**Project #3**

**Embedded Software Design**

1. **Fully Static Design.**

* System Hardware Requirements

ECU1

ECU2

D Sensor

L Switch

S Sensor

RL

LL

BUZ

CAN

**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

**Read Speed Sensor**

**Mail the Speed Value To the Communication Task**

**BLOCK 5 mm Sec**

Door Task

**Read Door Sensor**

**Mail the Door State Value To the Communication Task**

**BLOCK 10 mm Sec**

Switch Task

**Read Light Switch State**

**Mail the Light State Value To the Communication Task**

**BLOCK 20 mm Sec**

Communication Task

Read Mail Boxes

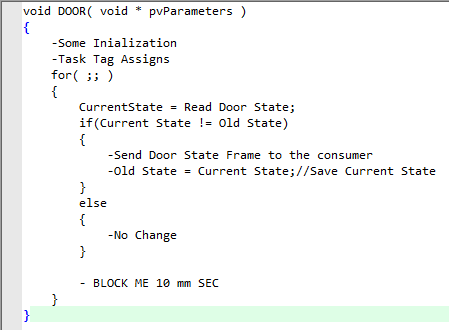
Build Data Frame

**Send the Data Frame**

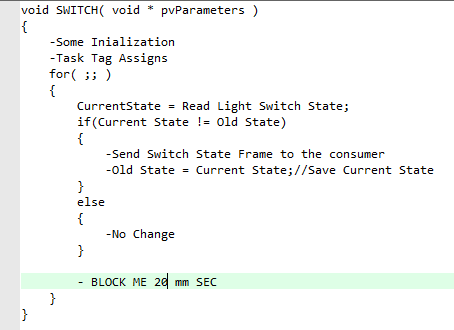
**To the CAN BUS**

**BLOCK 5 mm Sec**

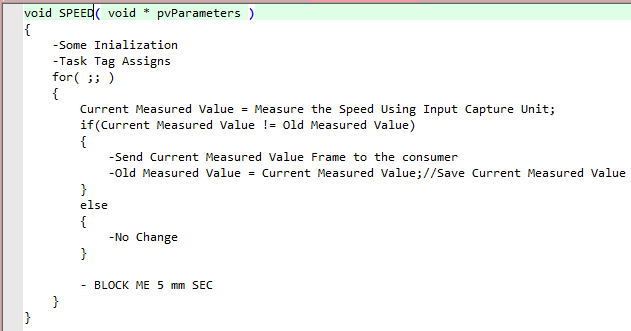
* ECU1 pseudo code
  + - DOOR Task pseudo code



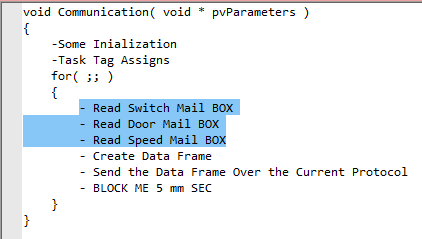
* + - Light Switch Task pseudo code



* + - Speed Measure Task pseudo code



* + - Communication Task pseudo code



* ECU1 Layered Architecture

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | |  |  | **ECU1** |
|  | **OS** |  | **Communication TASK** | **SWITCH TASK** | **SPEED TASK** | | **DOOR TASK** | **Application** |
|  |  | **Communication Manager** | | | | | **Service** |
|  |  | **Light Switch** | | **Door Sensor** | **Speed Sensor** | | **HAL** |
|  |  | **CAN** | **DIO** | **TIMER** | **ICU** | | **MCAL** |
|  |  |  |  |  |  |  | |  |

* 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 |

1. **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



New Frame Received?

YES

NO

D == Opend& S != 0 ?

YES

NO

B = ON

RL = ON

LL = ON

D == Opend& S == 0 ?

NO

B = OFF

RL = ON

LL = ON

YES

D == Closed & SW == ON?

NO

DELAY = 3 SEC

YES

RL = OFF

LL = OFF

S =! 0 & SW == ON ?

NO

YES

B = OFF

RL = ON

LL = ON

NO

YES

B = ON

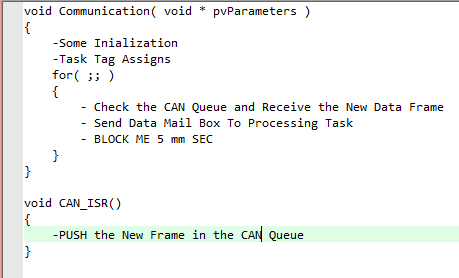
RL = ON

LL = ON

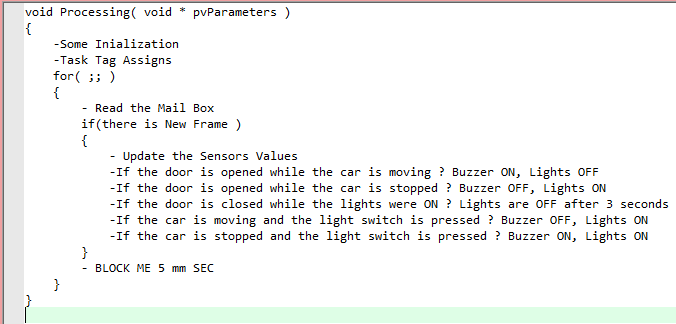
S == 0 & SW == ON ?

Processing Task

* ECU2 pseudo code
  + - Communication Task pseudo code



* + - Processing Task pseudo code



* ECU2 Layered Architecture

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | **ECU2** |
|  | **OS** |  | **Communication TASK** | | **Processing TASK** | | **Application** |
|  |  | **Communication Manager** | | | | **Service** |
|  |  | **BUZZER** | | **LIGHTS** | | **HAL** |
|  |  | **CAN** | **DIO** | **TIMER** | | **MCAL** |
|  |  |  |  |  |  |  |  |

* 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 |
| **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** | **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 |

1. **Folder Structure**

