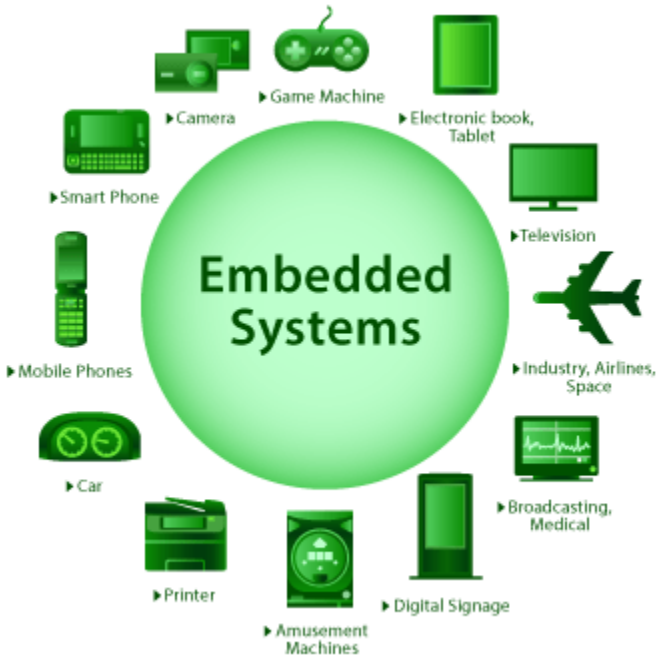
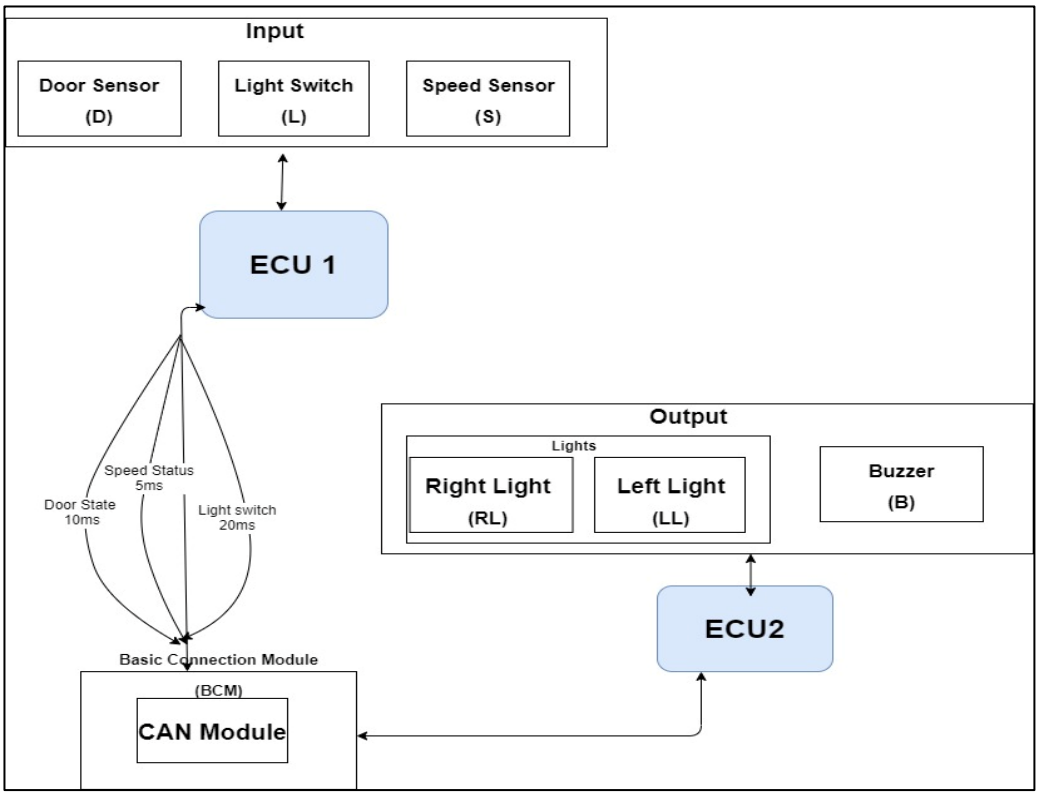


Car Door Design

STATIC AND DYNAMIC DESIGN

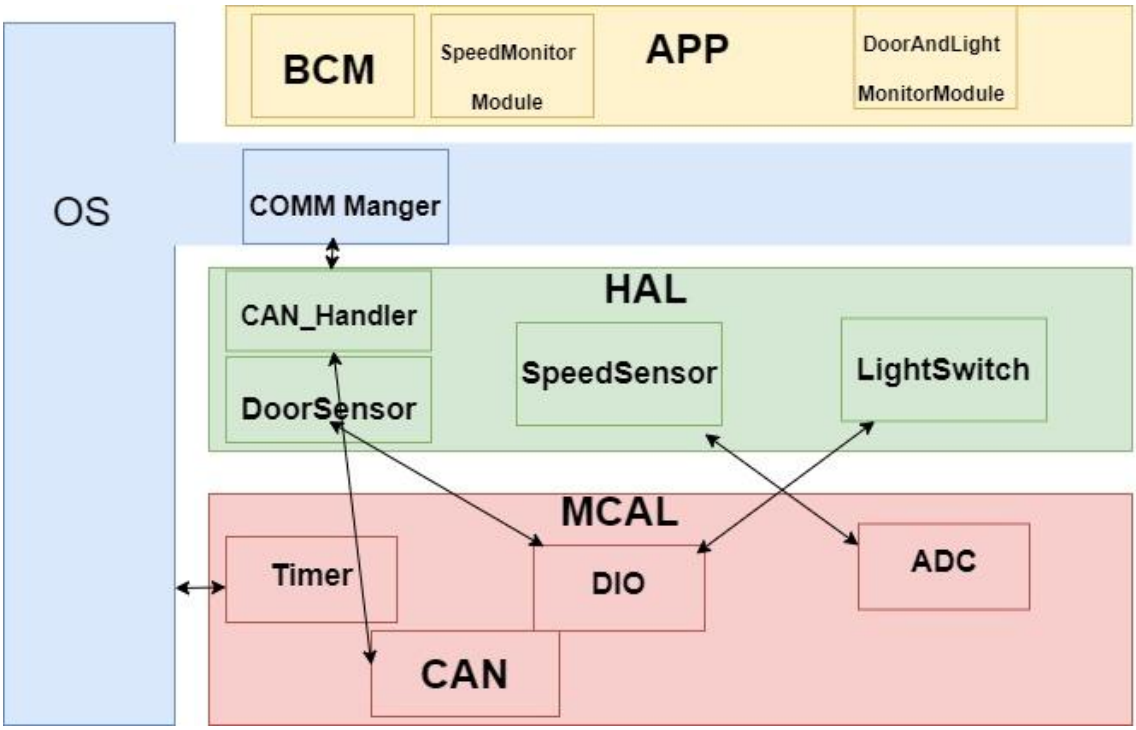
ziad | EGFWD | 9/6/2022

Requirement block diagram



STATIC DESIGN

ECU 1 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

/*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 2^{32} ,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 :

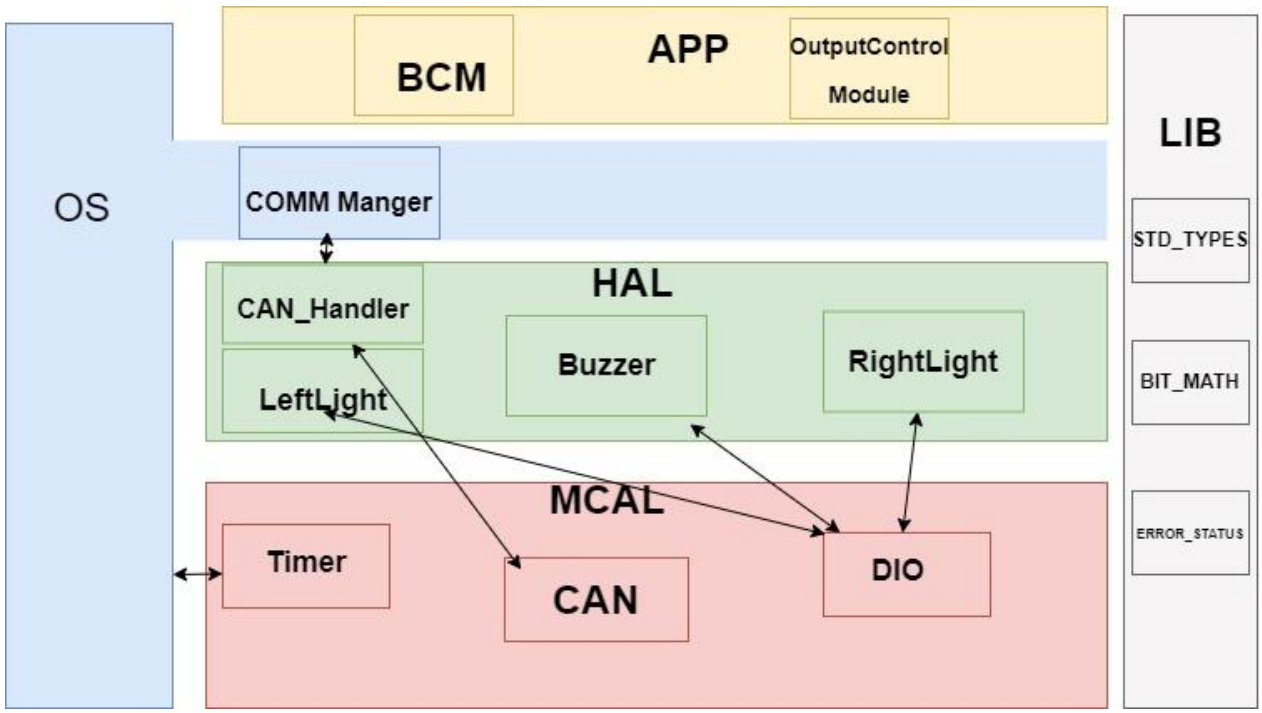
- void CAN_init (void)
  Description : init CAN

- void Can_SetBaudrate(u16 baudRate)
  Description: set the buad rate
  Arguments: two bytes variable , Range: 0 to 2^{16} 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 2^{16} 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 2^{32} ,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

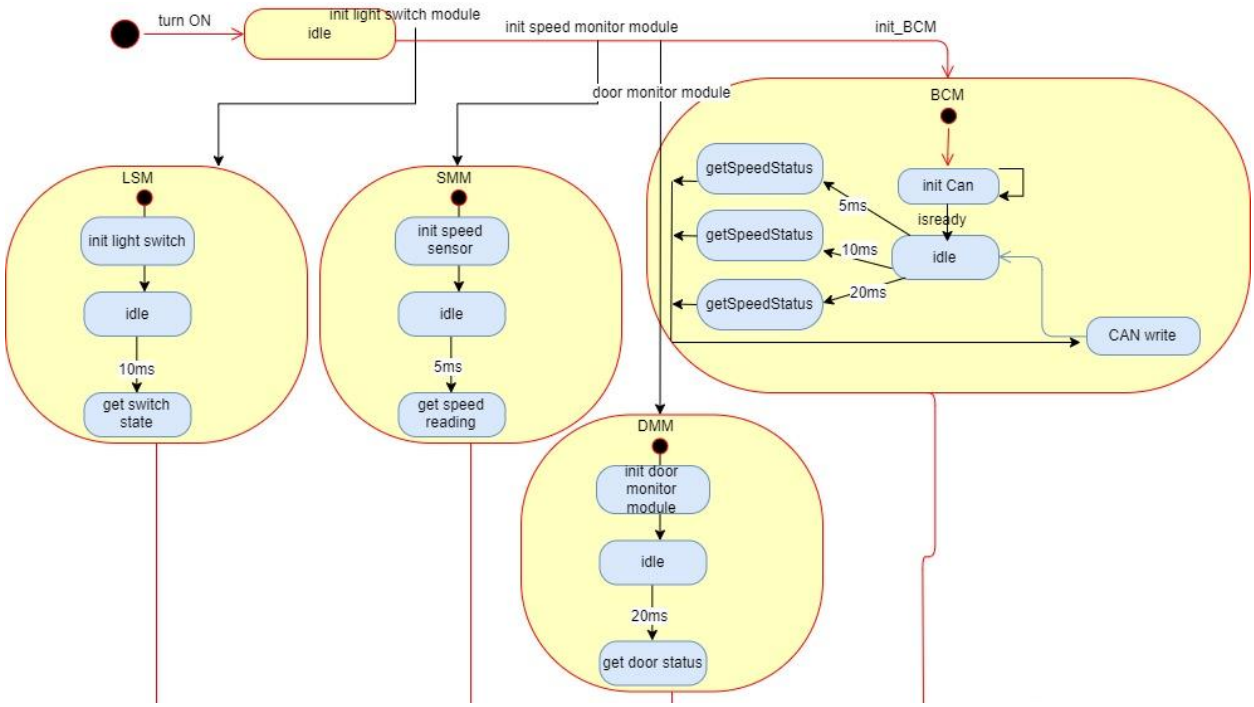
Arguments: two bytes variable , Range: 0 to 2^{16} Description: the new buad rate buadrate

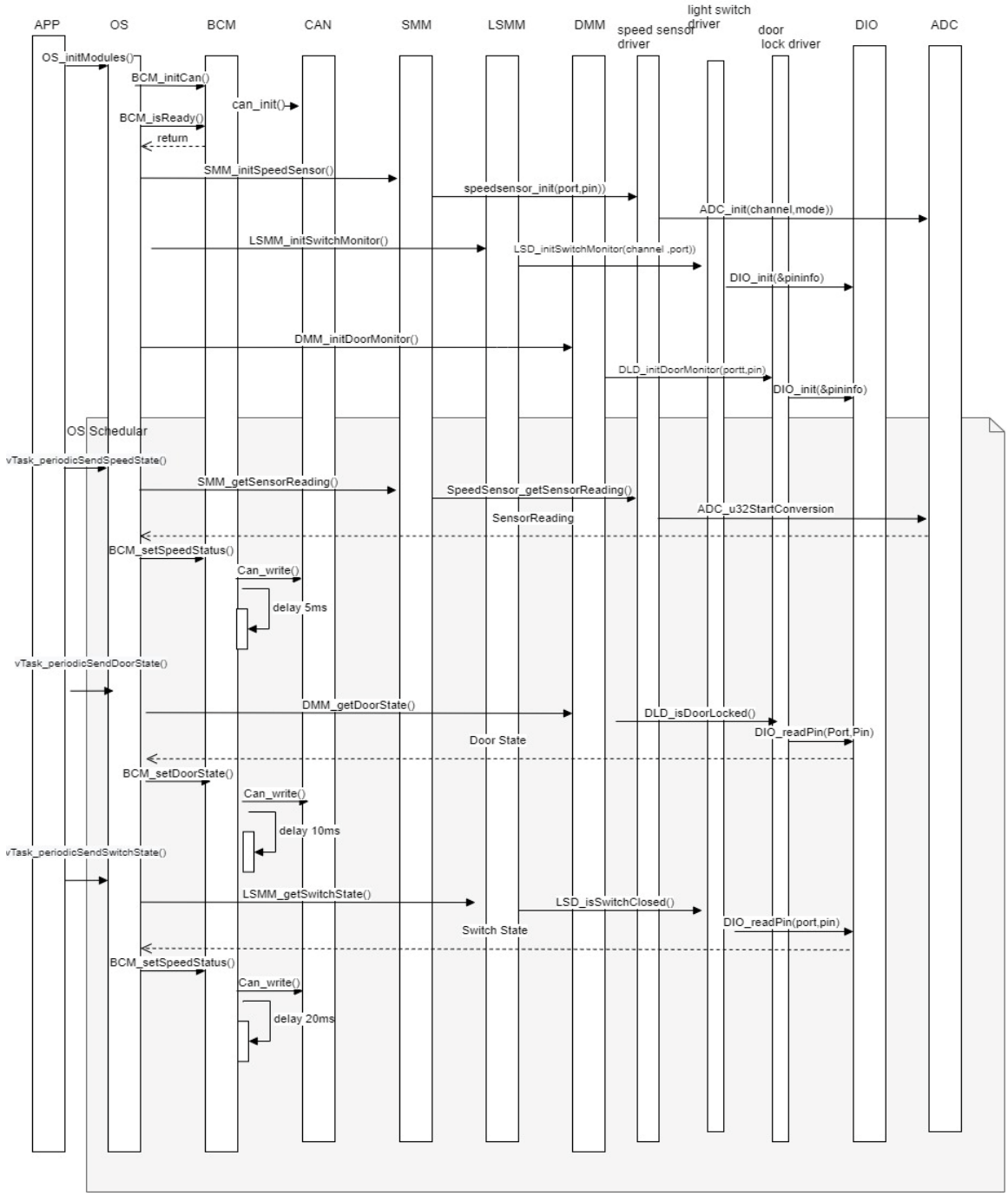
- u16 CAN_Read(void)
  Description: read from CAN bus the data , returns a 2 bytes variable

DYNAMIC DESIGN

ECU 1

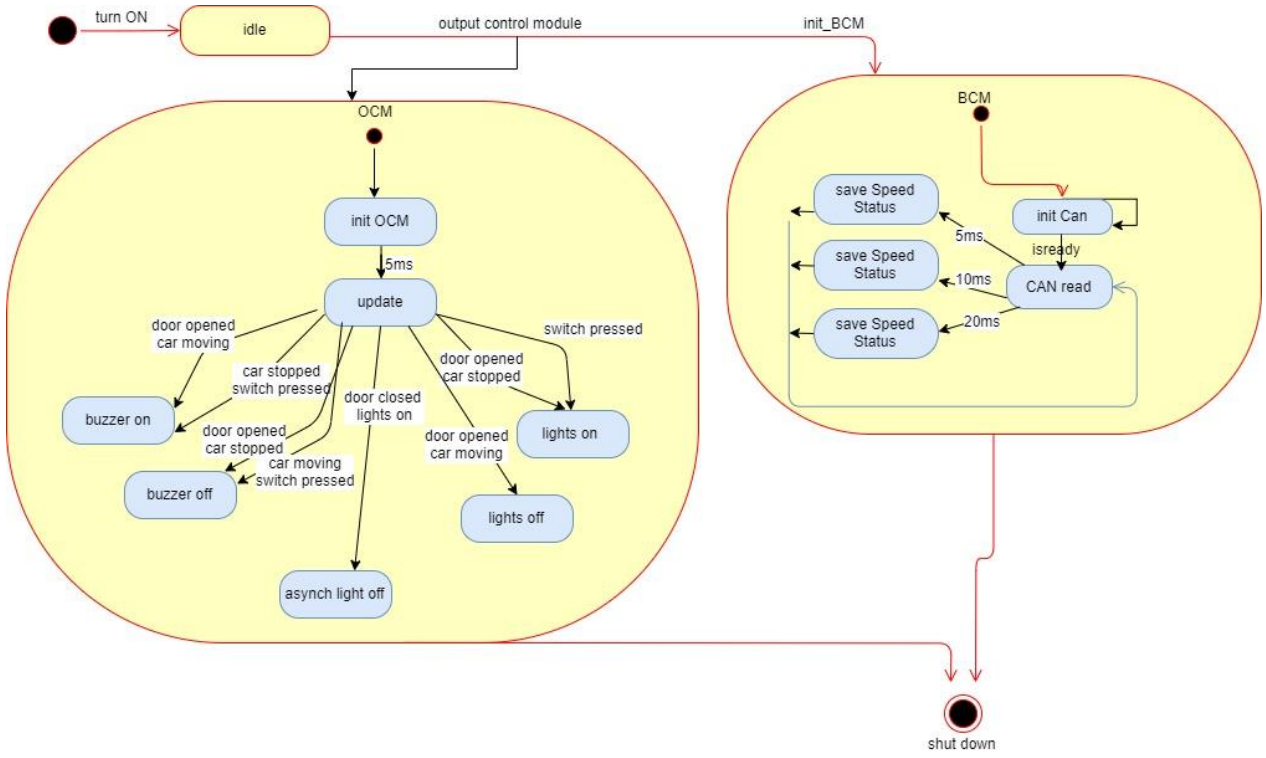
State Machine Diagram



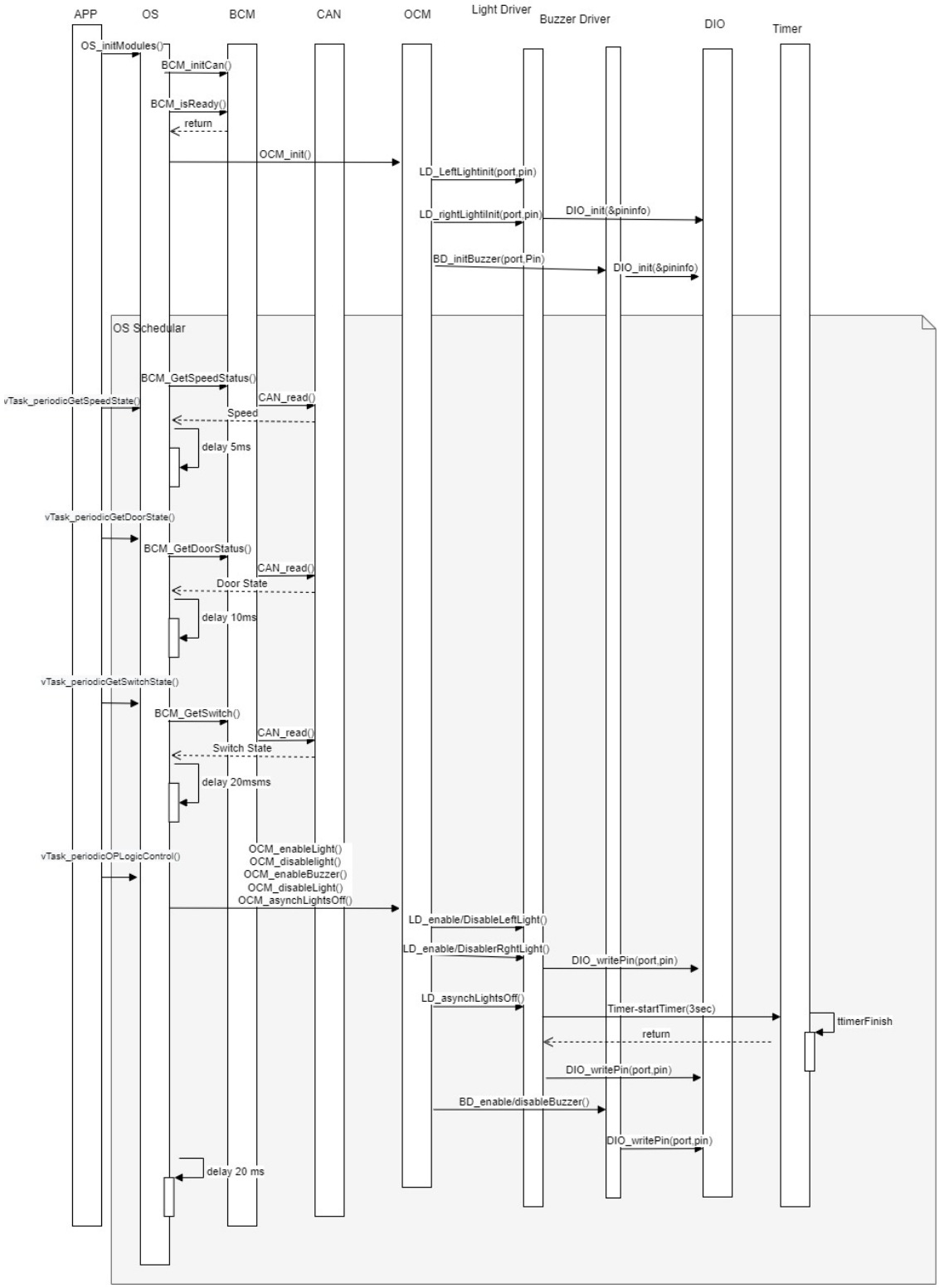

APP
OS
BCM
CAN
SMM
LSMM
DMM
speed sensor driver
light switch driver
door lock driver
DIO
ADC
OS_initModules()
BCM_initCan()
can_init()
BCM_isReady()
return
SMM_initSpeedSensor()
speedsensor_init(port,pin))
ADC_init(channel,mode))
LSMM_initSwitchMonitor()
LSD_initSwitchMonitor(channel ,port))
DIO_init(&pininfo)
DMM_initDoorMonitor()
DLD_initDoorMonitor(portt,pin)
DIO_init(&pininfo)
OS Schedular
vTask_periodicSendSpeedState()
SMM_getSensorReading()
SpeedSensor_getSensorReading()
ADC_u32StartConversion
SensorReading
BCM_setSpeedStatus()
Can_write()
delay 5ms
vTask_periodicSendDoorState()
DMM_getDoorState()
DLD_isDoorLocked()
DIO_readPin(Port,Pin)
Door State
BCM_setDoorState()
Can_write()
delay 10ms
vTask_periodicSendSwitchState()
LSMM_getSwitchState()
LSD_isSwitchClosed()
DIO_readPin(port,pin)
Switch State
BCM_setSpeedStatus()
Can_write()
delay 20ms

ECU 2

State Machine Diagram



Sequence Diagram



File Structure digram