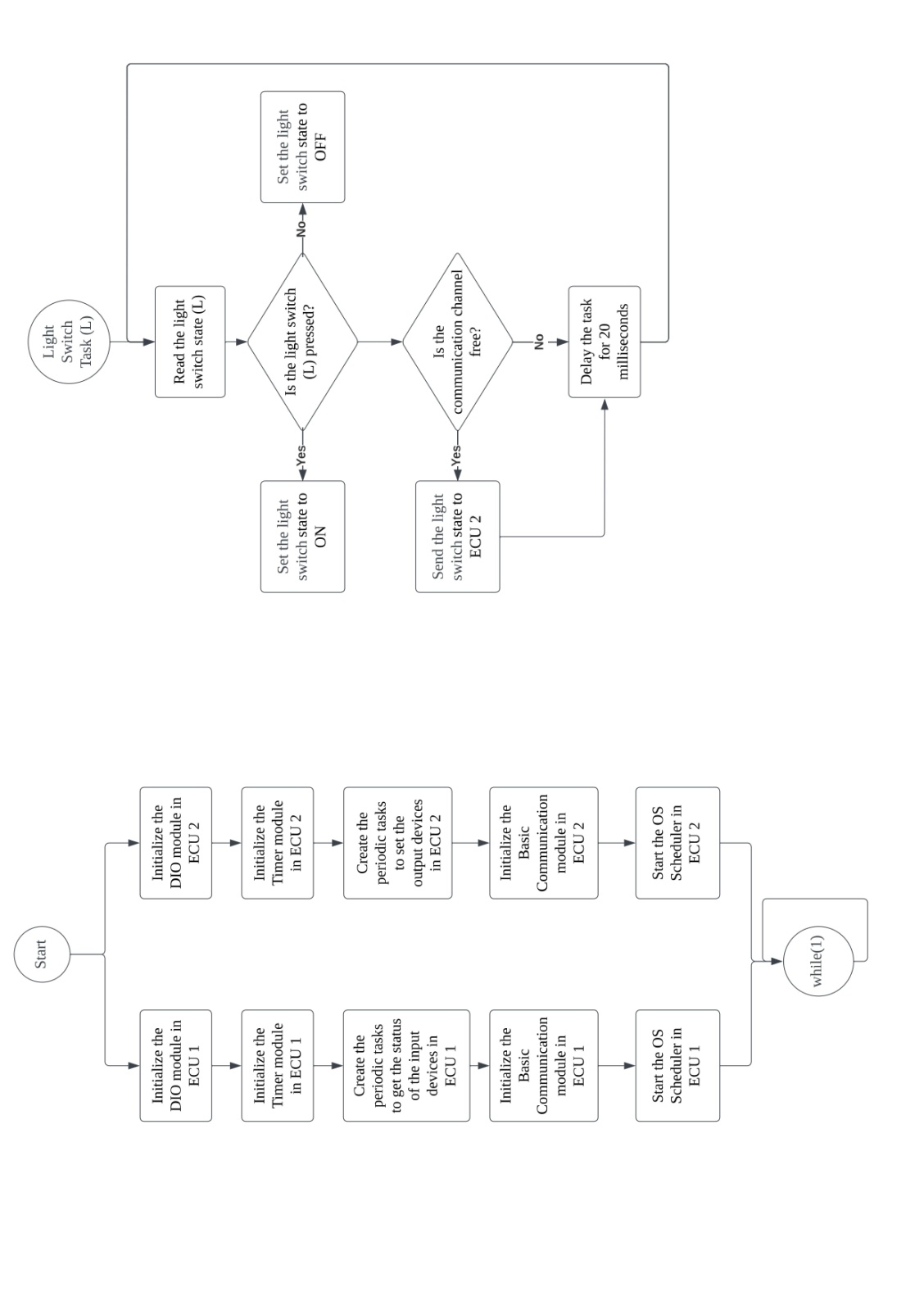
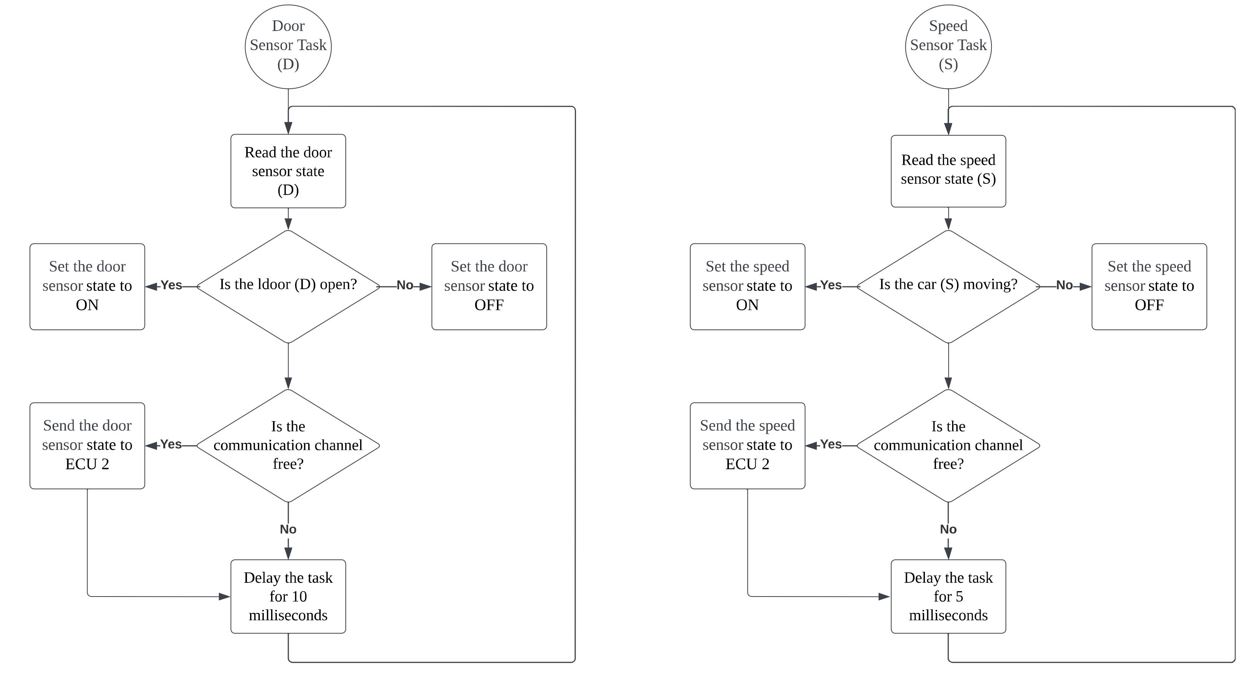
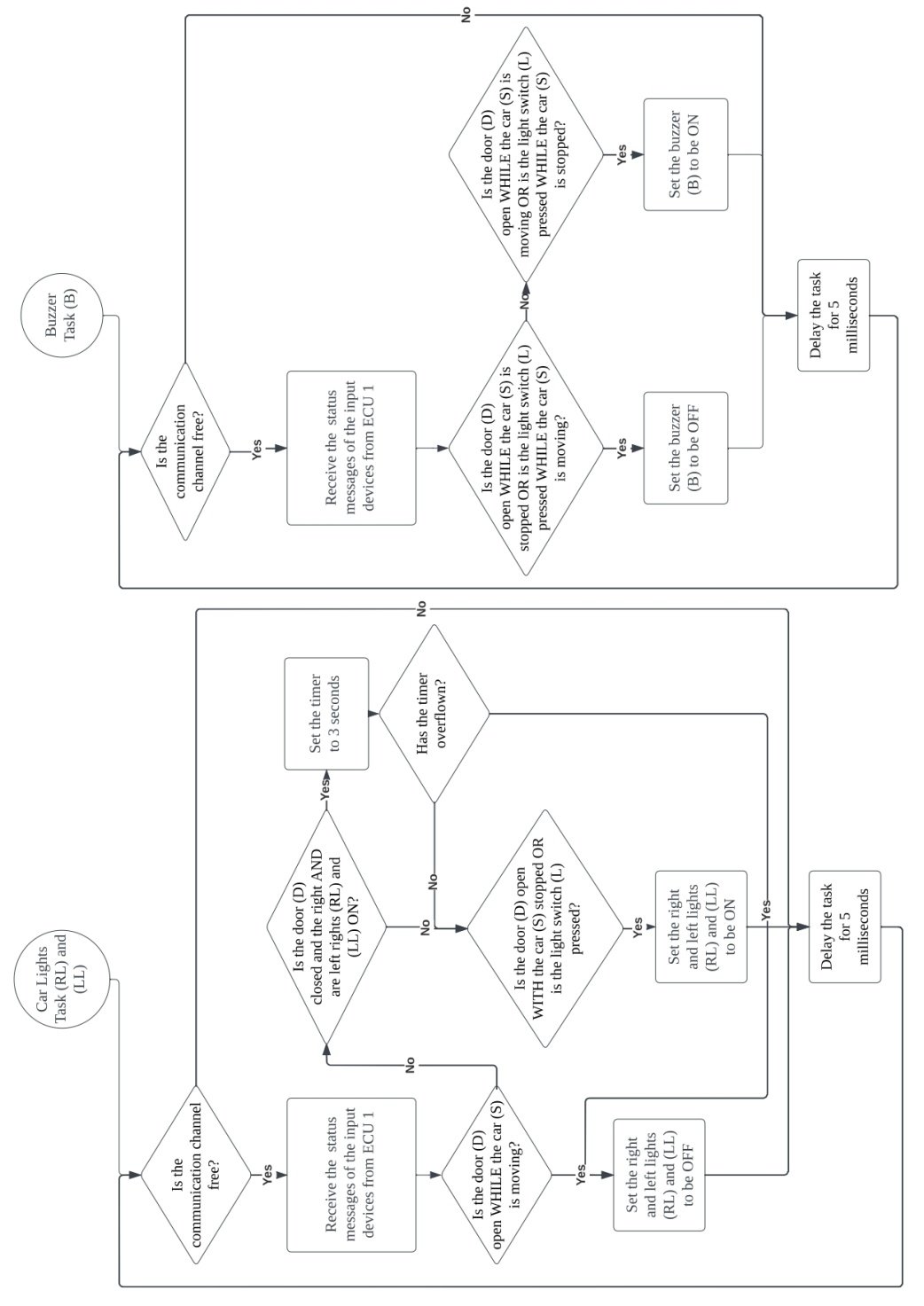
**Static Design Analysis of the Car System**

******

1. *Project Requirements:*

* *“You should draw and deliver the system schematic (Block Diagram) according to your requirements understanding, a screenshot is required.”*

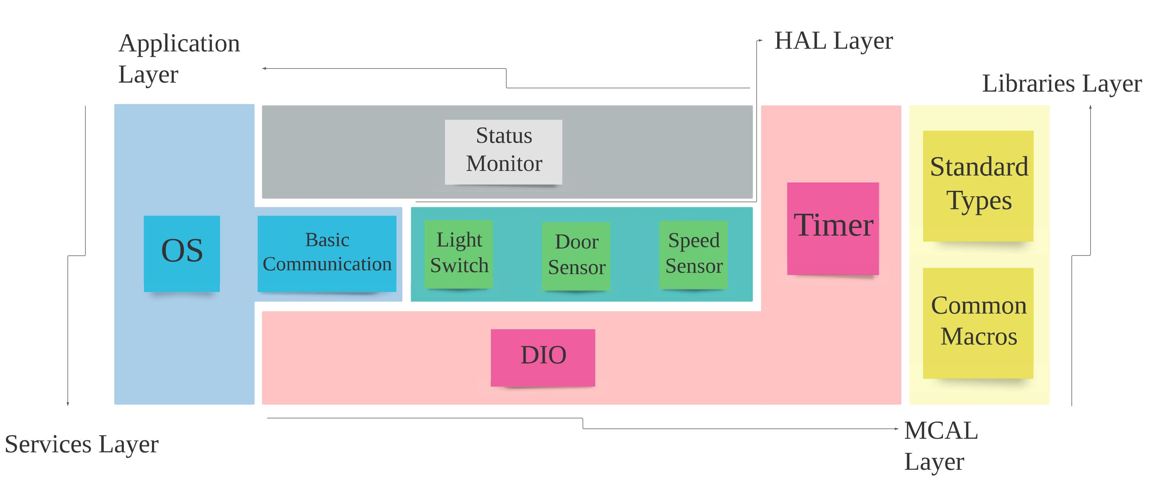
**

**

1. *Static Design Analysis:*

**For ECU 1:**

* *“Make the layered architecture.”*

****

* *“Specify ECU components and modules.”*
* “**DIO\_Init**”, “**DIO\_Read**”, “**DIO\_Write**” components 🡪 “**DIO**” module
* “**Timer\_Init**”, “**Timer\_Start**”, “**Timer\_Stop**” components 🡪 “**Timer**” module
* “**Task\_Create**”, “**Scheduler\_Start**” 🡪 “**OS**” module
* “**CAN\_Init**”, “**CAN\_Send**”, “**CAN\_Receive**”, “**CAN\_isFree**” 🡪 “**Basic Communication**” module
* “**LightSwitch\_Read**” 🡪 “**Light Switch**” module
* “**SpeedSensor\_Read**” 🡪 “**Speed Sensor**” module
* “**DoorSensor\_Read**” 🡪 “**Door Sensor**” module
* “**LightSwitch\_isPressed**”, “**Car\_isMoving**”, “**Door\_isOpen**” 🡪 “**Status Monitor**” module
* “**DIO\_StdType**”, “**Timer\_StdType**”, “**CAN\_StdType**”, “**LightSwitch\_StdType**”, “**SpeedSensor\_StdType**”, “**DoorSensor\_StdType**” 🡪 “**Standard Types**” module
* “**Platform\_Types**”, “**MCU\_HW**”, “**Compiler\_Config**” 🡪 “**Common Macros**” module
* *“Provide full detailed APIs for each module as well as a detailed description for the used typedefs.”*

APIs

|  |  |
| --- | --- |
| Service Name: | DIO\_Init |
| Syntax: | void DIO\_Init(const DIO\_ConfigType\* DIOConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | DIOConfig | a pointer to DIO configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the DIO module |
|  |  |
| Service Name: | DIO\_Read |
| Syntax: | DIO\_LevelType DIO\_Read(DIO\_PortType DIOPort, DIO\_PinType DIOPin) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read |
| Parameters (out): | DIOLevel | the level of the DIO read |
| Return Value: | DIO\_LevelType |
| Description: | This function reads the level of the DIO requested |
|  |  |
| Service Name: | DIO\_Write |
| Syntax: | void DIO\_Write(DIO\_PortType DIOPort, DIO\_PinType DIOPin, DIO\_LevelType DIOLevel) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read /  DIOLevel | the level of the DIO to write to the DIO |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function writes the level requested to the DIO |
|  |  |
| Service Name: | Timer\_Init |
| Syntax: | void Timer\_Init(const Timer\_ConfigType\* TimerConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TimerConfig | a pointer to the timer configuration |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the timer module |
|  |  |
| Service Name: | Timer\_Start |
| Syntax: | void Timer\_Start(Timer\_LevelType TimerValue) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TimerValue | the value of the timer count |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function starts the timer/counter with the value requested |
|  |  |
| Service Name: | Timer\_Stop |
| Syntax: | void Timer\_Stop(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function stops the timer/counter |
|  |  |
| Service Name: | Task\_Create |
| Syntax: | void Task\_Create(const Task\_ConfigType\* TaskConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TaskConfig | a pointer to the task configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the OS task |
|  |  |
| Service Name: | Scheduler\_Start |
| Syntax: | void Scheduler\_Start(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function starts the OS scheduler to select the task to run |
|  |  |
| Service Name: | CAN\_Init |
| Syntax: | void CAN\_Init(const CAN\_ConfigType\* CANConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANConfig | a pointer to the CAN channel configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the CAN channel of the basic communication module |
|  |  |
| Service Name: | CAN\_Send |
| Syntax: | void CAN\_Send(CAN\_BufferType\* CANBuffer, CAN\_LevelType CANBufferLength) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANBuffer | a pointer to the buffer to be sent through the CAN channel /  CANBufferLength | the length of the buffer to be sent through the CAN channel |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function sends the buffer with the specified length through the CAN channel |
|  |  |
| Service Name: | CAN\_Receive |
| Syntax: | void CAN\_Receive(CAN\_BufferType\* CANBuffer) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANBuffer | a pointer to the buffer to be received through the CAN channel |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function receives the buffer through the CAN channel |
|  |  |
| Service Name: | LightSwitch\_Read |
| Syntax: | LightSwitch\_LevelType LightSwitch\_Read(LightSwitch\_PortType LightSwitchPort,  LightSwitch\_PinType LightSwitchPin) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | LightSwitchPort | the port of the light switch to be read /  LightSwitchPin | the pin of the light switch to be read |
| Parameters (out): | LightSwitchLevel | the level of the light switch read |
| Return Value: | LightSwitch\_LevelType |
| Description: | This function reads the level of the light switch requested to see if pressed or not |
|  |  |
| Service Name: | SpeedSensor\_Read |
| Syntax: | SpeedSensor\_LevelType SpeedSensor\_Read  (SpeedSensor\_PortType SpeedSensorPort, SpeedSensor\_PinType SpeedSensorPin) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | SpeedSensorPort | the port of the speed sensor to be read /  SpeedSensorPin | the pin of the speed sensor to be read |
| Parameters (out): | SpeedSensorLevel | the level of the speed sensor read |
| Return Value: | SpeedSensor\_LevelType |
| Description: | This function reads the level of the speed sensor requested to see if the car is moving or not |
|  |  |
| Service Name: | DoorSensor\_Read |
| Syntax: | DoorSensor\_LevelType DoorSensor\_Read(DoorSensor\_PortType DoorSensorPort,  DoorSensor\_PinType DoorSensorPin) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | DoorSensorPort | the port of the door sensor to be read /  DoorSensorPin | the pin of the door sensor to be read |
| Parameters (out): | DoorSensorLevel | the level of the door sensor read |
| Return Value: | DoorSensor\_LevelType |
| Description: | This function reads the level of the door sensor requested to see if open or not |

|  |  |
| --- | --- |
| Service Name: | LightSwitch\_isPressed |
| Syntax: | LightSwitch\_LevelType LightSwitch\_isPressed(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | LightSwitchLevel | the level of the light switch read |
| Return Value: | LightSwitch\_LevelType |
| Description: | This function reads the level of the light switch configured to see if pressed or not |
|  |  |
| Service Name: | Car\_isMoving |
| Syntax: | SpeedSensor\_LevelType Car\_isMoving(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | SpeedSensorLevel | the level of the speed sensor read |
| Return Value: | SpeedSensor\_LevelType |
| Description: | This function reads the level of the speed sensor configured to see if the car is moving or not |
|  |  |
| Service Name: | Door\_isOpen |
| Syntax: | DoorSensor\_LevelType Door\_isOpen(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | DoorSensorLevel | the level of the door sensor read |
| Return Value: | DoorSensor\_LevelType |
| Description: | This function reads the level of the door sensor configured to see if the door is open or not |

Typedefs

|  |  |
| --- | --- |
| Name: | DIO\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the DIO module |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the DIO pin is LOW (0) or HIGH (1) |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the DIO pin stored as an enum member |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the DIO pin specified stored as an enum member |
| Available via: | DIO.h |
|  |  |
| Name: | Timer\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the timer module |
| Available via: | Timer.h |
|  |  |
| Name: | Timer\_LevelType |
| Type: | Int |
| Range: | 0 to 65,535 |
| Description: | The count value given to the timer/counter register |
| Available via: | Timer.h |
|  |  |
| Name: | Task\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the tasks |
| Available via: | OS.h |
|  |  |
| Name: | CAN\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the CAN module |
| Available via: | CAN.h |
|  |  |
| Name: | CAN\_BufferType |
| Type: | Char |
| Range: | 0 to 255 |
| Description: | The CAN frame holding the message sent/received |
| Available via: | CAN.h |
|  |  |
| Name: | CAN\_LevelType |
| Type: | Char |
| Range: | 0 to 255 |
| Description: | The length of the message to store in the CAN frame |
| Available via: | CAN.h |
|  |  |
| Name: | LightSwitch\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the light switch is RELEASED (0) or PRESSED (1) |
| Available via: | LightSwitch.h |
|  |  |
| Name: | LightSwitch\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the light switch pin based on the DIO enum numbers |
| Available via: | LightSwitch.h |
|  |  |
| Name: | LightSwitch\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the light switch pin based on the DIO enum members |
| Available via: | LightSwitch.h |
|  |  |
| Name: | DoorSensor\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the door is CLOSED (0) or OPEN (1) |
| Available via: | DoorSensor.h |
|  |  |
| Name: | DoorSensor\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the door sensor pin based on the DIO enum numbers |
| Available via: | DoorSensor.h |
|  |  |
| Name: | DoorSensor\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the door sensor pin based on the DIO enum members |
| Available via: | DoorSensor.h |
|  |  |
| Name: | SpeedSensor\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the car is STOPPED (0) or MOVING (1) |
| Available via: | SpeedSensor.h |
|  |  |
| Name: | SpeedSensor\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the speed sensor pin based on the DIO enum numbers |
| Available via: | SpeedSensor.h |
|  |  |
| Name: | SpeedSensor\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the speed sensor pin based on the DIO enum members |
| Available via: | SpeedSensor.h |
|  |  |

* *“Prepare your folder structure according to the previous points.”*

.

├── MCAL/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── DIO\_Cfg.h

│ │ │ ├── DIO\_Lcfg.h

│ │ │ ├── Timer\_Cfg.h

│ │ │ └── Timer\_Lcfg.h

│ │ └── src/

│ │ ├── DIO\_Lcfg.c

│ │ └── Timer\_Lcfg.c

│ ├── DIO/

│ │ ├── inc/

│ │ │ └── DIO.h

│ │ └── src/

│ │ └── DIO.c

│ └── Timer/

│ ├── inc/

│ │ └── Timer.h

│ └── src/

│ └── Timer.c

├── HAL/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── LightSwitch\_Cfg.h

│ │ │ ├── LightSwitch\_Lcfg.h

│ │ │ ├── SpeedSensor\_Cfg.h

│ │ │ ├── SpeedSensor\_Lcfg.h

│ │ │ ├── DoorSensor\_Cfg.h

│ │ │ └── DoorSensor\_Lcfg.h

│ │ └── src/

│ │ ├── LightSwitch\_Lcfg.c

│ │ ├── SpeedSensor\_Lcfg.c

│ │ └── DoorSensor\_Lcfg.c

│ ├── LightSwitch/

│ │ ├── inc/

│ │ │ └── LightSwitch.h

│ │ └── src/

│ │ └── LightSwitch.c

│ ├── SpeedSensor/

│ │ ├── inc/

│ │ │ └── SpeedSensor.h

│ │ └── src/

│ │ └── SpeedSensor.c

│ └── DoorSensor/

│ ├── inc/

│ │ └── DoorSensor.h

│ └── src/

│ └── DoorSensor.c

├── Service/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── OS\_Cfg.h

│ │ │ ├── OS\_Lcfg.h

│ │ │ ├── CAN\_Cfg.h

│ │ │ └── CAN\_Lcfg.h

│ │ └── src/

│ │ ├── OS\_Lcfg.c

│ │ └── CAN\_Lcfg.c

│ ├── OS/

│ │ ├── inc/

│ │ │ └── OS.h

│ │ └── src/

│ │ └── OS.c

│ └── CAN/

│ ├── inc/

│ │ └── CAN.h

│ └── src/

│ └── CAN.c

├── Library/

│ ├── StdTypes/

│ │ └── inc/

│ │ └── StdTypes.h

│ └── Common/

│ └── inc/

│ ├── PlatformTypes.h

│ ├── MCU\_HW.h

│ └── Compiler\_Cfg.h

└── Application/  
 └── main.c

└── StatusMonitor/

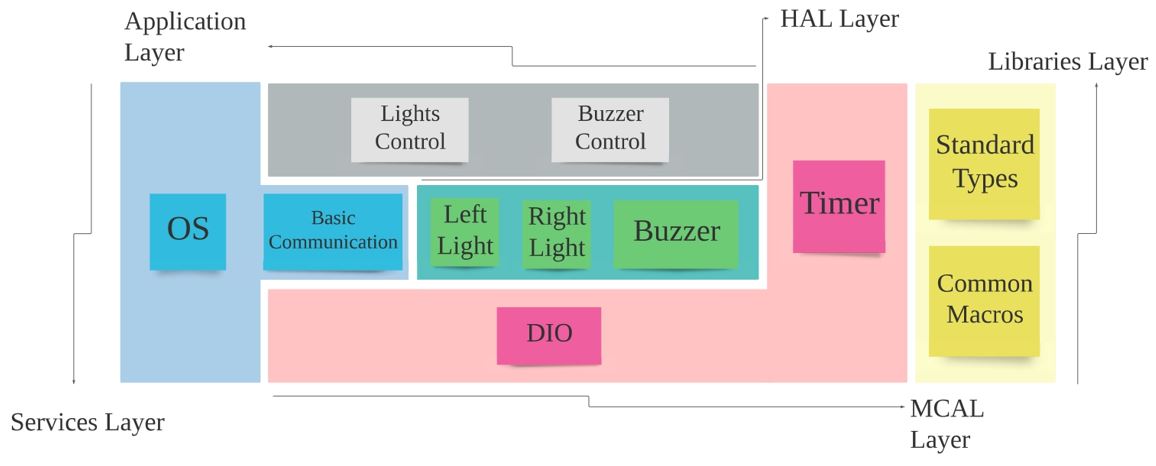
├── inc/

│ └── StatusMonitor.h

└── src/

└── StatusMonitor.c

**For ECU 2:**

* *“Make the layered architecture.”*
* *“Specify ECU components and modules.”*
* “**DIO\_Init**”, “**DIO\_Read**”, “**DIO\_Write**” components 🡪 “**DIO**” module
* “**Timer\_Init**”, “**Timer\_Start**”, “**Timer\_Stop**” components 🡪 “**Timer**” module
* “**Task\_Create**”, “**Scheduler\_Start**” 🡪 “**OS**” module
* “**CAN\_Init**”, “**CAN\_Send**”, “**CAN\_Receive**”, “**CAN\_isFree**” 🡪 “**Basic Communication**” module
* “**LeftLight\_Write**” 🡪 “**Left Light**” module
* “**RightLight\_Write**” 🡪 “**Right Light**” module
* “**Buzzer\_Write**” 🡪 “**Buzzer**” module
* “**LeftLight\_TurnOn**”, “**LeftLight\_TurnOff**”,

“**RightLight\_TurnOn**”, “**RightLight\_TurnOff**” 🡪 “**Lights Control**” module

* “**Buzzer\_TurnOn**”, “**Buzzer\_TurnOff**” 🡪 “**Buzzer Control**” module
* “**DIO\_StdType**”, “**Timer\_StdType**”, “**CAN\_StdType**”, “**LightLevel\_StdType**”, “**BuzzerLevel\_StdType**” 🡪 “**Standard Types**” module
* “**Platform\_Types**”, “**MCU\_HW**”, “**Compiler\_Config**” 🡪 “**Common Macros**” module
* *“Provide full detailed APIs for each module as well as a detailed description for the used typedefs.”*

APIs

|  |  |
| --- | --- |
| Service Name: | DIO\_Init |
| Syntax: | void DIO\_Init(const DIO\_ConfigType\* DIOConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | DIOConfig | a pointer to DIO configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the DIO module |
|  |  |
| Service Name: | DIO\_Read |
| Syntax: | DIO\_LevelType DIO\_Read(DIO\_PortType DIOPort, DIO\_PinType DIOPin) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read |
| Parameters (out): | DIOLevel | the level of the DIO read |
| Return Value: | DIO\_LevelType |
| Description: | This function reads the level of the DIO requested |
|  |  |
| Service Name: | DIO\_Write |
| Syntax: | void DIO\_Write(DIO\_PortType DIOPort, DIO\_PinType DIOPin, DIO\_LevelType DIOLevel) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | DIOPort | the port of the DIO to be read / DIOPin | the pin of the DIO to be read /  DIOLevel | the level of the DIO to write to the DIO |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function writes the level requested to the DIO |
|  |  |
| Service Name: | Timer\_Init |
| Syntax: | void Timer\_Init(const Timer\_ConfigType\* TimerConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TimerConfig | a pointer to the timer configuration |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the timer module |
|  |  |
| Service Name: | Timer\_Start |
| Syntax: | void Timer\_Start(Timer\_LevelType TimerValue) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TimerValue | the value of the timer count |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function starts the timer/counter with the value requested |
|  |  |
| Service Name: | Timer\_Stop |
| Syntax: | void Timer\_Stop(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function stops the timer/counter |
|  |  |
| Service Name: | Task\_Create |
| Syntax: | void Task\_Create(const Task\_ConfigType\* TaskConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | TaskConfig | a pointer to the task configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the OS task |
|  |  |
| Service Name: | Scheduler\_Start |
| Syntax: | void Scheduler\_Start(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function starts the OS scheduler to select the task to run |
|  |  |
| Service Name: | CAN\_Init |
| Syntax: | void CAN\_Init(const CAN\_ConfigType\* CANConfig) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANConfig | a pointer to the CAN channel configurations |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function initializes the CAN channel of the basic communication module |
|  |  |
| Service Name: | CAN\_Send |
| Syntax: | void CAN\_Send(CAN\_BufferType\* CANBuffer, CAN\_LevelType CANBufferLength) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANBuffer | a pointer to the buffer to be sent through the CAN channel /  CANBufferLength | the length of the buffer to be sent through the CAN channel |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function sends the buffer with the specified length through the CAN channel |
|  |  |
| Service Name: | CAN\_Receive |
| Syntax: | void CAN\_Receive(CAN\_BufferType\* CANBuffer) |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | CANBuffer | a pointer to the buffer to be received through the CAN channel |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function receives the buffer through the CAN channel |

|  |  |
| --- | --- |
| Service Name: | LeftLight\_Write |
| Syntax: | void LeftLight\_Write(Light\_PortType LeftLightPort, Light\_PinType LeftLightPin,  Light\_LevelType LeftLightLevel) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | LeftLightPort | the port of the left light to be written /  LeftLightPin | the pin of the left light to be written /  LeftLightLevel | the level of the left light to be written |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function writes the level specified to the left light |
|  |  |
| Service Name: | RightLight\_Write |
| Syntax: | void RightLight\_Write(Light\_PortType RightLightPort, Light\_PinType RightLightPin,  Light\_LevelType RightLightLevel) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | RightLightPort | the port of the right light to be written /  RightLightPin | the pin of the right light to be written /  RightLightLevel | the level of the right light to be written |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function writes the level specified to the right light |
|  |  |
| Service Name: | Buzzer\_Write |
| Syntax: | void Buzzer\_Write(Buzzer\_PortType BuzzerPort, Buzzer\_PinType BuzzerPin,  Buzzer\_LevelType BuzzerLevel) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | BuzzerPort | the port of the buzzer to be written /  BuzzerPin | the pin of the buzzer to be written /  BuzzerLevel | the level of the buzzer to be written |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function writes the level specified to the buzzer |
|  |  |
| Service Name: | LeftLight\_TurnOn |
| Syntax: | void LeftLight\_TurnOn(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns on the left light |
|  |  |
| Service Name: | LeftLight\_TurnOff |
| Syntax: | void LeftLight\_TurnOff(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns off the left light |
|  |  |
| Service Name: | RightLight\_TurnOn |
| Syntax: | void RightLight\_TurnOn(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns on the right light |
|  |  |
| Service Name: | RightLight\_TurnOff |
| Syntax: | void RightLight\_TurnOff(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns off the right light |
|  |  |
| Service Name: | Buzzer\_TurnOn |
| Syntax: | void Buzzer\_TurnOn(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns on the buzzer |
|  |  |
| Service Name: | Buzzer\_TurnOff |
| Syntax: | void Buzzer\_TurnOff(void) |
| Sync/Async: | Synchronous |
| Reentrancy: | Non-reentrant |
| Parameters (in): | None |
| Parameters (out): | None |
| Return Value: | None |
| Description: | This function turns off the buzzer |

Typedefs

|  |  |
| --- | --- |
| Name: | DIO\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the DIO module |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the DIO pin is LOW (0) or HIGH (1) |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the DIO pin stored as an enum member |
| Available via: | DIO.h |
|  |  |
| Name: | DIO\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the DIO pin specified stored as an enum member |
| Available via: | DIO.h |
|  |  |
| Name: | Timer\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the timer module |
| Available via: | Timer.h |
|  |  |
| Name: | Timer\_LevelType |
| Type: | Int |
| Range: | 0 to 65,535 |
| Description: | The count value given to the timer/counter register |
| Available via: | Timer.h |
|  |  |
| Name: | Task\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the tasks |
| Available via: | OS.h |
|  |  |
| Name: | CAN\_ConfigType |
| Type: | Structure |
| Range: | Implementation Specific |
| Description: | The data structure containing the overall initialization data for the CAN module |
| Available via: | CAN.h |
|  |  |
| Name: | CAN\_BufferType |
| Type: | Char |
| Range: | 0 to 255 |
| Description: | The CAN frame holding the message sent/received |
| Available via: | CAN.h |
|  |  |
| Name: | CAN\_LevelType |
| Type: | Char |
| Range: | 0 to 255 |
| Description: | The length of the message to store in the CAN frame |
| Available via: | CAN.h |
|  |  |
| Name: | LightSwitch\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the light switch is RELEASED (0) or PRESSED (1) |
| Available via: | LightSwitch.h |
|  |  |
| Name: | LightSwitch\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the light switch pin based on the DIO enum numbers |
| Available via: | LightSwitch.h |
|  |  |
| Name: | LightSwitch\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the light switch pin based on the DIO enum members |
| Available via: | LightSwitch.h |
|  |  |
| Name: | DoorSensor\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the door is CLOSED (0) or OPEN (1) |
| Available via: | DoorSensor.h |
|  |  |
| Name: | DoorSensor\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the door sensor pin based on the DIO enum numbers |
| Available via: | DoorSensor.h |
|  |  |
| Name: | DoorSensor\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the door sensor pin based on the DIO enum members |
| Available via: | DoorSensor.h |
|  |  |
| Name: | SpeedSensor\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value indicating if the car is STOPPED (0) or MOVING (1) |
| Available via: | SpeedSensor.h |
|  |  |
| Name: | SpeedSensor\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the speed sensor pin based on the DIO enum numbers |
| Available via: | SpeedSensor.h |
|  |  |
| Name: | SpeedSensor\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the speed sensor pin based on the DIO enum members |
| Available via: | SpeedSensor.h |
|  |  |
| Name: | Light\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value setting the car light OFF (0) or ON (1) |
| Available via: | LeftLight.h/RightLight.h |
|  |  |
| Name: | Light\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the car light pin based on the DIO enum numbers |
| Available via: | LeftLight.h/RightLight.h |
|  |  |
| Name: | Light\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the car light pin based on the DIO enum members |
| Available via: | LeftLight.h/RightLight.h |
|  |  |
| Name: | Buzzer\_LevelType |
| Type: | Char |
| Range: | 0 to 1 |
| Description: | The boolean value setting the buzzer OFF (0) or ON (1) |
| Available via: | Buzzer.h |
|  |  |
| Name: | Buzzer\_PortType |
| Type: | Char |
| Range: | 0 to Number of Ports |
| Description: | The number of the port containing the buzzer pin based on the DIO enum numbers |
| Available via: | Buzzer.h |
|  |  |
| Name: | Buzzer\_PinType |
| Type: | Char |
| Range: | 0 to 7 |
| Description: | The number of the buzzer pin based on the DIO enum members |
| Available via: | Buzzer.h |

* *“Prepare your folder structure according to the previous points.”*

.

├── MCAL/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── DIO\_Cfg.h

│ │ │ ├── DIO\_Lcfg.h

│ │ │ ├── Timer\_Cfg.h

│ │ │ └── Timer\_Lcfg.h

│ │ └── src/

│ │ ├── DIO\_Lcfg.c

│ │ └── Timer\_Lcfg.c

│ ├── DIO/

│ │ ├── inc/

│ │ │ └── DIO.h

│ │ └── src/

│ │ └── DIO.c

│ └── Timer/

│ ├── inc/

│ │ └── Timer.h

│ └── src/

│ └── Timer.c

├── HAL/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── LeftLight\_Cfg.h

│ │ │ ├── LeftLight\_Lcfg.h

│ │ │ ├── RightLight\_Cfg.h

│ │ │ ├── RightLight\_Lcfg.h

│ │ │ ├── Buzzer\_Cfg.h

│ │ │ └── Buzzer\_Lcfg.h

│ │ └── src/

│ │ ├── Buzzer\_Lcfg.c

│ │ ├── Buzzer\_Lcfg.c

│ │ └── Buzzer\_Lcfg.c

│ ├── LeftLight/

│ │ ├── inc/

│ │ │ └── LeftLight.h

│ │ └── src/

│ │ └── LeftLight.c

│ ├── RightLight/

│ │ ├── inc/

│ │ │ └── RightLight.h

│ │ └── src/

│ │ └── RightLight.c

│ └── Buzzer/

│ ├── inc/

│ │ └── Buzzer.h

│ └── src/

│ └── Buzzer.c

├── Service/

│ ├── Configuration/

│ │ ├── inc/

│ │ │ ├── OS\_Cfg.h

│ │ │ ├── OS\_Lcfg.h

│ │ │ ├── CAN\_Cfg.h

│ │ │ └── CAN\_Lcfg.h

│ │ └── src/

│ │ ├── OS\_Lcfg.c

│ │ └── CAN\_Lcfg.c

│ ├── OS/

│ │ ├── inc/

│ │ │ └── OS.h

│ │ └── src/

│ │ └── OS.c

│ └── CAN/

│ ├── inc/

│ │ └── CAN.h

│ └── src/

│ └── CAN.c

├── Library/

│ ├── StdTypes/

│ │ └── inc/

│ │ └── StdTypes.h

│ └── Common/

│ └── inc/

│ ├── PlatformTypes.h

│ ├── MCU\_HW.h

│ └── Compiler\_Cfg.h

└── Application/  
 ├── main.c

├── LightsControl/

│ ├── inc/

│ │ └── LightsControl.h

│ └── src/

│ └── LightsControl.c

└── BuzzerControl/

├── inc/

│ └── BuzzerControl.h

└── src/

└── BuzzerControl.c