EgFWD

Advanced Embedded Systems

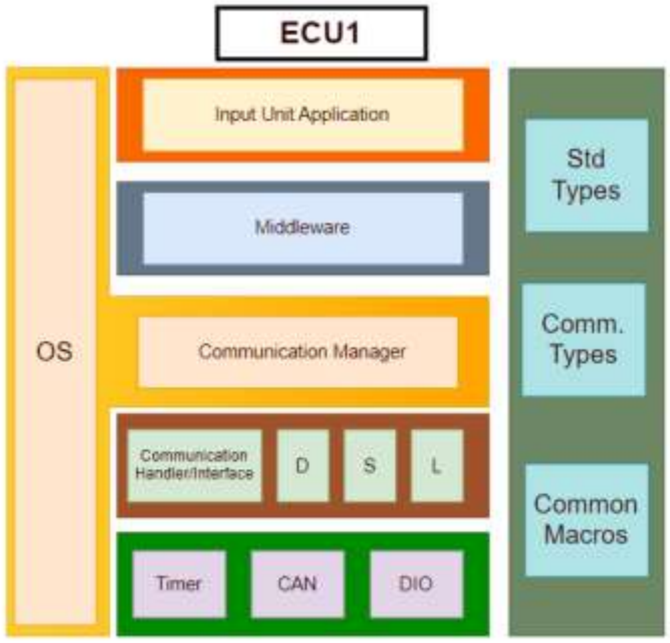
Project (3) | System Design

Automotive Door Control System Design Report

Static Design

For ECU 1:

1) Make the layered architecture



2) Specify ECU components and modules

a. Application: Where system logic and tasks are implemented.
   i. main.c
b. Middleware: For routing the system to its desired destination. For example, either to send data from input devices and so being routed to communication manager or else saving this data in EEPROM and so being routed to memory manager.
   i. middleware.c | middleware.h
c. Service:
   i. RTOS: Handling tasks and operating system as stated in project rubric.
      1. All RTOS related files from previous course(RTOS Porting Section | memory management, heap, tasks.c, config.h …etc)
   ii. Communication(BCM): Used to send Status messages.
      1. BCM.c, BCM.h
d. Common: For standard data types and common macros.
   i. std_types.h
   ii. common_macros.h
   iii. communication_types.h
e. HAL:
   i. Onboard/ECUAL: containing Door sensor, Light switch, Speed sensor and all external input devices.
      1. doorSensor.c, doorSensor.h

        2. lightSwitch.c, lightSwitch.h
        3. speedSensor.c, speedSensor.h
    ii. MCAL:
        1. Timer
            a. timer.c, timer.h
            b. timer_cfg.c, timer_cfg.h
        2. CAN
            a. CAN.c, CAN.h
            b. CAN_cfg.c, CAN_cfg.h
        3. DIO
            a. DIO.c. DIO.h
            b. DIO_cfg.c, DIO_cfg.h

3) Provide full detailed APIs for each module as well as a detailed description for the used typedefs

- ❖ Application Module:
  - ➢ API-Functions:
    - ▪ App_InitError_t App_init ();
      - To create tasks and call all modules' init APIs as GPIO, Timer…etc.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
    - ▪ void App_Start ();
      - All program logic goes here.
      - Reentrant, Asynchronous, Non-Recursive Function.
    - ▪ void timerSetCallback();
      - Non-Reentrant, Synchronous, Non-Recursive Function.
  - ➢ API-Types:
    - ▪ App_InitError
      - To return initialization state, either success or fail.
- ❖ Middleware Module:
  - ➢ API-Functions:
    - ▪ Middleware_SendState SendData (commProtocol_t, uint32_t data, dataSize_t);
      - To send data using specified communication protocol.
      - Reentrant, Asynchronous, Non-Recursive Function.

    - Middleware_SaveState SaveData (memoryType_t, dataSize_t, uint32_t data);
        - To save data on internal EEPROM, external one or any other.
        - Reentrant, Asynchronous, Non-Recursive Function.
  - API-Types:
    - Middleware_SaveState:
        - To return save state, either success or fail.
    - Middleware_SendState:
        - To return send state, either success or fail.
- BCM Module:
  - API-Function:
    - Send_CAN (CAN_config_ptr, dataSize_t, uint32_t data);
        - To route data to CAN module with specified configurations.
        - Reentrant, Asynchronous, Non-Recursive Function.
    - Send_UART (UART_config_ptr, dataSize_t, uint32_t data);
        - To route data to UART module with specified configurations.
        - Reentrant, Asynchronous, Non-Recursive Function.
    - Send_SPI (SPI_config_ptr, dataSize_t, uint32_t data);
        - To route data to SPI module with specified configurations.
        - Reentrant, Asynchronous, Non-Recursive Function.
    - Send_I2C (I2C_config_ptr, dataSize_t, uint32_t data);
        - To route data to I2C module with specified configurations.
        - Reentrant, Asynchronous, Non-Recursive Function.
  - API-Types:
    - BCM_SendState:
        - To return send state, either success or fail.
- Std_Types Module:
  - API-Types: (mentioned below are just a few examples, feel free to add what’s needed)
    - uint8, uint16, uint32, int8, int16, int32…etc.
    - Boolean_t
    - NULL

- Communication_Types Module:
  - API-Types:
    - commProtocol_t
      - struct for all communication protocols as SPI, UART…etc.
    - baudRate_t
      - struct for all available baud rates.
    - parityType_t
      - struct for either even or edd parity.
    - memoryType_t
      - struct for internal, external EEPROM or any other memory.
    - dataSize_t
      - size of data to be sent, could be a uint32_t
- Common_Macros Module:
  - API-Functions: (mentioned below are just a few examples, feel free to add what's needed)
    - SetBit(Port, Pin)
      - Function-like macro.
    - ClearBit(Port, Pin)
      - Function-like macro.
    - ToggleBit(Port, Pin)
      - Function-like macro.
    - RotateRight(Port, n)
      - Function-like macro.
    - RotateLeft(Port, n)
      - Function-like macro.
- Timer Module:
  - API-Function:
    - Timer_initState Timer_init(Timer_config_ptr_t);
      - Reentrant, Asynchronous, Non-Recursive Function.
    - void Timer_Start();

        - Non-Reentrant, Synchronous, Non-Recursive Function.
    - void Timer_Stop();
        - Non-Reentrant, Synchronous, Non-Recursive Function.
    - void setCallback();
        - To set ISR function passed from app layer.
        - Non-Reentrant, Synchronous, Non-Recursive ISR function.
    - uint32_t Timer_getElapsedTime();
        - returns time from beginning of timer's counting.
        - Non-Reentrant, Synchronous, Non-Recursive Function.
    - uint32_t Timer_ getRemainingTime ();
        - returns remaining time from specified initial time to be counted.
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
    - void Timer_delay (uint32_t);
        - counts in milliseconds
        - Non-Reentrant, Synchronous, Non-Recursive Function.
    - void Timer_delayMicroseconds(uint32_t);
        - Non-Reentrant, Synchronous, Non-Recursive Function.
  - API-Types:
    - Timer_Length_t
        - Normal or wide, 16 or 32 bit depending on architecture and available timers.
    - Timer_Mode_t
        - Normal, One-shot…etc.
    - Timer_config_ptr_t
        - To specify certain configurations for timer/s used.
- CAN Module:
  - API-Functions
    - CAN_initState CAN_init(CAN_config_ptr_t);
        - To initialize CAN Protocol.
        - Reentrant, Asynchronous, Non-Recursive Function.
    - void CAN_Send(uint32_t data, dataSize_t);

        - Non-Reentrant, Synchronous, Non-Recursive Function.
    - void CAN_Receive();
        - Non-Reentrant, Synchronous, Non-Recursive Function.
- ➢ API-Types:
    - CAN_sendState
        - Either sending success or fail.
    - CAN_receiveState
        - Either receiving success or fail.
    - CAN_initState
        - Initialization success or fail.
    - CAN_config_ptr_t

❖ DIO Module:

- ➢ API-Function:
    - Dio_initState DIO_init (DIO_config_ptr_t);
        - To initialize different DIOs as input/output, ADC/Normal…etc.
        - Reentrant, Asynchronous, Non-Recursive Function.
    - uint8_t DIO_write (Port_t, Pin_t, LevelType_t);
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
    - uint8_t DIO_toggle (Port_t, Pin_t);
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
    - uint8_t DIO_read (Port_t, Pin_t);
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
    - Uint32_t DIO_readPort(Port_t, Pin_t);
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
    - Uint32_t DIO_writePort(Port_t);
        - Non-Reentrant, Asynchronous, Non-Recursive Function.
- ➢ API-Types:
    - Port_t: struct for Ports depending on microcontroller used.
    - Pin_t: struct for pins from 0 to 8/16/32 depending on architecture.
    - LevelType_t: either high or low.

        - Direction_t: input or output.
        - AttachType_t: PULL_UP, PULL_DOWN, OPEN_DRAIN
        - PinType_t: ADC, Normal…etc
        - Current_t: how much can a pin withstand current, as in Arm 2,4,8mA
        - Dio_config_ptr_t
        - Dio_initState
            - Either initialization success or fail.

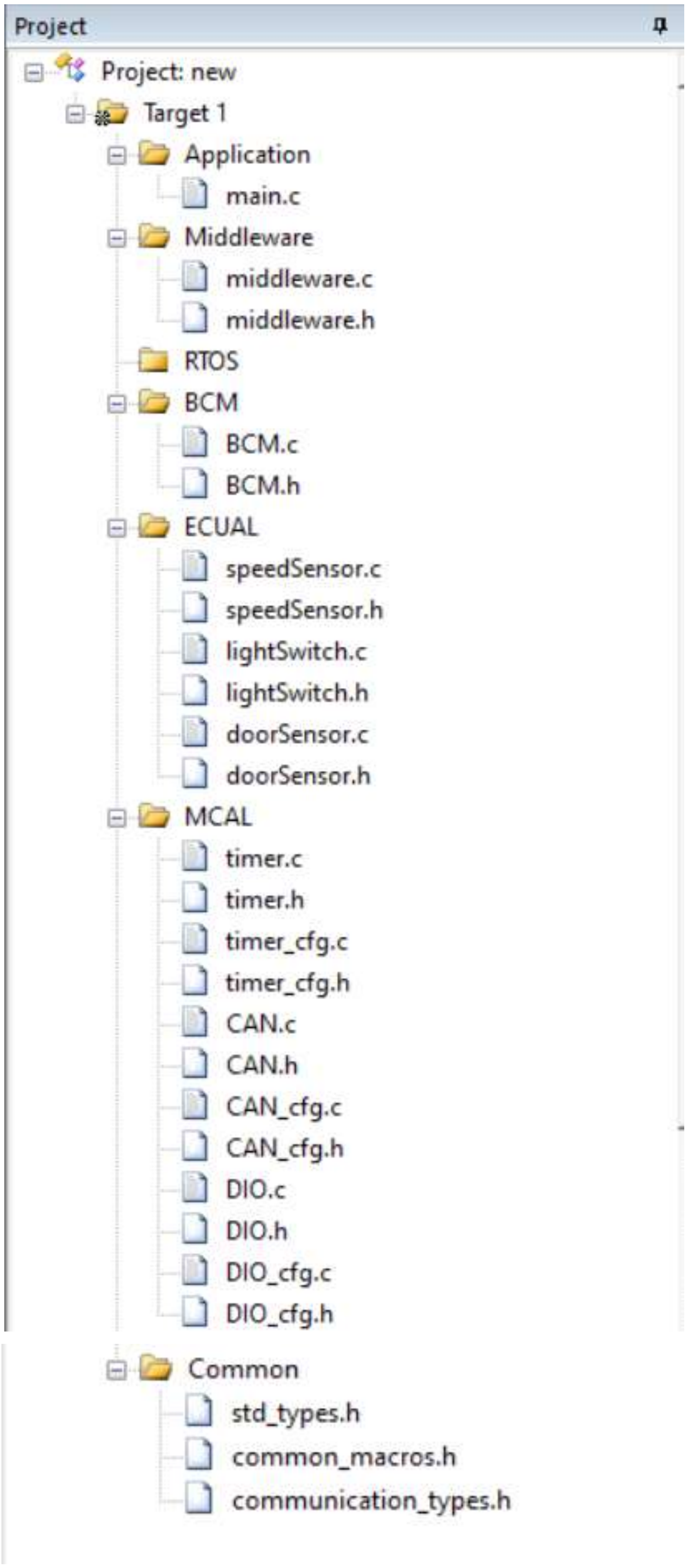
*PREVIOUS MODULES ARE SAME FOR BOTH ECUs, NEXT 3 INPUT MODULES ARE WHAT'S DIFFERENT

- Door Sensor Module:
    - API-Function:
        - DoorSensor_initState DoorSensor_init(Port_t, Pin_t);
            - Initialize it as input pin.
            - Reentrant, Asynchronous, Non-Recursive Function.
        - LevelType_t DoorState(Port_t, Pin_t);
            - Returns either door opened/closed with corresponding pin level.
            - Non-Reentrant, Synchronous, Non-Recursive Function.
    - API-Types:
        - Door_initState
            - Either initialization success or fail.
- Light Switch Module:
    - API-Function:
        - LightSwitch_initState LightSwitch_init(Port_T, Pin_t);
            - Reentrant, Asynchronous, Non-Recursive Function.
        - LevelType_t LightSwitchToggle (Port_T, Pin_t);
            - Non-Reentrant, Synchronous, Non-Recursive Function.
        - LevelType_t LightSwitchState (Port_t, Pin_t);
            - Non-Reentrant, Synchronous, Non-Recursive Function.
    - API-Types:
        - LightSwitch_initState
            - Either initialization success or fail.

- Speed Sensor Module:
  - API-Function:
    - SpeedSensor_initState SpeedSensorInit(Port_t, Pin_t);
      - To initialize speed sensor pin as input.
      - Reentrant, Asynchronous, Non-Recursive Function.
    - uint32_t SpeedSensorValue (void);
      - returns speed value read from speed sensor.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
  - API-Types:
    - SpeedSensor_initState
      - Either initialization success or fail.

4) Prepare your folder structure according to the previous points

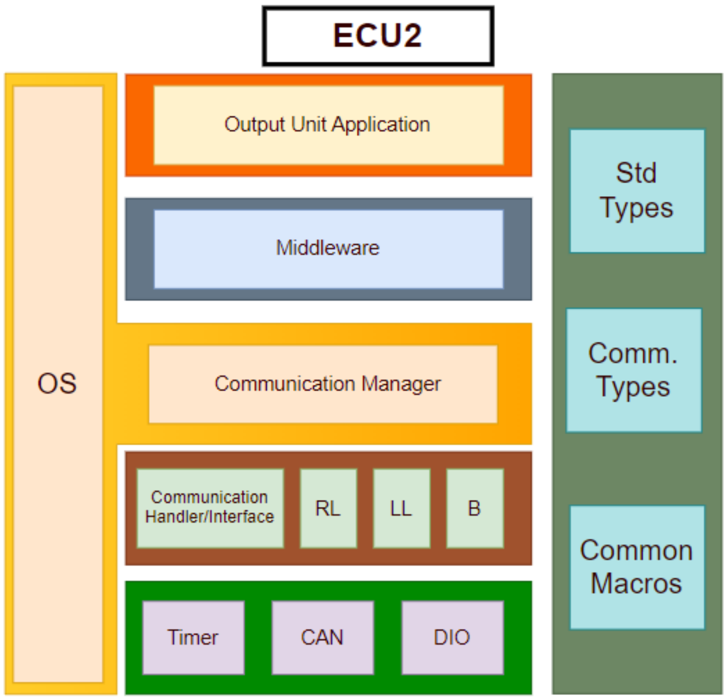
- Screenshot from Keil



```
Project
⊟ Project: new
  ⊟ Target 1
    ⊟ Application
        main.c
    ⊟ Middleware
        middleware.c
        middleware.h
      RTOS
    ⊟ BCM
        BCM.c
        BCM.h
    ⊟ ECUAL
        speedSensor.c
        speedSensor.h
        lightSwitch.c
        lightSwitch.h
        doorSensor.c
        doorSensor.h
    ⊟ MCAL
        timer.c
        timer.h
        timer_cfg.c
        timer_cfg.h
        CAN.c
        CAN.h
        CAN_cfg.c
        CAN_cfg.h
        DIO.c
        DIO.h
        DIO_cfg.c
        DIO_cfg.h
    ⊟ Common
        std_types.h
        common_macros.h
        communication_types.h
```

For ECU 2:

1. Make the layered architecture



2. Specify ECU components and modules
    1. Application: Where system logic and tasks are implemented.
    2. Middleware: For routing the system to its desired destination.
    3. RTOS: Handling tasks and operating system as stated in project rubric.
    4. Common: For standard data types and common macros.
    5. HAL:
        1. Onboard/ECUAL: containing Right Light , Left Light, Buzzer and all external output devices.
        2. MCAL: containing Timer, CAN and DIO modules.
3. Provide full detailed APIs for each module as well as a detailed description for the used typedefs
    - ❖ Application Module
    - ❖ Middleware Module
    - ❖ BCM Module
    - ❖ Std_Types Module
    - ❖ Common_Macros Module
    - ❖ Timer Module
    - ❖ CAN Module
    - ❖ DIO Module

*PREVIOUS MODULES ARE SAME FOR BOTH ECUs, NEXT 3 OUTPUT MODULES ARE WHAT'S DIFFERENT

- Right Light Module:
  - API-Function:
    - RightLight _initState RightLight _init(Port_t, Pin_t);
      - Sets light as output.
      - Reentrant, Asynchronous, Non-Recursive Function.
    - Boolean_t RightLightOn(Port_t, Pin_t);
      - Turns light on.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
    - Boolean _t RightLightOff(Port_t, Pin_t);
      - Turns light off.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
    - LevelType_t RightLightState(Port_t, Pin_t);
      - Returns light state either on or off according to corresponding level.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
  - API-Types:
    - RightLight_initState
      - Either initialization success or fail.
- Left Light Module:
  - API-Function:
    - LeftLight_initState LeftLight_init(Port_t, Pin_t);
      - Sets light as output.
      - Reentrant, Asynchronous, Non-Recursive Function.
    - Boolean_t LeftLightOn (Port_t, Pin_t);
      - Turns light on.
      - Non-Reentrant, Synchronous, Non-Recursive Function.
    - Boolean _t LeftLightOff(Port_t, Pin_t);
      - Turns light off.
      - Non-Reentrant, Synchronous, Non-Recursive Function.

- LevelType_t LeftLightState(Port_t, Pin_t);
  - Returns light state either on or off according to corresponding level.
  - Non-Reentrant, Synchronous, Non-Recursive Function.

➢ API-Types:

- LeftLight_initState
  - Either initialization success or fail.

❖ Buzzer Module:

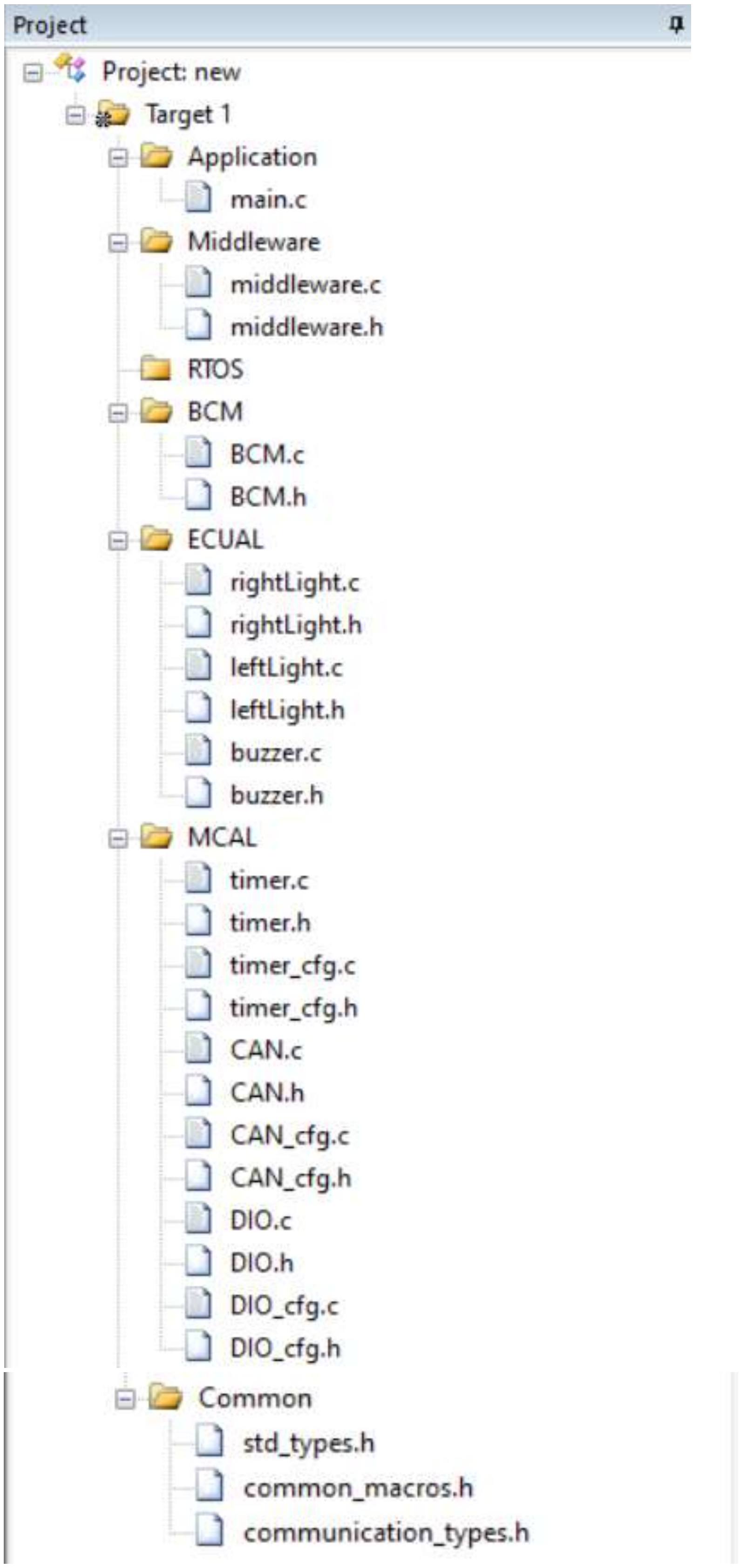
➢ API-Function:

- Buzzer_initState Buzzer_init(Port_t, Pin_t);
  - Sets buzzer pin as output.
  - Reentrant, Asynchronous, Non-Recursive Function.
- Boolean_t Buzzer_On (void);
  - Turns buzzer on and returns Boolean value either action was taken or not.
  - Non-Reentrant, Synchronous, Non-Recursive Function.
- Boolean_t Buzzer_Off (void);
  - Turns buzzer off and returns Boolean value either action was taken or not.
  - Non-Reentrant, Synchronous, Non-Recursive Function.

➢ API-Types:

- Buzzer_initState
  - Either initialization success or fail.

4. Prepare your folder structure according to the previous points



```
Project
Project: new
  Target 1
    Application
      main.c
    Middleware
      middleware.c
      middleware.h
    RTOS
    BCM
      BCM.c
      BCM.h
    ECUAL
      rightLight.c
      rightLight.h
      leftLight.c
      leftLight.h
      buzzer.c
      buzzer.h
    MCAL
      timer.c
      timer.h
      timer_cfg.c
      timer_cfg.h
      CAN.c
      CAN.h
      CAN_cfg.c
      CAN_cfg.h
      DIO.c
      DIO.h
      DIO_cfg.c
      DIO_cfg.h
    Common
      std_types.h
      common_macros.h
      communication_types.h
```