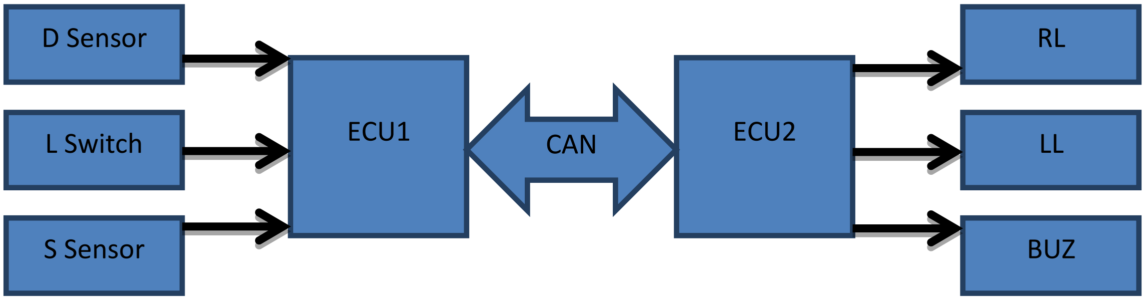
Project #3

Embedded Software Design

1. Fully Static Design.

- System Hardware Requirements



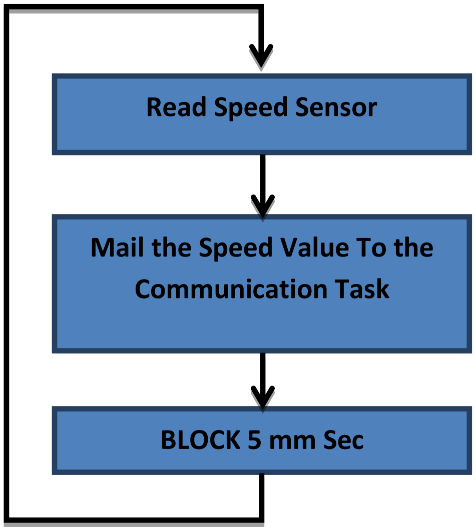
BLOCK Diagram

- System Software Requirements:

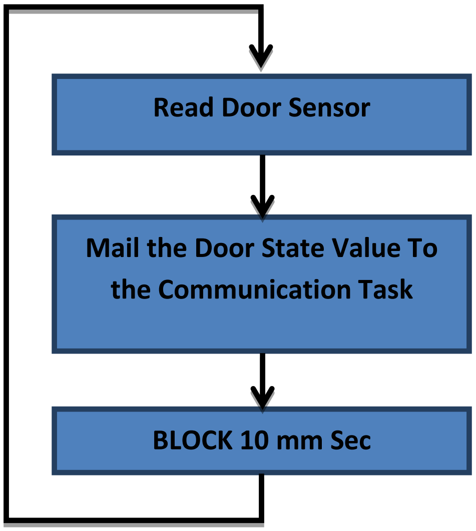
1. ECU1

- ECU1 will designed to have a Real-Time Operating System (RTOS) that read the provided switch &sensors State Values and send them to Specific ECU periodically via CAN Bus
- EDF Scheduler Will be Implemented
- ECU1 has 4 Tasks and the Idle Task
    1. SPEED Task [5 mmsec]
    2. Switch Task [20 mmsec]
    3. Door Task [10 mmsec]
    4. Communication Task [5 mm sec]
- Task Flowchart

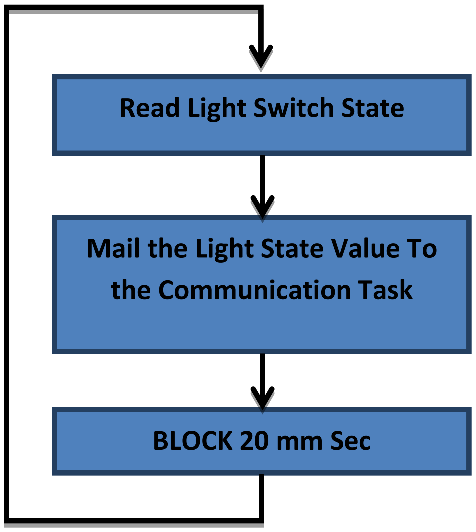
SPEED Task



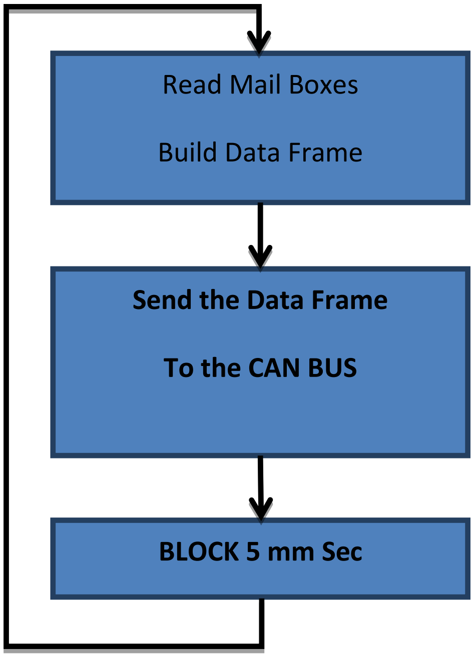
Door Task



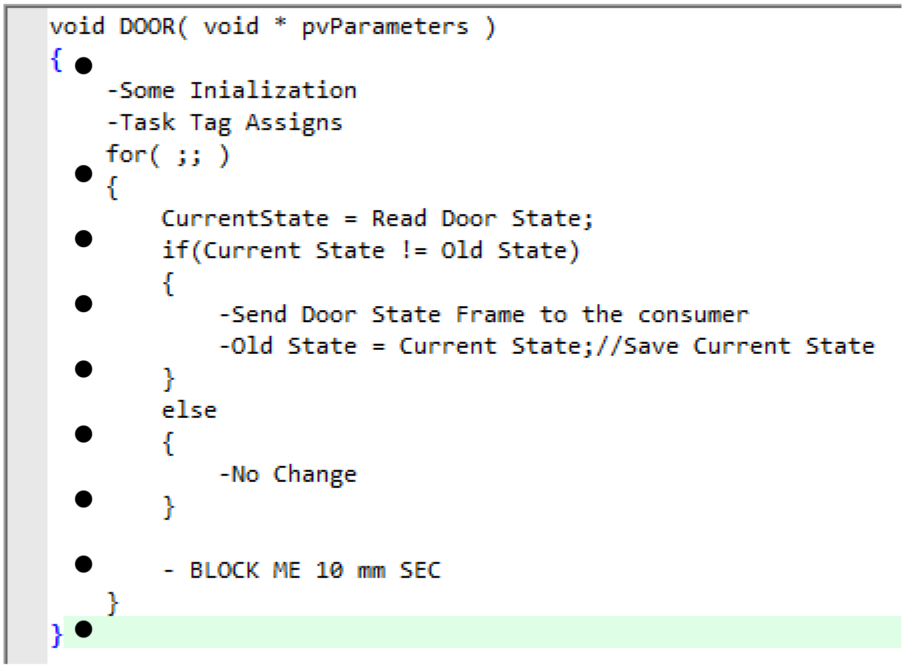
Switch Task



Communication Task

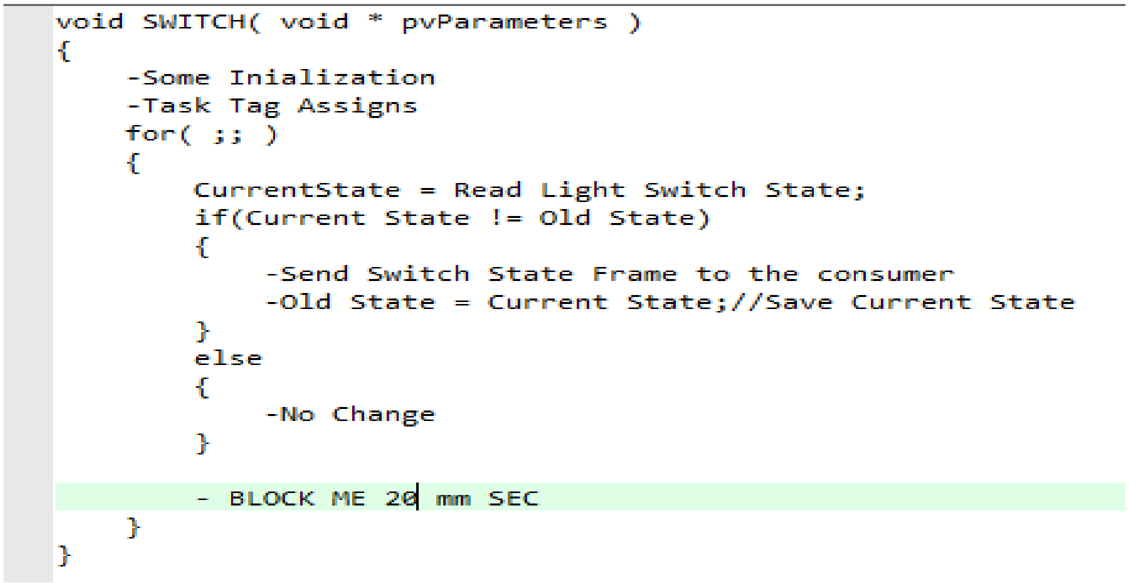


- ECU1 pseudo code
  - DOOR Task pseudo code



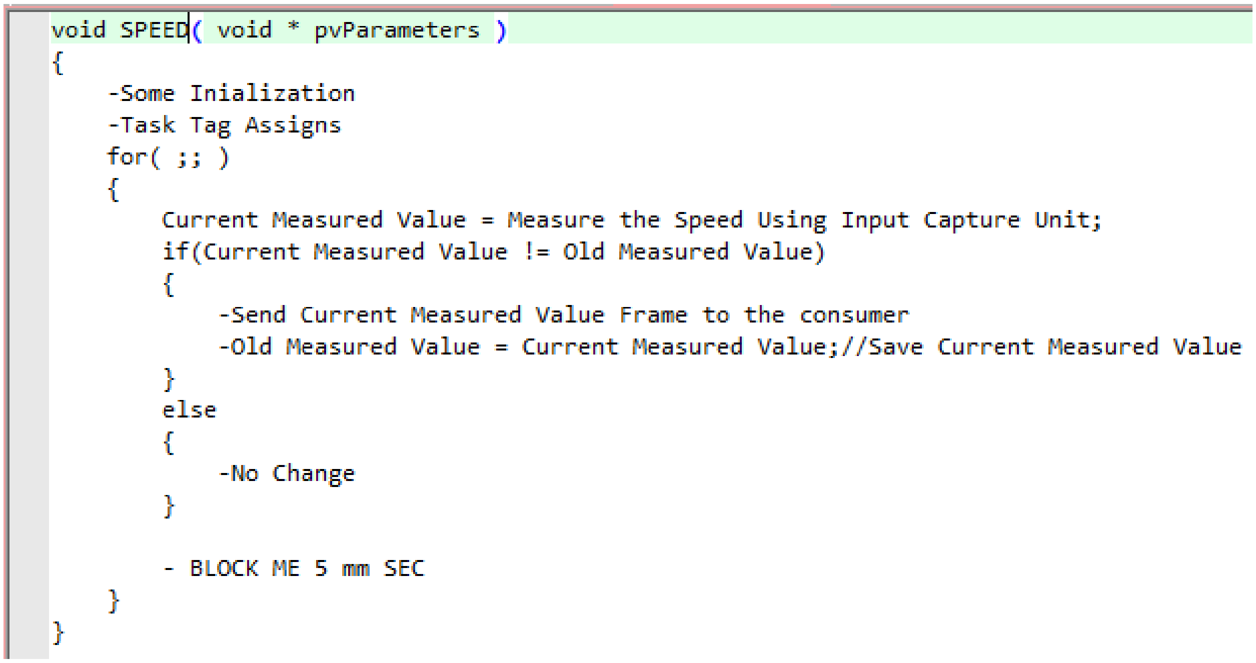
```
void DOOR( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        CurrentState = Read Door State;
        if(Current State != Old State)
        {
            -Send Door State Frame to the consumer
            -Old State = Current State;//Save Current State
        }
        else
        {
            -No Change
        }

        - BLOCK ME 10 mm SEC
    }
}
```

- Light Switch Task pseudo code



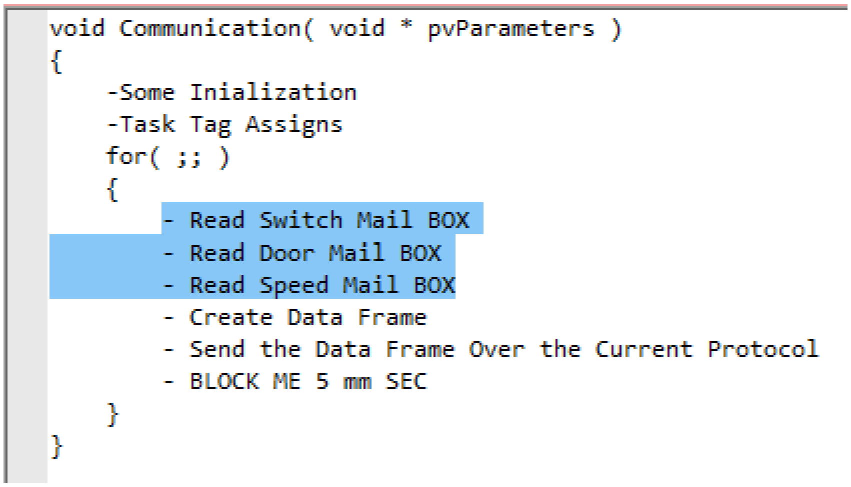
```
void SWITCH( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        CurrentState = Read Light Switch State;
        if(Current State != Old State)
        {
            -Send Switch State Frame to the consumer
            -Old State = Current State;//Save Current State
        }
        else
        {
            -No Change
        }

        - BLOCK ME 20 mm SEC
    }
}
```

- Speed Measure Task pseudo code



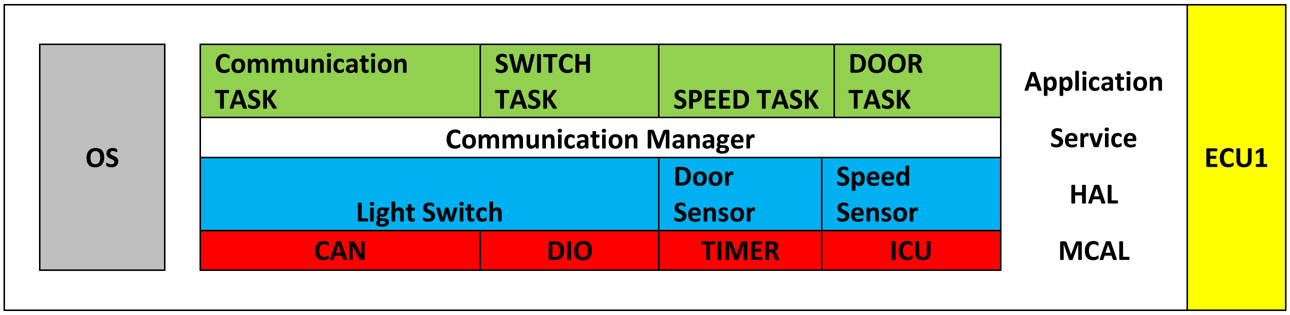
```
void SPEED( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        Current Measured Value = Measure the Speed Using Input Capture Unit;
        if(Current Measured Value != Old Measured Value)
        {
            -Send Current Measured Value Frame to the consumer
            -Old Measured Value = Current Measured Value;//Save Current Measured Value
        }
        else
        {
            -No Change
        }

        - BLOCK ME 5 mm SEC
    }
}
```

- Communication Task pseudo code



```
void Communication( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        - Read Switch Mail BOX
        - Read Door Mail BOX
        - Read Speed Mail BOX
        - Create Data Frame
        - Send the Data Frame Over the Current Protocol
        - BLOCK ME 5 mm SEC
    }
}
```

- ECU1 Layered Architecture



- ECU1 Modules APIs Description

Layer	Module	Function Statement	Arguments	Return Description
MCAL	DIO	void DIO_Init(DioConfigPtr_Type *ptr) Function	Struct holds the configurations for GPIO port-pin	void
		void DIO_Write(Pin_Type pin, Port_Type port, Value_Type Value)	Port – Pin - value	void
		value_TypeDIO_Read(Pin_Type pin, Port_Type port)	Port - Pin	Value_Typeenum states pin value (HIGH/LOW)
		void DIO_Toggle(Pin_Type pin, Port_Type port) Function Toggles some GPIO port-pin state	Port - Pin	void
	ICU	void ICU_Init(DioConfigPtr_Type *ptr) Function Initialize some GPIO port-pin as ICU	Struct holds the configurations for some GPIO port-pin	void
		uint16 Capture(TimerType,Pin_Type pin, Port_Type port) Function Reads the Frequency value from some GPIO port-pin Connected To Specific Timer	Timer- Port - Pin	Voltage on pin Decimal value
	CAN	void CAN_Init(DioConfigPtr_Type *ptr) Function Initialize some GPIO port-pin as CAN	Struct holds the configurations for some GPIO port-pin	void
		void CAN_Send(uint32_t *Data) Function send data via CAN Bus	Pointer to the data to be sent	void
		void CAN_Receive(uint32 *Data) Function receive data from CAN Bus	Pointer to store received data in it	void
	TIMER	void Timer_Init(TimerConfigPtr_Type *ptr)	Struct holds the configurations for Timer	void
		void StartTimer(TimerType)	timer	void
		void StopTimer(TimerType)	timer	void
		Void DelayMs(ms)	Delay value in millisecond	void
HAL	DOOR	Void Init_DoorSensor (DoorConfigPtr *ptr) Function initialize some GPIO pin to work with the sensor	Struct holds the configurations for initializing pin to work with the sensor	void
		DoorState_TypeGet_DoorState(DoorConfigPtr *ptr)	Pointer refers to the required door sensor	DoorState_Typeenum with states OPENED/CLOSED

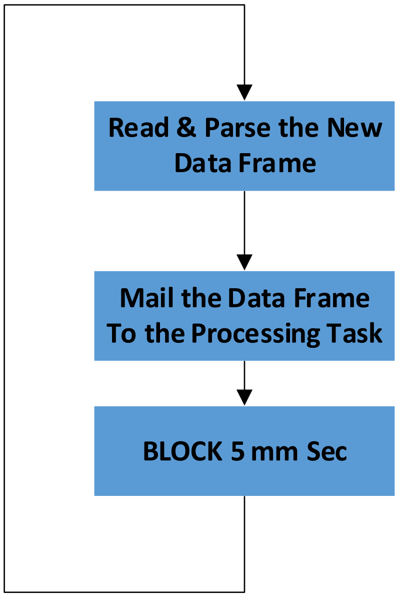
	SPEED	void Init_SpeedSensor (SpeedConfigPtr *ptr)	Struct holds the configurations for initializing ICU pin	void
		uint16 Measure(SpeedConfigPtr *ptr) Function returns some speed sensor Decimal value	Pointer refers to the required speed sensor	Speed Decimal value
	LIGHTS	Void Init_Switch (SwitchConfigPtr *ptr)	Struct holds the configurations for initializing pin	void
		SwitchState_TypeGet_SwitchState(SwitchConfigPtr *ptr)	Pointer refers to the required switch	SwitchState_Typeenum with states Pressed/Released
Services	COMMUNICATION	Void Comm(u8 ID, u32 *Data)	ID : represents the required Comm protocol to send via Data : Pointer to data to be sent	void
Application	OS	SPEED Task - Switch Task - Door Task -Communication Task	4 Tasks	void

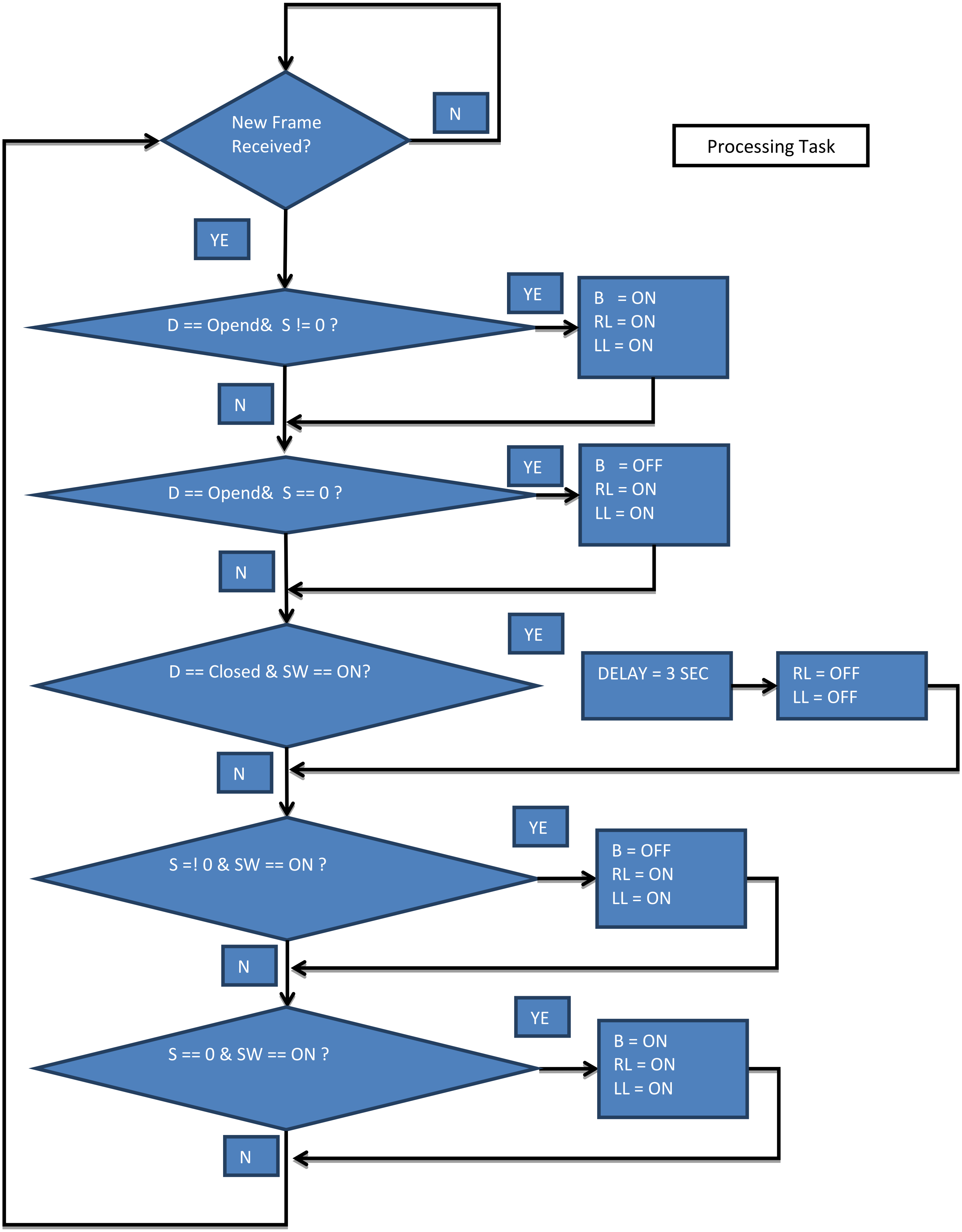
2. ECU2

- ECU2 will designed to have a Real-Time Operating System (RTOS) thatReceive the provided Speed , switch & doorState Values and Perform some processing over that values periodically every 5 mm Sec
- Fixed priority Scheduler Will be Implemented
- ECU1 has 2 Tasks and the Idle Task
  1. Communication Task [5 mm sec]
  2. Processing Task [5 mm sec]
- When receives the sensors/switch states from ECU1 via CAN Bus then accordingly controls
  - ✓ Left Light(LL)
  - ✓ Right Light(RL)
  - ✓ Buzzer

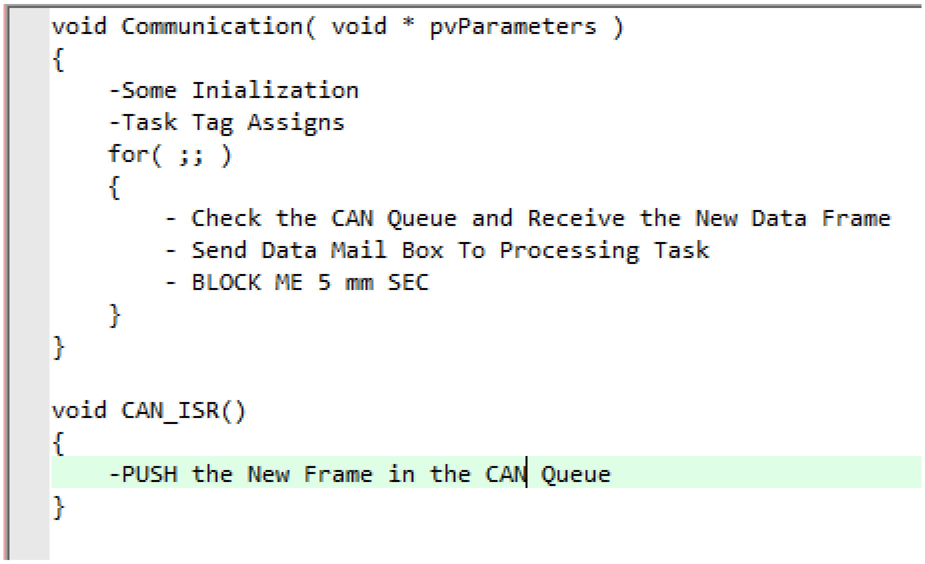
- Task Flowchart

Communication Task


Read & Parse the New Data Frame
Mail the Data Frame To the Processing Task
BLOCK 5 mm Sec

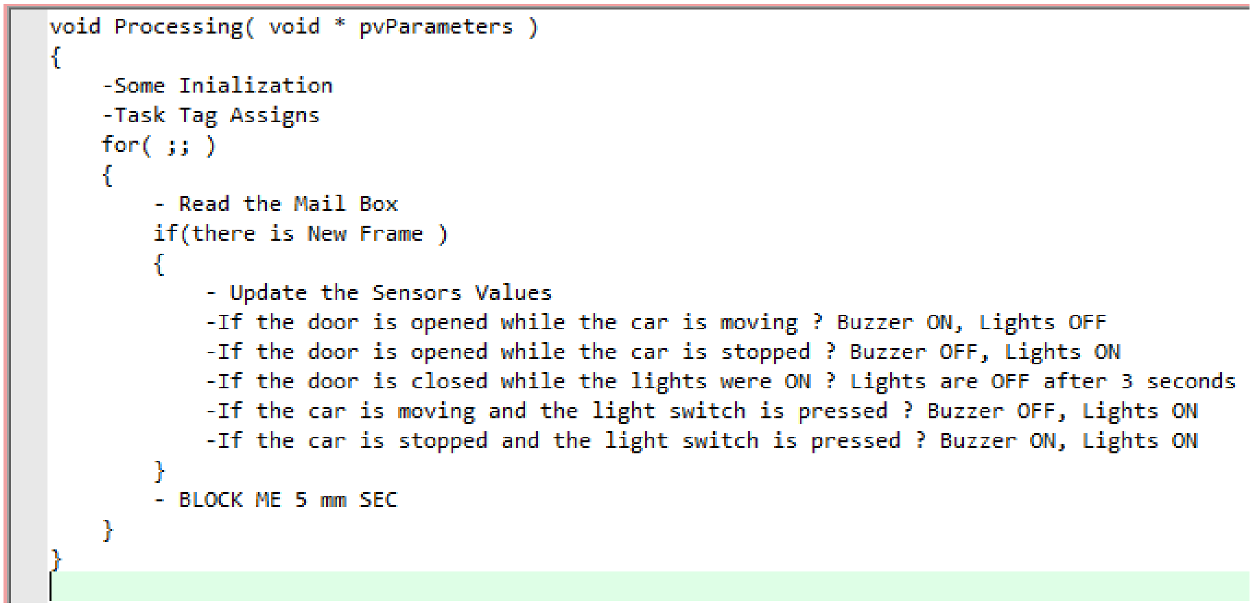

New Frame Received?
N
Processing Task
YE
D == Opend& S != 0 ?
YE
B   = ON
RL = ON
LL = ON
N
D == Opend& S == 0 ?
YE
B   = OFF
RL = ON
LL = ON
N
YE
D == Closed & SW == ON?
DELAY = 3 SEC
RL = OFF
LL = OFF
N
YE
S =! 0 & SW == ON ?
B = OFF
RL = ON
LL = ON
N
YE
S == 0 & SW == ON ?
B = ON
RL = ON
LL = ON
N

- ECU2 pseudo code
    - Communication Task pseudo code



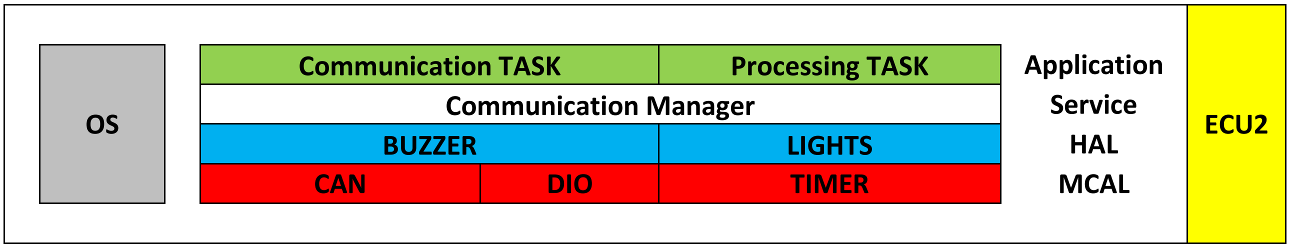
```
void Communication( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        - Check the CAN Queue and Receive the New Data Frame
        - Send Data Mail Box To Processing Task
        - BLOCK ME 5 mm SEC
    }
}

void CAN_ISR()
{
    -PUSH the New Frame in the CAN Queue
}
```

    - Processing Task pseudo code



```
void Processing( void * pvParameters )
{
    -Some Inialization
    -Task Tag Assigns
    for( ;; )
    {
        - Read the Mail Box
        if(there is New Frame )
        {
            - Update the Sensors Values
            -If the door is opened while the car is moving ? Buzzer ON, Lights OFF
            -If the door is opened while the car is stopped ? Buzzer OFF, Lights ON
            -If the door is closed while the lights were ON ? Lights are OFF after 3 seconds
            -If the car is moving and the light switch is pressed ? Buzzer OFF, Lights ON
            -If the car is stopped and the light switch is pressed ? Buzzer ON, Lights ON
        }
        - BLOCK ME 5 mm SEC
    }
}
```

- ECU2 Layered Architecture



OS

Communication TASK | Processing TASK

Communication Manager

BUZZER | LIGHTS

CAN | DIO | TIMER

Application

Service

HAL

MCAL

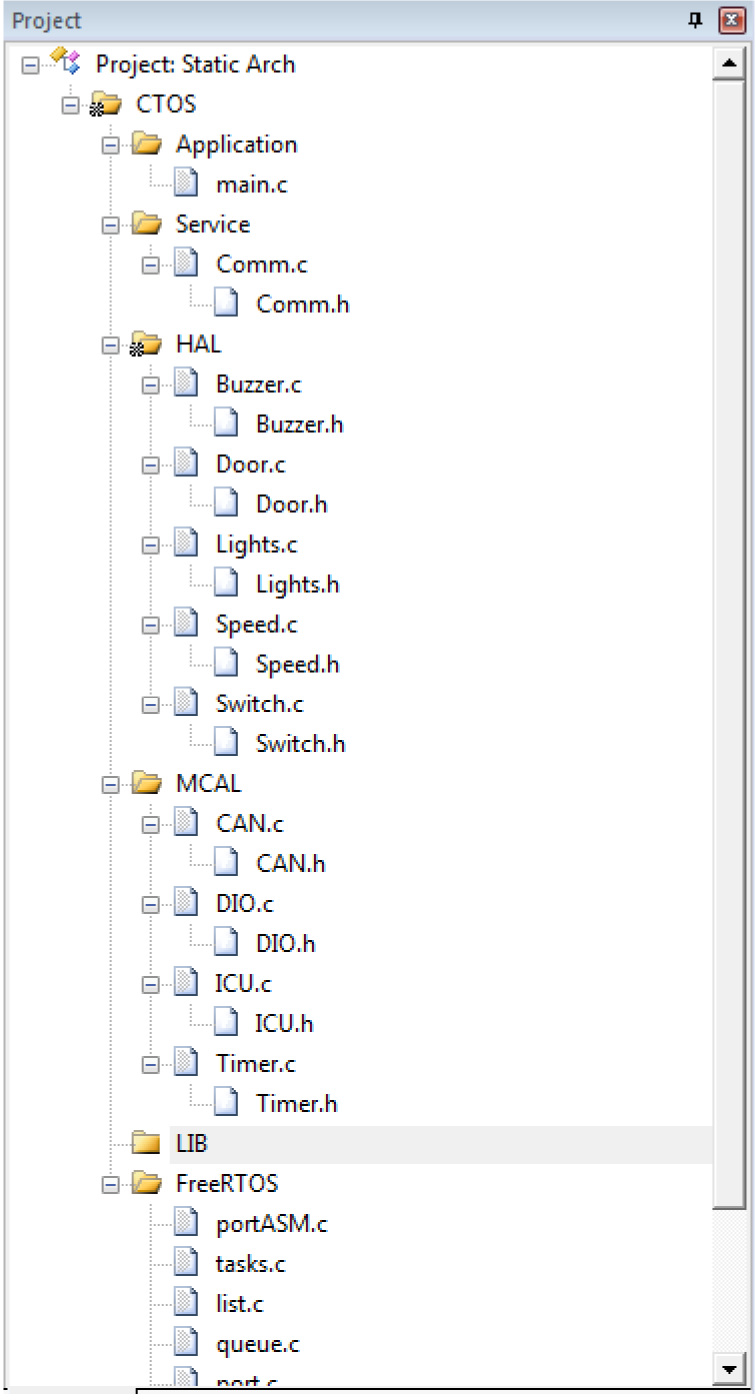
ECU2

- ECU2Modules APIs Description

Layer	Module	Function Statement	Arguments	Return Description
MCAL	DIO	void DIO_Init(DioConfigPtr_Type *ptr) Function	Struct holds the configurations for GPIO port-pin	void
		void DIO_Write(Pin_Type pin, Port_Type port, Value_Type Value)	Port – Pin - value	void
		value_TypeDIO_Read(Pin_Type pin, Port_Type port)	Port - Pin	Value_Typeenum states pin value (HIGH/LOW)
		void DIO_Toggle(Pin_Type pin, Port_Type port) Function Toggles some GPIO port-pin state	Port - Pin	void
	CAN	void CAN_Init(DioConfigPtr_Type *ptr) Function Initialize some GPIO port-pin as CAN	Struct holds the configurations for some GPIO port-pin	void
		void CAN_Send(uint32_t *Data) Function send data via CAN Bus	Pointer to the data to be sent	void
		void CAN_Receive(uint32 *Data) Function receive data from CAN Bus	Pointer to store received data in it	void

HAL	TIMER	void Timer_Init(TimerConfigPtr_Type *ptr)	Struct holds the configurations for Timer	void
		void StartTimer(TimerType)	timer	void
		void StopTimer(TimerType)	timer	void
		Void DelayMs(ms)	Delay value in millisecond	void
	Lights	Void Init_Lights (LightsConfigPtr *ptr)	Struct holds the configurations for initializing pin to work with the sensor	void
		void Set_LightState(LightsConfigPtr *ptr, StateType state)	Pointer refers to the required light GPIO	void
	Buzzer	Void Init_Buzzer (BuzzerConfigPtr *ptr)	Struct holds the configurations for initializing pin to work with the Buzzer	void
		void Set_BuzzerState(BuzzerConfigPtr *ptr, StateType state)	ptr:Pointer refers to the Buzzer State:Active/Disactive	void
Services	COMMUNICATION	Void Comm(u8 ID, u32 *Data)	ID : represents the required Comm protocol to send via Data : Pointer to data to be sent	void
Application	OS	Processing Task - Communication Task	2 Tasks	void

3. Folder Structure



```
Project
Project: Static Arch
  CTOS
    Application
      main.c
    Service
      Comm.c
        Comm.h
    HAL
      Buzzer.c
        Buzzer.h
      Door.c
        Door.h
      Lights.c
        Lights.h
      Speed.c
        Speed.h
      Switch.c
        Switch.h
    MCAL
      CAN.c
        CAN.h
      DIO.c
        DIO.h
      ICU.c
        ICU.h
      Timer.c
        Timer.h
    LIB
    FreeRTOS
      portASM.c
      tasks.c
      list.c
      queue.c
      port.c
```