buzzer by calling buzzer driver

* void OCM\_disableBuzzer()
* description : disable buzzer by calling buzzer driver

* OS Layer

APIs :

* void vTask\_periodicSendSpeedState()

Description : calls BCM get speed function to read speed reading periodically from CAN bus

* void vTask\_periodicSendSwitchState()

Description : calls BCM get switch state function to read switch state periodically from CAN bus

* void vTask\_periodicSendDoorState()

Description : calls BCM get door state function to read door state periodically from CAN bus

* void vTask\_peridocOutputLogicControl()

description calls update function from Output control module to perform state machine

* void StartSchedular()

Description: enter super loop and start OS scheduling algorithm

* HAL Layer
* Light Driver

APIs :

* void LD\_LeftLightInit(

LD\_CHANNEL\_t,

LD\_PORT\_t)

Description: init left light with the configurble pin that it is connected to

Argument :

- LD\_CHANNEL\_t,: enum ,Input , Range from 0 to 7 , decription:- specify the DIO Pin connected to the sensor

- LD\_PORT\_t : enum ,Input , Range from 0 to 4 , decription:- specify the DIO Port connected to the sensor

* -void LD\_RightLightInit(

LD\_CHANNEL\_t,

LD\_PORT\_t)

Description: init right light with the configurble pin that it is connected to

Argument :

- LD\_CHANNEL\_t,: enum ,Input , Range from 0 to 7 , decription:- specify the DIO Pin connected to the sensor

- LD\_PORT\_t : enum ,Input , Range from 0 to 4 , decription:- specify the DIO Port connected to the sensor

* void LD\_enableLiftlight()

description: enable left light by calling write pin func in DIO

* void LD\_disnableLiftlight()

description: disable left light by calling write pin func in DIO

* void LD\_enableRightlight()

description: enable right light by calling write pin func in DIO

* void LD\_disableRightlight()
* description: disable right light by calling write pin func in DIO

Typedefs: - enum LD\_CHANNEL\_t/\*from 0 to 7\*/

- enum LD\_PORT\_t/\*from 0 to 3\*/

- enum SMM\_CAR\_STATE\_t/\*zero or one\*/

* Buzzer Driver

APIs :

* void BD\_initBuzzer( BD\_CHANNEL\_t,

BD\_PORT\_t )

Description: init buzzer with the configurble pin that it is connected to

Argument :

-BD\_CHANNEL\_t,: enum ,Input , Range from 0 to 7 , decription:- specify the DIO Pin connected to the sensor

-BD\_PORT\_t: enum ,Input , Range from 0 to 4 , decription:- specify the DIO Port connected to the sensor

* Void BD\_enableBuzzer()

Description : enable buzzer by calling DIO

* Void BD\_disableBuzzer

Description : disable buzzer by calling DIO

Typedefs: - enum BD\_CHANNEL\_t/\*from 0 to 7\*/

- enum BD\_PORT\_t/\*from 0 to 3\*/

* MCAL Layer
* DIO

APIs :

* void DIO\_vidInit (PIN\_INFO\_t)

Description: init DIO with the configuration for every pin

Argument :

- PIN\_INFO\_t, Array of struct ,Input , Range is defined by a mcro specify the numer of pins used

decription:- specify the DIO Pin configuration like direction Mode Level , PullUP/down etc…

* Void DIO\_vidWritePin(PORT , PIN,LEVEL)

Description: write on the pin the rquired value

Argument :

-PORT: enum, input , range from 0 to 4 , deccription: specify the port

- PIN: enum, input , range from 0 to 7 , deccription: specify the pin

-LEVEL: enum, input , range from 0 to 1 , deccription: specify the logical level

* LEVEL DIO\_vidReadPin(PORT , PIN)

Description: reads the value of the rquired pin, returns either 0 or 1

Argument :

-PORT: enum, input , range from 0 to 4 , deccription: specify the port

- PIN: enum, input , range from 0 to 7 , deccription: specify the pin

Typedefs: - struct PIN\_INFO\_t/\*PORT,PIN ,MODE , LEVEL\*/

- enum PORT/\*0 to 3\*/

- enum PIN/\*0 to 7\*/

- enum LEVEL/\*one or zero\*/

-enum MODE /\*one or zero\*/

* TIMERS

APIs :

* void TIMER\_init ( CHANNEL\_INFO\_t \* )

description: init timer with the used channels and their modes of operation

arguments:

* CHANNEL\_INFO\_t: Array of struct , range is specified by a Macro , input , description contains the configuration of every timer channel /\*timer mode timer Interrupt state continuous or one shot \*/
* Void TIMER\_vidStart(u32 ticks)

Description : start timer and put ticks in the counter register

Argument :

* Ticks : 4bytes variable , input , range from 0 to 232 ,specify the numer of ticks to be counted by the timer
* void TIMER\_vidStop()

Description : stop timer

* u32 TIMER\_u32GetTimeElapsed()

Description : returns the number of ticks counted till now

* u32 TIMER\_u32GetTimeRemaining()

Description : returns the number of ticks remaining till counter finishes

Typedefs: -struct CHANNEL\_INFO\_t/\*Timer Mode , Timer Int sate , continuous or one shot\*/

* CAN

APIs :

* void CAN\_init ( void )

Description : init CAN

* void Can\_SetBaudrate(u16 baudRate)

Description: set the buad rate

Arguments: two bytes variable , Range: 0 to 216 Description: the new buad rate buadrate

* u16 CAN\_Read(void)

Description: read from CAN bus the data , returns a 2 bytes variable

## Dynamic Design

* ECU 1

State Machine Diagram

Diagram

Description automatically generated

Sequence Diagram

Diagram, schematic

Description automatically generated

* ECU 2

State Machine Diagram

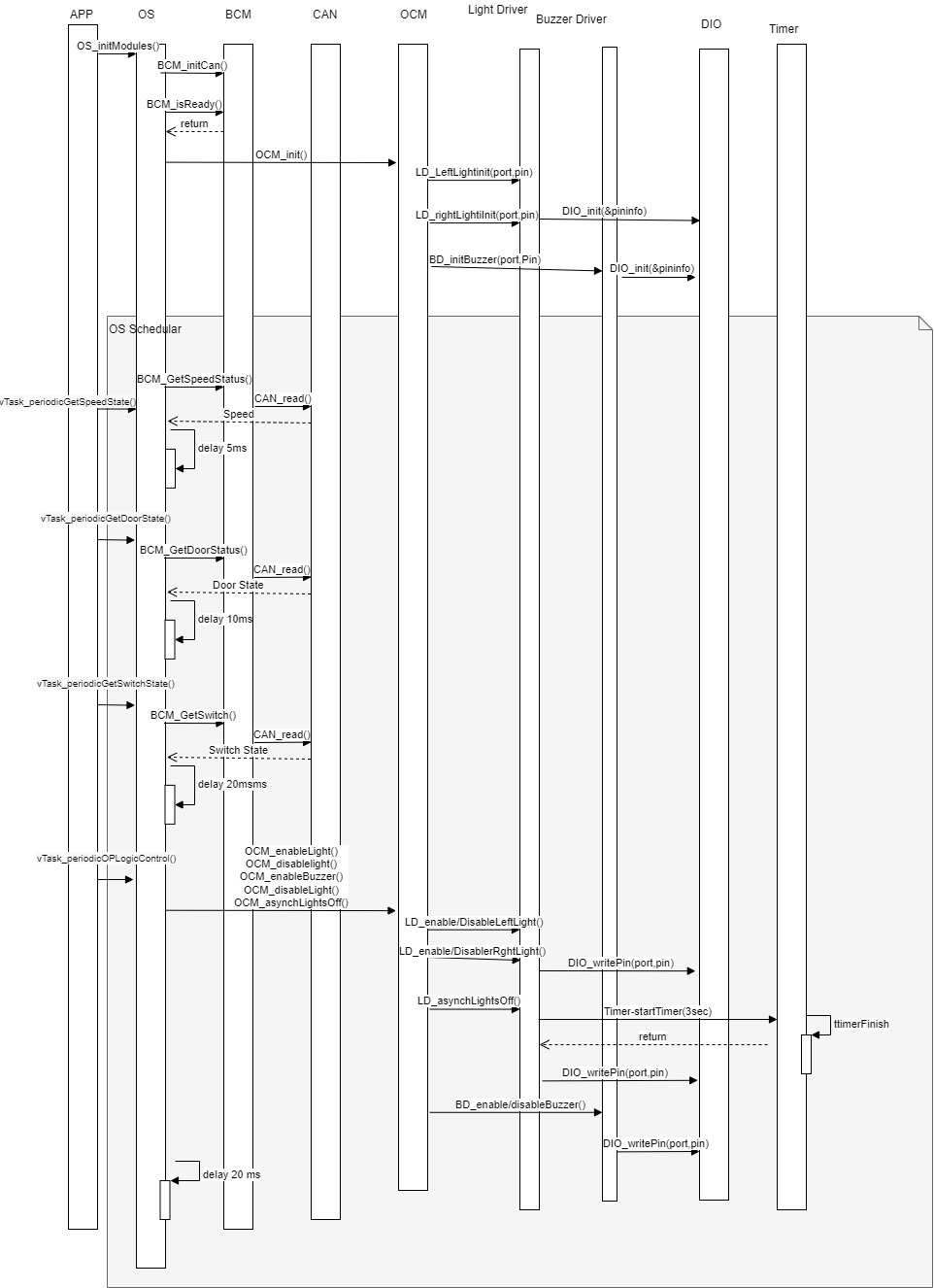
Diagram

Description automatically generated

Diagram

Description automatically generated

Sequence Diagram



File Structure digram 