Moving Car Design

Designed by:

Ahmed Mostafa Mahmoud Ahmed Mohamed

Table of Contents

1-INTRODUCTION

The development of a moving car that seamlessly mixes hardware and software components is an interesting and difficult task in the field of embedded systems and digital design. In this project, four motors are controlled with precision and effectiveness by the ATmega32 microcontroller and Timer1.

In this project, we'll examine the ATmega32's architecture, look into Timer1's features, and create a control scheme to guarantee the synchronized operation of the four motors. This project provides a fulfilling trip into the realm of embedded systems and digital design, whether it's the excitement of programming the microcontroller or the gratification of watching the automobile move in reaction to your code.

The system will consist of the following components :

- Four motors (M1, M2, M3, M4)
- One button to start (PB1)
- One button for stop (PB2)
- Four LEDs (LED1, LED2, LED3, LED4)

The system will require to follow these procedures:

1. The car starts initially from 0 speed.
2. When PB1 is pressed, the car will move forward after 1 second.
3. The car will move forward to create the longest side of the rectangle for 3 seconds with 50% of its maximum speed.
4. After finishing the first longest side the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second.
5. The car will move to create the short side of the rectangle at 30% of its speed for 2 seconds.
6. After finishing the shortest side, the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second.
7. Steps 3 to 6 will be repeated infinitely until you press the stop button (PB2).
8. PB2 acts as a sudden break, and it has the highest priority.
9. LEDs Operations

   A. LED1: On means moving forward on the long side.

B. LED2: On means moving forward on the short side.
C. LED3: On means stop.
D. LED4: On means Rotating.

2-HIGH-LEVEL DESIGN

2.1-Layered Architecture

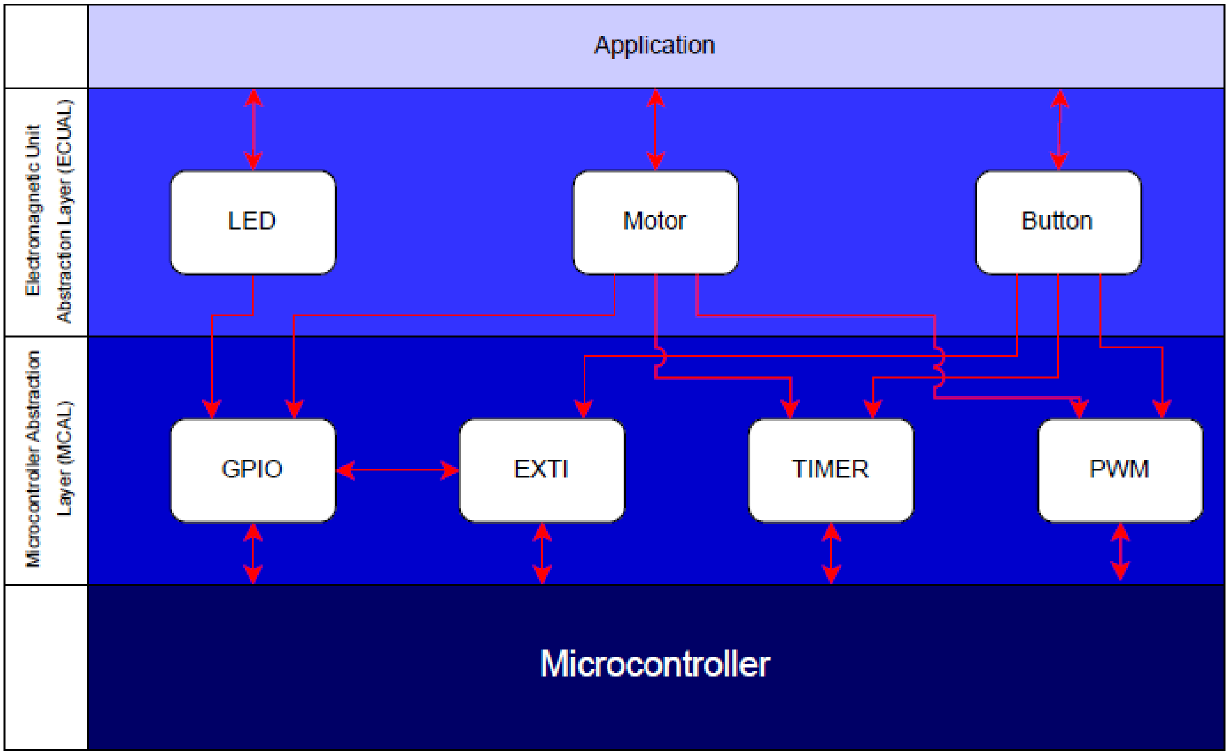

Application
Electromagnetic Unit Abstraction Layer (ECUAL)
LED
Motor
Button
Microcontroller Abstraction Layer (MCAL)
GPIO
EXTI
TIMER
PWM
Microcontroller

Figure 1: Layered Architecture

2.2-Modules Description

2.2.1-LED

An easy and organized solution to manage LEDs in the systems is provided by this LED module. It simplifies the control of LEDs in the application by abstracting the low-level hardware interactions. It consists of 4 methods: **ECUAL_led_init** for initializing the led, **ECUAL_led_on** for turning on the led, **ECUAL_led_off** for turning off the led, and **ECUAL_led_toggle** for toggling the led.

2.2.2-Button

A software component called the Button Module was created to simplify and abstract the functions of a physical button. The Application Programming Interfaces (APIs) provided by this module make it easy to interact with buttons without having to worry about intricate hardware specifics. This module consists of 3 methods: **ECUAL_button_init** for initializing the button, **ECUAL_button_read** for reading the input of the button, and **ECUAL_button_enable_EXTI** for making the button work as external interrupt.

2.2.3-Motor

The motor module was created to control all of the 4 motors in a simple way and made it easy to interact with movement of the car without getting into the detail. The motors are divided into two parts, right and left each containing 2 motors. The module consists of 7 methods: **ECUAL_motor_init** for initializing the motors, **ECUAL_motor_move_forward** configures the motors to move forward, **ECUAL_motor_move_backward** configures the motors to move backward, **ECUAL_motor_stop** makes the motors stop, **ECUAL_motor_move_all** makes the both sides move at the same speed, **ECUAL_motor_move_right** make the car move to the right by configuring the motors in a specific way, **ECUAL_motor_move_left** make the car move to the left by configuring the motors in a specific way.

2.2.4-DIO

The module was created to abstract and make it easier to manage the pins on microcontrollers. DIO pins are flexible because they may be set up as input or output and utilized for a variety of tasks, including reading sensors, operating peripherals, or connecting to other devices. Its major goal is to simplify DIO pin manipulation, encouraging code reuse and maintainability in embedded systems. It consists of 6 methods: **MCAL_dio_init** for initializing the pins, **MCAL_dio_write_pin** for writing on specific pin, **MCAL_dio_write_port** for writing on specific port, **MCAL_dio_toggle_pin** for toggling data on specific pin, **MCAL_dio_read_pin** for reading from a specific pin, and **MCAL_dio_read_port** for reading from a specific port.

2.2.5-EXTI

External interrupts are critical for responding to events from external sources, such as sensors, buttons, or other peripherals, without constantly polling their states. It consists of 1 method **MCAL_EXTI_init** for initializing the external interrupts.

2.2.6-Timer

Timers are crucial for a variety of purposes, such as creating exact time delays, calculating intervals, and managing recurring processes. The timer module consists of 2 methods **MCAL_timer_init** for initializing the timer according to the preferred working conditions, and **MCAL_timer_delay_ms** for creating a delay.

2.2.7-PWM

The PWM (Pulse Width Modulation) Module is designed to abstract and make it easier to generate and control PWM signals. PWM signals are frequently utilized for many different tasks, such as creating analog-like signals and adjusting LED brightness and motor speed. This module consists of 2 methods **MCAL_pwm_init** for initializing the pwm driver, and **MCAL_pwm_set_duty_cycle** for setting a duty cycle.

2.2.8-Application

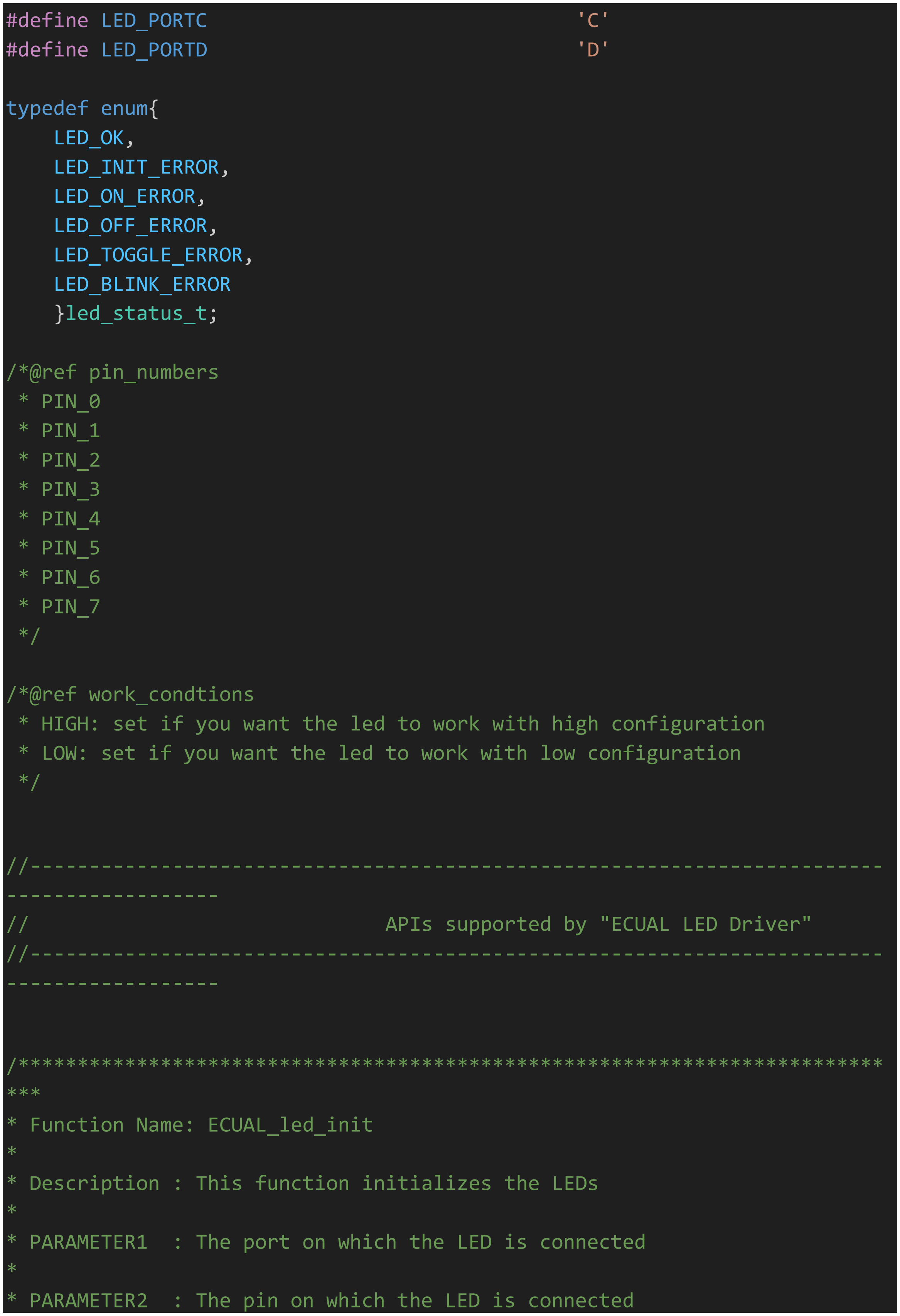
Here is where the magic happens. This module controls all the logic of the car moving system it is responsible for when to move, stop, rotate, etc. This module consists of 2 methods: **application_car_moving_system_init** initializes the entire system, and **application_car_moving_system** controls the system.

2.3- Drivers' documentation

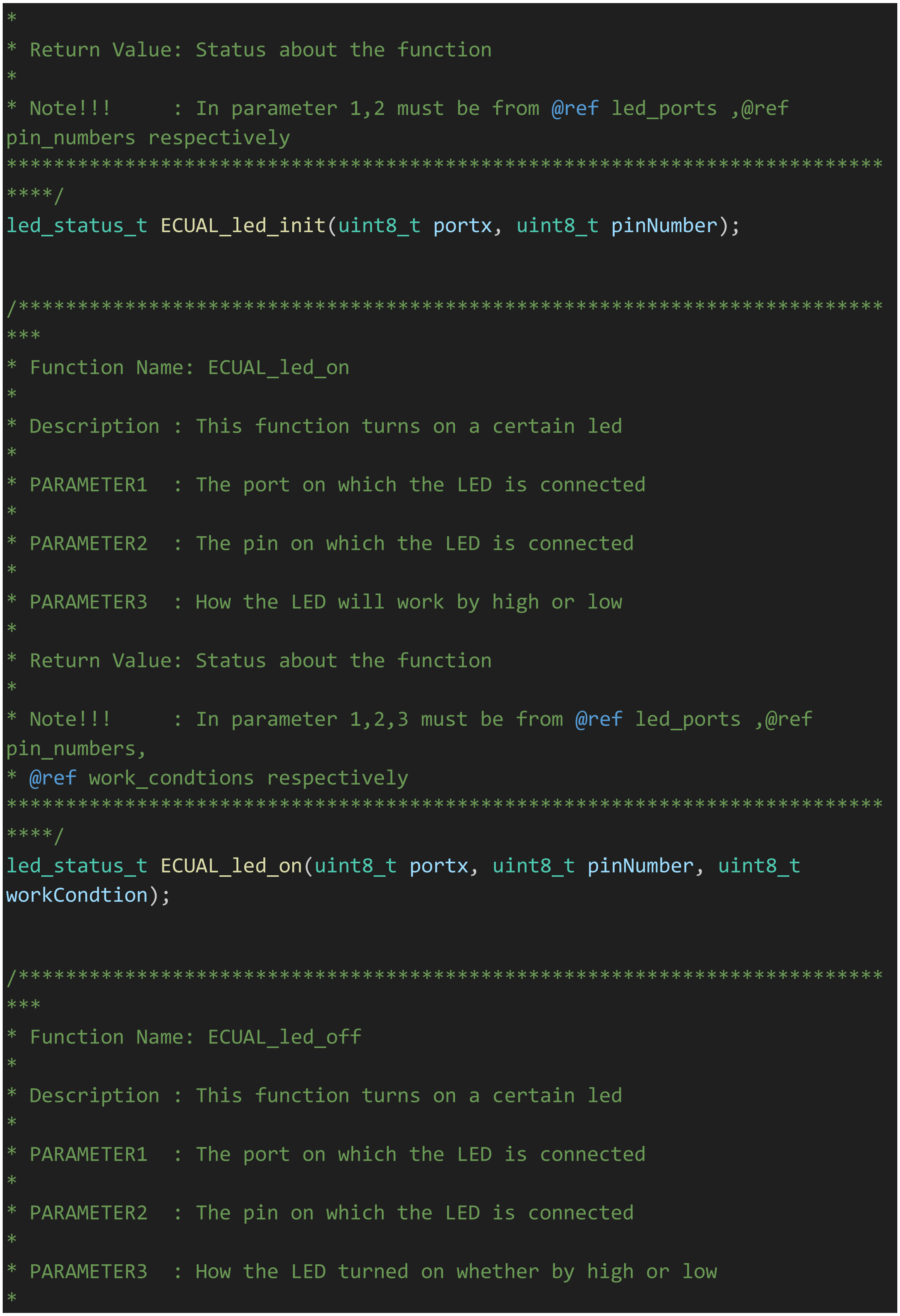
In this section I will provide each module header file it contains each API provided by the module and a brief description of how to use each API.

2.3.1-LED

```
/*
 * led.h
 *
 * Created: 26/08/2023 13:03:56
 *  Author: Ahmed
 */


#ifndef LED_H_
#define LED_H_


//--------------------------------
//          Includes
//--------------------------------


#include "../../INC/platform_types.h"
#include "../../MCAL/DIO/dio.h"
#include "../../MCAL/TIMER/timer.h"



//--------------------------------
//Macros Configuration References
//--------------------------------

//@ref led_ports
#define LED_PORTA                               'A'
#define LED_PORTB                               'B'
```



```
#define LED_PORTC                               'C'
#define LED_PORTD                               'D'

typedef enum{
    LED_OK,
    LED_INIT_ERROR,
    LED_ON_ERROR,
    LED_OFF_ERROR,
    LED_TOGGLE_ERROR,
    LED_BLINK_ERROR
    }led_status_t;

/*@ref pin_numbers
 * PIN_0
 * PIN_1
 * PIN_2
 * PIN_3
 * PIN_4
 * PIN_5
 * PIN_6
 * PIN_7
 */

/*@ref work_condtions
 * HIGH: set if you want the led to work with high configuration
 * LOW: set if you want the led to work with low configuration
 */


//------------------------------------------------------------------------
------------------
//                              APIs supported by "ECUAL LED Driver"
//------------------------------------------------------------------------
------------------


/*************************************************************************
***
* Function Name: ECUAL_led_init
*
* Description : This function initializes the LEDs
*
* PARAMETER1  : The port on which the LED is connected
*
* PARAMETER2  : The pin on which the LED is connected
```

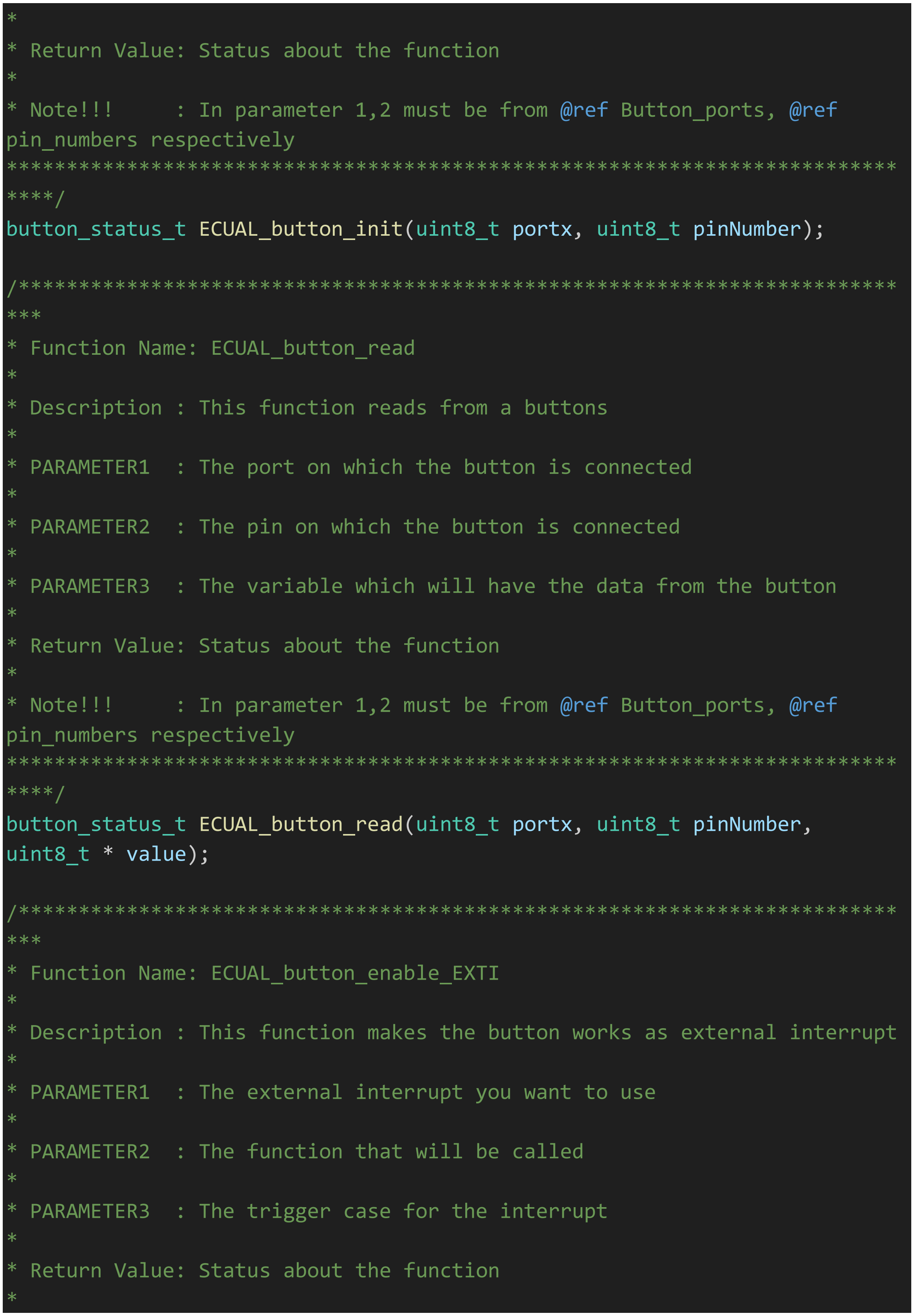


```
*
* Return Value: Status about the function
*
* Note!!!     : In parameter 1,2 must be from @ref led_ports ,@ref
pin_numbers respectively
**************************************************************************
****/
led_status_t ECUAL_led_init(uint8_t portx, uint8_t pinNumber);


/*************************************************************************
***
* Function Name: ECUAL_led_on
*
* Description : This function turns on a certain led
*
* PARAMETER1  : The port on which the LED is connected
*
* PARAMETER2  : The pin on which the LED is connected
*
* PARAMETER3  : How the LED will work by high or low
*
* Return Value: Status about the function
*
* Note!!!     : In parameter 1,2,3 must be from @ref led_ports ,@ref
pin_numbers,
* @ref work_condtions respectively
**************************************************************************
****/
led_status_t ECUAL_led_on(uint8_t portx, uint8_t pinNumber, uint8_t
workCondtion);


/*************************************************************************
***
* Function Name: ECUAL_led_off
*
* Description : This function turns on a certain led
*
* PARAMETER1  : The port on which the LED is connected
*
* PARAMETER2  : The pin on which the LED is connected
*
* PARAMETER3  : How the LED turned on whether by high or low
*
```

```
* Return Value: Status about the function
*
* Note!!!      : In parameter 1,2,3 must be from @ref led_ports ,@ref
pin_numbers,
* @ref work_condtions respectively
**************************************************************************
****/
led_status_t ECUAL_led_off(uint8_t portx, uint8_t pinNumber, uint8_t
workCondtion);


/*************************************************************************
***
* Function Name: ECUAL_led_toggle
*
* Description : This function toggles a certain led
*
* PARAMETER1   : The port on which the LED is connected
*
* PARAMETER2   : The pin on which the LED is connected
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 1,2 must be from @ref led_ports ,@ref
pin_numbers, respectively
**************************************************************************
****/
led_status_t ECUAL_led_toggle(uint8_t portx, uint8_t pinNumber);

#endif /* LED_H_ */
```

2.3.2-Button

```
/*
 * button.h
 *
 * Created: 26/08/2023 15:02:19
 *  Author: Ahmed
 */


#ifndef BUTTON_H_
#define BUTTON_H_

//--------------------------------
//          Includes
//--------------------------------


#include "../../INC/platform_types.h"
#include "../../MCAL/DIO/dio.h"
#include "../../MCAL/EXTERNAL_INTERRUPT/external_interrupt.h"


//--------------------------------
//Macros Configuration References
//--------------------------------

#define BUTTON_SET                                  1
#define BUTTON_RESET                                0

//@ref Button_ports
#define BUTTON_PORTA                                'A'
#define BUTTON_PORTB                                'B'
#define BUTTON_PORTC                                'C'
#define BUTTON_PORTD                                'D'

typedef enum{
    BUTTON_OK,
    BUTTON_INIT_ERROR,
    BUTTON_READ_ERROR,
    BUTTON_EXTI_ERROR
}button_status_t;


/*@ref pin_numbers
 * PIN_0
 * PIN_1
 * PIN_2
 * PIN_3
 * PIN_4
 * PIN_5
 * PIN_6
 * PIN_7
 */


/*
 * For @ref EXTI_Number use these macros:
 * EXTI0
 * EXTI1
 * EXTI2
 */

/*
 * For @ref Trigger_Cases use these macros:
 * This for INT0 & INT1 ONLY!!!
 * TRIGGER_CASE_LOW_LEVEL
 * TRIGGER_CASE_ANY_CHANGE
 * TRIGGER_CASE_FALLING_EDGE
 * TRIGGER_CASE_RISING_EDGE

 * This for INT2 ONLY!!!
 * TRIGGER_INT2_CASE_FALLING_EDGE
 * TRIGGER_INT2_CASE_RISING_EDGE
 */

//-------------------------------------------------------------------------
------------------
//                              APIs supported by "ECUAL Button Driver"
//-------------------------------------------------------------------------
------------------

/*************************************************************************
***
* Function Name: ECUAL_button_init
*
* Description : This function initializes the buttons
*
* PARAMETER1  : The port on which the button is connected
*
* PARAMETER2  : The pin on which the button is connected
```

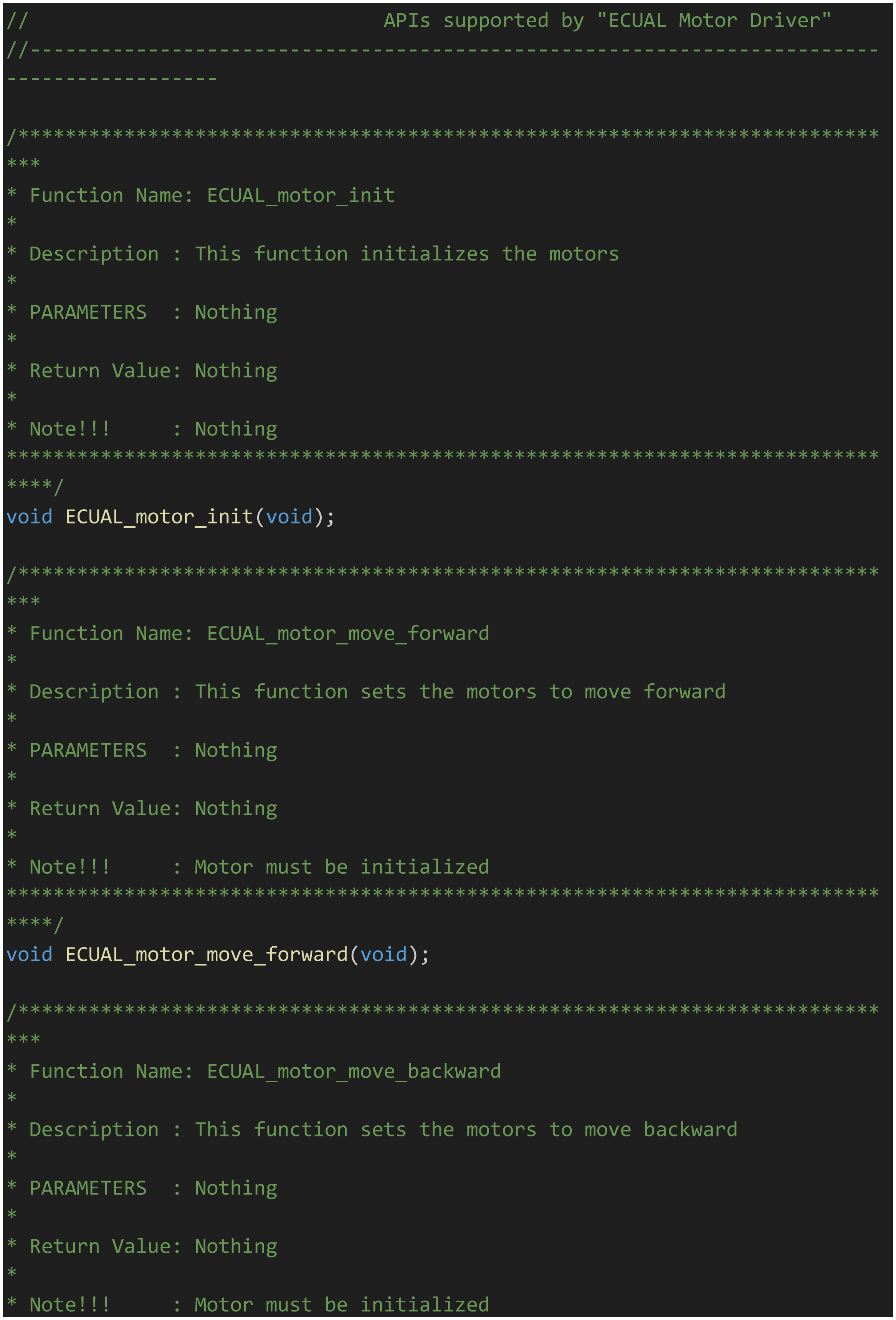


```
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 1,2 must be from @ref Button_ports, @ref
pin_numbers respectively
**************************************************************************
****/
button_status_t ECUAL_button_init(uint8_t portx, uint8_t pinNumber);

/*************************************************************************
***
* Function Name: ECUAL_button_read
*
* Description : This function reads from a buttons
*
* PARAMETER1   : The port on which the button is connected
*
* PARAMETER2   : The pin on which the button is connected
*
* PARAMETER3   : The variable which will have the data from the button
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 1,2 must be from @ref Button_ports, @ref
pin_numbers respectively
**************************************************************************
****/
button_status_t ECUAL_button_read(uint8_t portx, uint8_t pinNumber,
uint8_t * value);

/*************************************************************************
***
* Function Name: ECUAL_button_enable_EXTI
*
* Description : This function makes the button works as external interrupt
*
* PARAMETER1   : The external interrupt you want to use
*
* PARAMETER2   : The function that will be called
*
* PARAMETER3   : The trigger case for the interrupt
*
* Return Value: Status about the function
*
```

```
* Note!!!     : In parameter 1,3 must be from @ref EXTI_Number, @ref
Trigger_Cases respectively
**************************************************************************
****/
button_status_t ECUAL_button_enable_EXTI(uint8_t EXTINumber, void (*
callback)(void), uint8_t triggerCase);

#endif /* BUTTON_H_ */
```

2.3.3-Motor

```
/*
 * motor.h
 *
 * Created: 18/09/2023 21:49:53
 *  Author: Ahmed
 */


#ifndef MOTOR_H_
#define MOTOR_H_

//------------------------------------------------------------------------
------------------
//                                       Includes
//------------------------------------------------------------------------
------------------

#include "../../MCAL/DIO/dio.h"
#include "../../MCAL/TIMER/timer.h"
#include "../../MCAL/PWM/pwm.h"

//------------------------------------------------------------------------
------------------
//                                       General Macros
//------------------------------------------------------------------------
------------------

#define MOTOR_PORT                      'a'
#define MOTOR_R_L_PORT                  'b'
#define MOTOR_S1M_PIN                   PIN_6
#define MOTOR_S2M_PIN                   PIN_7

//------------------------------------------------------------------------
------------------
```



```
//                              APIs supported by "ECUAL Motor Driver"
//------------------------------------------------------------------------
------------------

/*************************************************************************
***
* Function Name: ECUAL_motor_init
*
* Description : This function initializes the motors
*
* PARAMETERS  : Nothing
*
* Return Value: Nothing
*
* Note!!!     : Nothing
**************************************************************************
****/
void ECUAL_motor_init(void);

/*************************************************************************
***
* Function Name: ECUAL_motor_move_forward
*
* Description : This function sets the motors to move forward
*
* PARAMETERS  : Nothing
*
* Return Value: Nothing
*
* Note!!!     : Motor must be initialized
**************************************************************************
****/
void ECUAL_motor_move_forward(void);

/*************************************************************************
***
* Function Name: ECUAL_motor_move_backward
*
* Description : This function sets the motors to move backward
*
* PARAMETERS  : Nothing
*
* Return Value: Nothing
*
* Note!!!     : Motor must be initialized
```

```
**************************************************************************
****/
void ECUAL_motor_move_backward(void);

/*************************************************************************
***
* Function Name: ECUAL_motor_stop
*
* Description : This function sets the motors to stop moving move
*
* PARAMETERS  : Nothing
*
* Return Value: Nothing
*
* Note!!!     : Motor must be initialized
**************************************************************************
****/
void ECUAL_motor_stop(void);

/*************************************************************************
***
* Function Name: ECUAL_motor_move_all
*
* Description : This function makes the motors move according how they
were
*configured
*
* PARAMETER1  : The percentage of the speed a value between 0% and 100%
*
* Return Value: Nothing
*
* Note!!!     : Must configure how to move first by using
ECUAL_motor_move_backward
* of ECUAL_motor_move_forward functions
**************************************************************************
****/
void ECUAL_motor_move_all(uint8_t speed);

/*************************************************************************
***
* Function Name: ECUAL_motor_move_right
*
* Description : This function makes the motors move to the right
*
* PARAMETER1  : The percentage of the speed a value between 0% and 100%
```

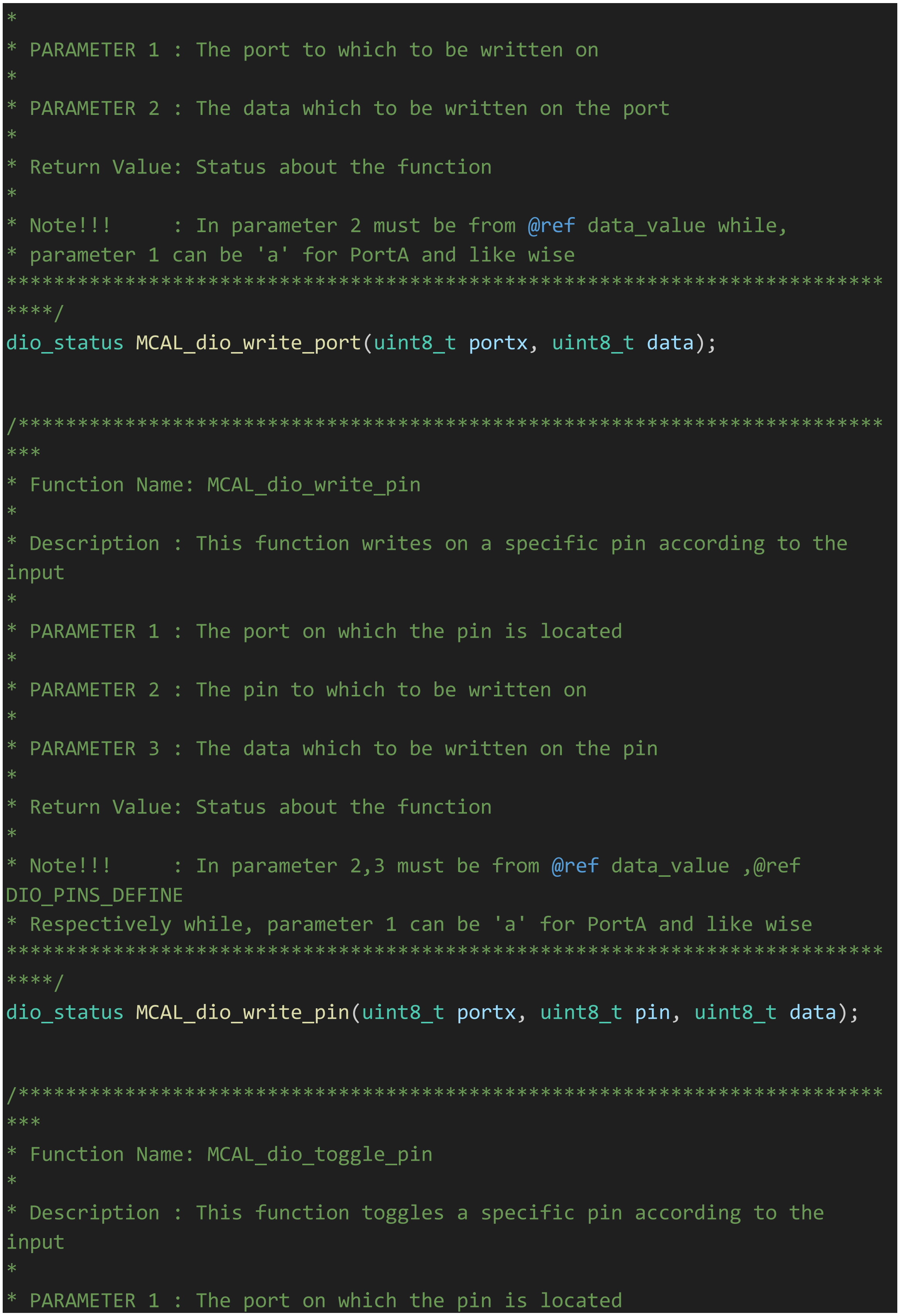
```c
*
* Return Value: Nothing
*
* Note!!!     : Must configure how to move first by using
ECUAL_motor_move_backward
* of ECUAL_motor_move_forward functions
**************************************************************************
****/
void ECUAL_motor_move_right(uint8_t speed);

/*************************************************************************
***
* Function Name: ECUAL_motor_move_left
*
* Description : This function makes the motors move to the left
*
* PARAMETER1  : The percentage of the speed a value between 0% and 100%
*
* Return Value: Nothing
*
* Note!!!     : Must configure how to move first by using
ECUAL_motor_move_backward
* of ECUAL_motor_move_forward functions
**************************************************************************
****/
void ECUAL_motor_move_left(uint8_t speed);




#endif /* MOTOR_H_ */
```

2.3.4-DIO

```c
/*
 * DIO.h
 *
 * Created: 25/08/2023 20:24:14
 *  Author: Ahmed
 */
#ifndef DIO_H_
#define DIO_H_

//--------------------------------
//          Includes
```

```c
//--------------------------------

#include "../../INC/platform_types.h"
#include "../../INC/memory_map.h"

//--------------------------------
//Macros Configuration References
//--------------------------------

typedef struct{
    uint8_t             pinNumber;          /*Specifies the DIO pins to be
configured
                                this parameter must be a value of @ref
DIO_PINS_DEFINE*/

    uint8_t             dioDirection;       /*Specifies the operating mode for
the selected pins
                               this parameter must be a value of @ref
DIO_MODE_DEFINE*/

    uint8_t             portOptionEnable;   /*Specifies the if you want to
configure the whole port. Note enabling this
                               will disregard the pinNumber this parameter
mus be a value of @ref PORT_OPTION*/
    }pin_config_t;

//@ref DIO_PINS_DEFINE
#define PIN_0                   ((uint8_t)0x01) // Pin 0 with shifting
#define PIN_1                   ((uint8_t)0x02) // Pin 1 with shifting
#define PIN_2                   ((uint8_t)0x04) // Pin 2 with shifting
#define PIN_3                   ((uint8_t)0x08) // Pin 3 with shifting
#define PIN_4                   ((uint8_t)0x10) // Pin 4 with shifting
#define PIN_5                   ((uint8_t)0x20) // Pin 5 with shifting
#define PIN_6                   ((uint8_t)0x40) // Pin 6 with shifting
#define PIN_7                   ((uint8_t)0x80) // Pin 7 with shifting

//@ref DIO_MODE_DEFINE
#define INPUT_MODE                  0
#define OUTPUT_MODE                 1

//@ref PORT_OPTION
#define PORT_OPTION_ENABLE          0
#define PORT_OPTION_DISABLE         1

//Status definition
```

```
typedef enum{
    DIO_OK,
    DIO_INIT_ERROR,
    DIO_PORT_WRITE_ERROR,
    DIO_PIN_WRITE_ERROR,
    DIO_PIN_TOGGLE_ERROR,
    DIO_READ_PIN_ERROR,
    DIO_READ_PORT_ERROR
    }dio_status;

//@ref data_value
#define LOW                         0
#define HIGH                        1

//------------------------------------------------------------------------------------------
//                              APIs supported by "MCAL DIO Driver"
//------------------------------------------------------------------------------------------
/****************************************************************************
* Function Name: MCAL_dio_init
*
* Description : This function initialize the I/O of the microcontroller according
* to the configured data
*
* PARAMETER 1 : The port to which to be configured
*
* PARAMETER 2 : Structure with the preferred configuration
*
* Return Value: Status about the function
*
* Note        : In parameter 1 may write 'a' for PortA and like wise for the others ports
******************************************************************************/
dio_status MCAL_dio_init(uint8_t portx, pin_config_t * pinconfig);

/****************************************************************************
* Function Name: MCAL_dio_write_port
*
* Description : This function writes on the whole port according to the input
```



```
*
* PARAMETER 1 : The port to which to be written on
*
* PARAMETER 2 : The data which to be written on the port
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 2 must be from @ref data_value while,
* parameter 1 can be 'a' for PortA and like wise
**************************************************************************
****/
dio_status MCAL_dio_write_port(uint8_t portx, uint8_t data);


/*************************************************************************
***
* Function Name: MCAL_dio_write_pin
*
* Description : This function writes on a specific pin according to the
input
*
* PARAMETER 1 : The port on which the pin is located
*
* PARAMETER 2 : The pin to which to be written on
*
* PARAMETER 3 : The data which to be written on the pin
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 2,3 must be from @ref data_value ,@ref
DIO_PINS_DEFINE
* Respectively while, parameter 1 can be 'a' for PortA and like wise
**************************************************************************
****/
dio_status MCAL_dio_write_pin(uint8_t portx, uint8_t pin, uint8_t data);


/*************************************************************************
***
* Function Name: MCAL_dio_toggle_pin
*
* Description : This function toggles a specific pin according to the
input
*
* PARAMETER 1 : The port on which the pin is located
```

```
*
* PARAMETER 2 : The pin to which to be toggled
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 2 must be from @ref DIO_PINS_DEFINE while,
* parameter 1 can be 'a' for PortA and like wise
**************************************************************************
****/
dio_status MCAL_dio_toggle_pin(uint8_t portx, uint8_t pin);


/*************************************************************************
***
* Function Name: MCAL_dio_read_pin
*
* Description : This function reads data from a specific pin according to
the input
*
* PARAMETER 1 : The port on which the pin is located
*
* PARAMETER 2 : The pin to which to be read from
*
* PARAMETER 3 : The data which will be take the value from the pin
*
* Return Value: Status about the function
*
* Note!!!      : In parameter 2 must be from @ref DIO_PINS_DEFINE while,
* parameter 1 can be 'a' for PortA and like wise
**************************************************************************
****/
dio_status MCAL_dio_read_pin(uint8_t portx, uint8_t pin, uint8_t* value);


/*************************************************************************
***
* Function Name: MCAL_dio_read_port
*
* Description : This function reads from whole port according to the input
*
* PARAMETER 1 : The port to which the data will be read from
*
* PARAMETER 2 : The data which will have the value from the port
*
* Return Value: Status about the function
```

```
*
* Note!!!     : Parameter 1 can be 'a' for PortA and like wise
***************************************************************************
****/
dio_status MCAL_dio_read_port(uint8_t portx, uint8_t* value);


#endif /* DIO_H_ */
```

2.3.5-EXTI

```
/*
 * external_interrupt.h
 *
 * Created: 31/08/2023 13:27:46
 *  Author: Ahmed
 */


#ifndef EXTERNAL_INTERRUPT_H_
#define EXTERNAL_INTERRUPT_H_

//--------------------------------
//          Includes
//--------------------------------

#include "../../INC/platform_types.h"
#include "../../INC/memory_map.h"
#include "../../INC/utilities.h"
#include "../../MCAL/DIO/dio.h"
#include "../../INC/Interrupt.h"

//--------------------------------
//Macros Configuration References
//--------------------------------

typedef enum{
    EXTI_OK,
    EXTI_INIT_ERROR
    }EXTI_status;

typedef struct{
    uint8_t         extiEnable; /*Specifics whether to enable or disable
the interrupt
```

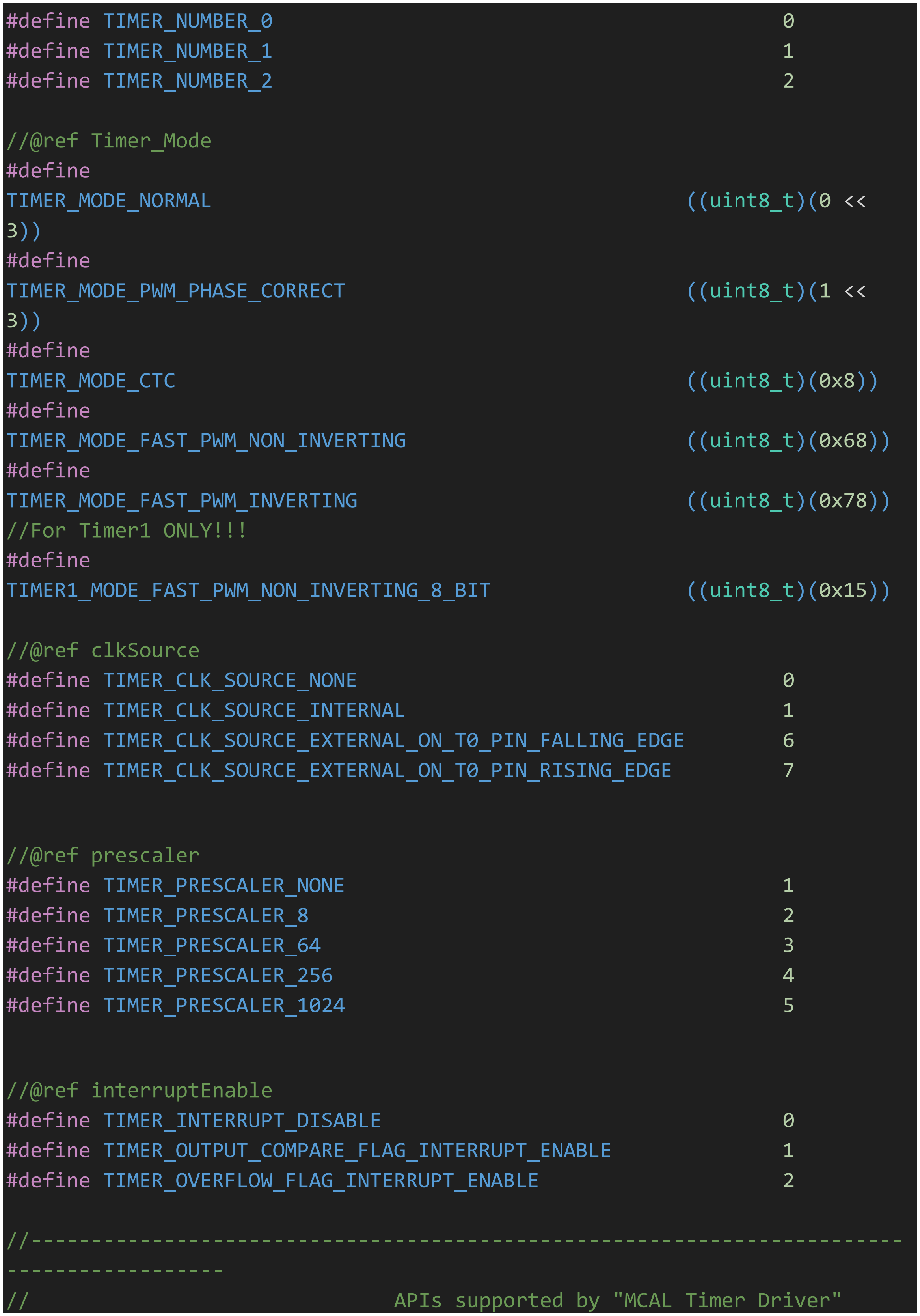
```
                                this parameter must be a value of @ref 
EXTI_Enable*/

    uint8_t         extiNumber; /*Specifies Which external interrupt to 
enable
                                this parameter must be a value of @ref 
EXTI_Number */

    uint8_t         triggerCase; /*Specifies the mode at which the 
interrupt will be trigged
                                this parameter must be a value of @ref 
Trigger_Cases*/

    void (* P_IRQ_callback)(void);  /*Set the C Function() which will be 
called once interrupt happens*/

    }exti_config_t;

//@ref EXTI_Enable
#define EXTI_DISABLE                            0
#define EXTI_ENABLE                             1


//@ref EXTI_Number
#define EXTI0                                   0
#define EXTI1                                   1
#define EXTI2                                   2


//@ref Trigger_Cases
/*
0 0 The low level generates an interrupt request.
0 1 Any logical change generates an interrupt request.
1 0 The falling edge generates an interrupt request.
1 1 The rising edge generates an interrupt request.
*/
//This for INT0 & INT1 ONLY!!!
#define TRIGGER_CASE_LOW_LEVEL                  0
#define TRIGGER_CASE_ANY_CHANGE                 1
#define TRIGGER_CASE_FALLING_EDGE               2
#define TRIGGER_CASE_RISING_EDGE                3

//This for INT2 ONLY!!!
#define TRIGGER_INT2_CASE_FALLING_EDGE          0
#define TRIGGER_INT2_CASE_RISING_EDGE           1
```

```
//---------------------------------------------------------------------------
------------------
//                              APIs supported by "MCAL EXTI Driver"
//---------------------------------------------------------------------------
------------------

/***************************************************************************
***
* Function Name: MCAL_EXTI_init
*
* Description : This function configures how the external interrupt will
work
*
* PARAMETER 1 : Structure with the preferred configuration
*
* Return Value: Status about the function
*
* Note!!!     : Nothing
****************************************************************************
****/
EXTI_status MCAL_EXTI_init(exti_config_t* config);



#endif /* EXTERNAL_INTERRUPT_H_ */
```

2.3.6-Timer

```
/*
 * timer.h
 *
 * Created: 08/09/2023 21:30:19
 *  Author: Ahmed
 */


#ifndef TIMER_H_
#define TIMER_H_

#include "../../INC/platform_types.h"
#include "../../INC/memory_map.h"
```

```c
#include "../../INC/interrupt.h"
#include "../../INC/utilities.h"
#include "../DIO/dio.h"



typedef struct{
    uint8_t                 timerNumber; /*Specifics which timer to program
this parameter
                                    must be a value of @ref Timer_Number*/

    uint8_t                 mode; /*Specifics the mode of which the timer will
work on
                                this parameter must be a value of @ref
Timer_Mode*/

    uint8_t                 clkSource; /*Specifics the clock source of the
timer whether
                                it is internal or external source this
parameter must be
                                 a value of @ref clkSource*/

    uint8_t                 prescaler; /*Specifics the prescaler value of
which the timer
                                will work on this parameter must be a
value of @ref prescaler*/

    uint8_t                 interruptEnable; /*Specifics whether to enable or
disable the interrupt
                                this parameter must be a value of @ref
interruptEnable*/

    void (* P_IRQ_callback)(void);  /*Set the C Function() which will be
called once interrupt happens*/

    }timer_config_t;


typedef enum{
    TIMER_OK,
    TIMER_INIT_ERROR,
    }timer_status_t;


//@ref Timer_Number
```



```
#define TIMER_NUMBER_0                                                  0
#define TIMER_NUMBER_1                                                  1
#define TIMER_NUMBER_2                                                  2

//@ref Timer_Mode
#define
TIMER_MODE_NORMAL                                        ((uint8_t)(0 <<
3))
#define
TIMER_MODE_PWM_PHASE_CORRECT                             ((uint8_t)(1 <<
3))
#define
TIMER_MODE_CTC                                           ((uint8_t)(0x8))
#define
TIMER_MODE_FAST_PWM_NON_INVERTING                        ((uint8_t)(0x68))
#define
TIMER_MODE_FAST_PWM_INVERTING                            ((uint8_t)(0x78))
//For Timer1 ONLY!!!
#define
TIMER1_MODE_FAST_PWM_NON_INVERTING_8_BIT                 ((uint8_t)(0x15))

//@ref clkSource
#define TIMER_CLK_SOURCE_NONE                                           0
#define TIMER_CLK_SOURCE_INTERNAL                                       1
#define TIMER_CLK_SOURCE_EXTERNAL_ON_T0_PIN_FALLING_EDGE                6
#define TIMER_CLK_SOURCE_EXTERNAL_ON_T0_PIN_RISING_EDGE                 7


//@ref prescaler
#define TIMER_PRESCALER_NONE                                            1
#define TIMER_PRESCALER_8                                               2
#define TIMER_PRESCALER_64                                              3
#define TIMER_PRESCALER_256                                             4
#define TIMER_PRESCALER_1024                                            5


//@ref interruptEnable
#define TIMER_INTERRUPT_DISABLE                                         0
#define TIMER_OUTPUT_COMPARE_FLAG_INTERRUPT_ENABLE                      1
#define TIMER_OVERFLOW_FLAG_INTERRUPT_ENABLE                            2

//---------------------------------------------------------------------------
------------------
//                               APIs supported by "MCAL Timer Driver"
```

```
//------------------------------------------------------------------------
------------------

/*************************************************************************
***
* Function Name: MCAL_timer_init
*
* Description : This function initializes a Timer to the preferred mode
*
* PARAMETER 1 : The timer which will be initialized
*
* Return Value: Status about the function
*
* Note!!!     : Nothing
**************************************************************************
****/
timer_status_t MCAL_timer_init(timer_config_t * config);

/*************************************************************************
***
* Function Name: MCAL_timer_delay_ms
*
* Description : This function creates a delay according to the input
*
* PARAMETER 1 : The delay you want but in milliseconds
*
* Return Value: Nothing
*
* Note!!!     : This function requires timer0 to work and timer0 in normal
mode
**************************************************************************
****/
void MCAL_timer_delay_ms(uint64_t delay);


#endif /* TIMER_H_ */
```

2.2.7-PWM

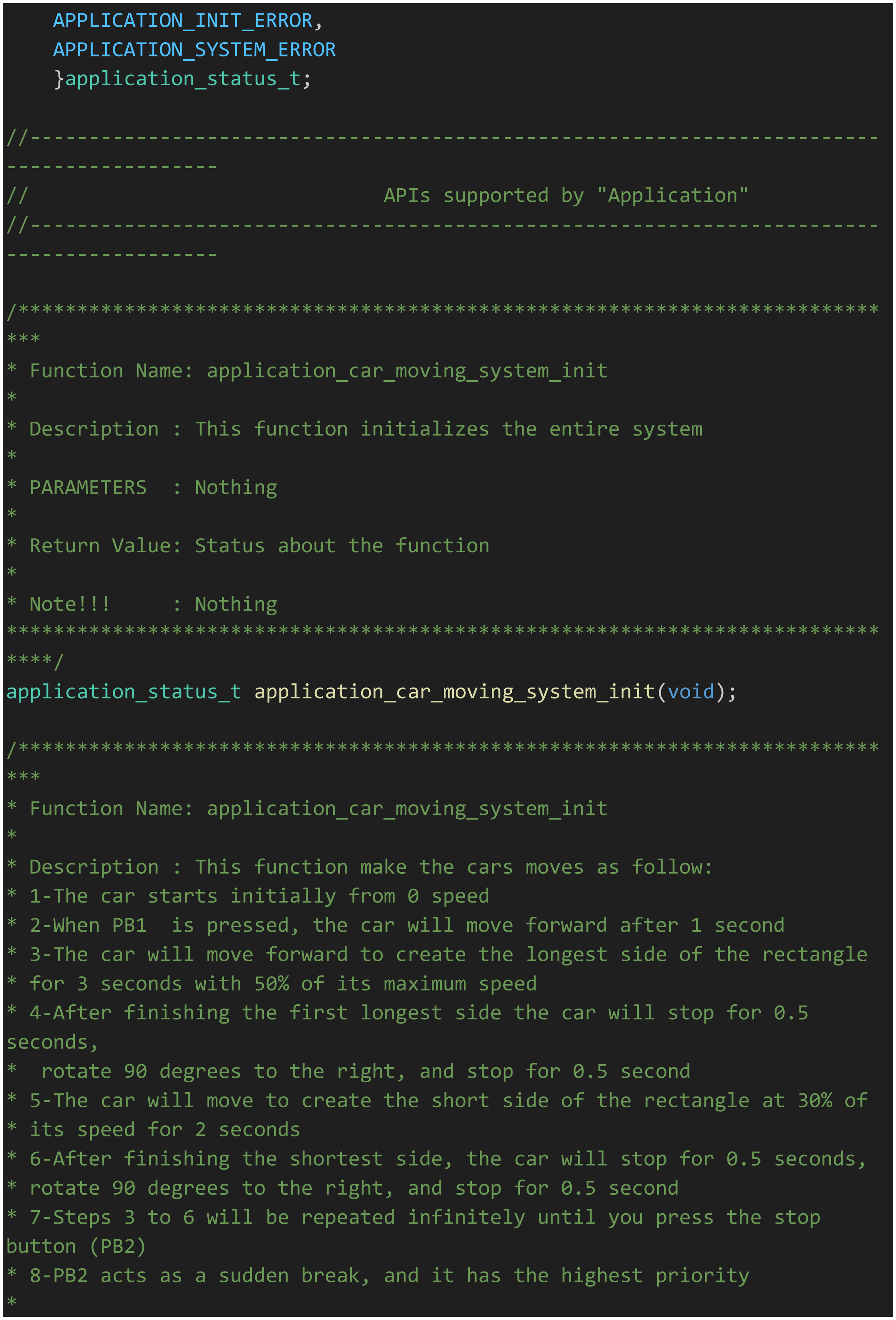
```
/*
 * pwm.h
 *
 * Created: 19/09/2023 20:56:04
 *  Author: Ahmed
 */


#ifndef PWM_H_
#define PWM_H_

#include "../../INC/platform_types.h"
#include "../../INC/memory_map.h"
#include "../../INC/utilities.h"
#include "../DIO/dio.h"
#include "avr/interrupt.h"



typedef struct{
    uint8_t             pwmNumber; /*Specifics which timer to program this
parameter
                                    must be a value of @ref Timer_Number*/

    uint8_t             clkSource; /*Specifics the clock source of the
timer whether
                                as it is internal or external source this
parameter must be
                                 a value of @ref clkSource*/

    uint8_t             prescaler; /*Specifics the prescaler value of
which the timer
                                will work on this parameter must be a
value of @ref prescaler*/

    }pwm_config_t;


typedef enum{
    PWM_OK,
    PWM_INIT_ERROR
    }pwm_status_t;
```

```
//@ref PWM_Number
#define PWM_NUMBER_0                                            0
#define PWM_NUMBER_1                                            1
#define PWM_NUMBER_2                                            2

//@ref clkSource
#define TIMER_CLK_SOURCE_NONE                                   0
#define TIMER_CLK_SOURCE_INTERNAL                               1
#define TIMER_CLK_SOURCE_EXTERNAL_ON_T0_PIN_FALLING_EDGE        6
#define TIMER_CLK_SOURCE_EXTERNAL_ON_T0_PIN_RISING_EDGE         7


//@ref prescaler
#define TIMER_PRESCALER_NONE                                    1
#define TIMER_PRESCALER_8                                       2
#define TIMER_PRESCALER_64                                      3
#define TIMER_PRESCALER_256                                     4
#define TIMER_PRESCALER_1024                                    5

//@ref PWM Ports
#define PWM0_PORT                                               'b'
#define PWM1_PORT                                               'd'
#define PWM2_PORT                                               'd'

//------------------------------------------------------------------------
------------------
//                              APIs supported by "MCAL PWM Driver"
//------------------------------------------------------------------------
------------------
/*************************************************************************
***
* Function Name: MCAL_pwm_init
*
* Description : This function initializes a Timer to work as like pwm mode
* using normal mode
*
* PARAMETER 1 : The timer which will be initialized
*
* Return Value: Status about the function
*
* Note        : Nothing
**************************************************************************
****/
pwm_status_t MCAL_pwm_init(pwm_config_t * config);
```

```
/*************************************************************************
***
* Function Name: MCAL_pwm_set_duty_cycle
*
* Description : Sets duty cycle for the PWM using normal mode
*
* PARAMETER 1 : The timer which will be used. It must be set from @ref
PWM_Number
*
* Return Value: Nothing
*
* Note!!!     : Must initialize the timer using MCAL_pwm_init function
**************************************************************************
****/
void MCAL_pwm_set_duty_cycle(uint8_t pwmX, uint32_t value);




#endif /* PWM_H_ */
```

2.2.8-Application

```
/*
 * application.h
 *
 * Created: 26/08/2023 16:06:52
 *  Author: Ahmed
 */


#ifndef APPLICATION_H_
#define APPLICATION_H_


//--------------------------------
//          Includes
//--------------------------------
#include "../ECUAL/LED/led.h"
#include "../ECUAL/BUTTON/button.h"
#include "../ECUAL/MOTOR/motor.h"
#include "../INC/platform_types.h"


typedef enum{
    APPLICATION_OK,
```



```
    APPLICATION_INIT_ERROR,
    APPLICATION_SYSTEM_ERROR
    }application_status_t;

//----------------------------------------------------------------------
------------------
//                              APIs supported by "Application"
//----------------------------------------------------------------------
------------------

/**********************************************************************
***
* Function Name: application_car_moving_system_init
*
* Description : This function initializes the entire system
*
* PARAMETERS  : Nothing
*
* Return Value: Status about the function
*
* Note!!!     : Nothing
***********************************************************************
****/
application_status_t application_car_moving_system_init(void);

/**********************************************************************
***
* Function Name: application_car_moving_system_init
*
* Description : This function make the cars moves as follow:
* 1-The car starts initially from 0 speed
* 2-When PB1  is pressed, the car will move forward after 1 second
* 3-The car will move forward to create the longest side of the rectangle
* for 3 seconds with 50% of its maximum speed
* 4-After finishing the first longest side the car will stop for 0.5
seconds,
*  rotate 90 degrees to the right, and stop for 0.5 second
* 5-The car will move to create the short side of the rectangle at 30% of
* its speed for 2 seconds
* 6-After finishing the shortest side, the car will stop for 0.5 seconds,
* rotate 90 degrees to the right, and stop for 0.5 second
* 7-Steps 3 to 6 will be repeated infinitely until you press the stop
button (PB2)
* 8-PB2 acts as a sudden break, and it has the highest priority
*
```

```
*
* PARAMETERS  : Nothing
*
* Return Value: Nothing
*
* Note!!!     : The system must be initialized before calling this
function
***************************************************************************
****/
void application_car_moving_system(void);


#endif /* APPLICATION_H_ */
```

3-LOW-LEVEL DESIGN

3.1-LED

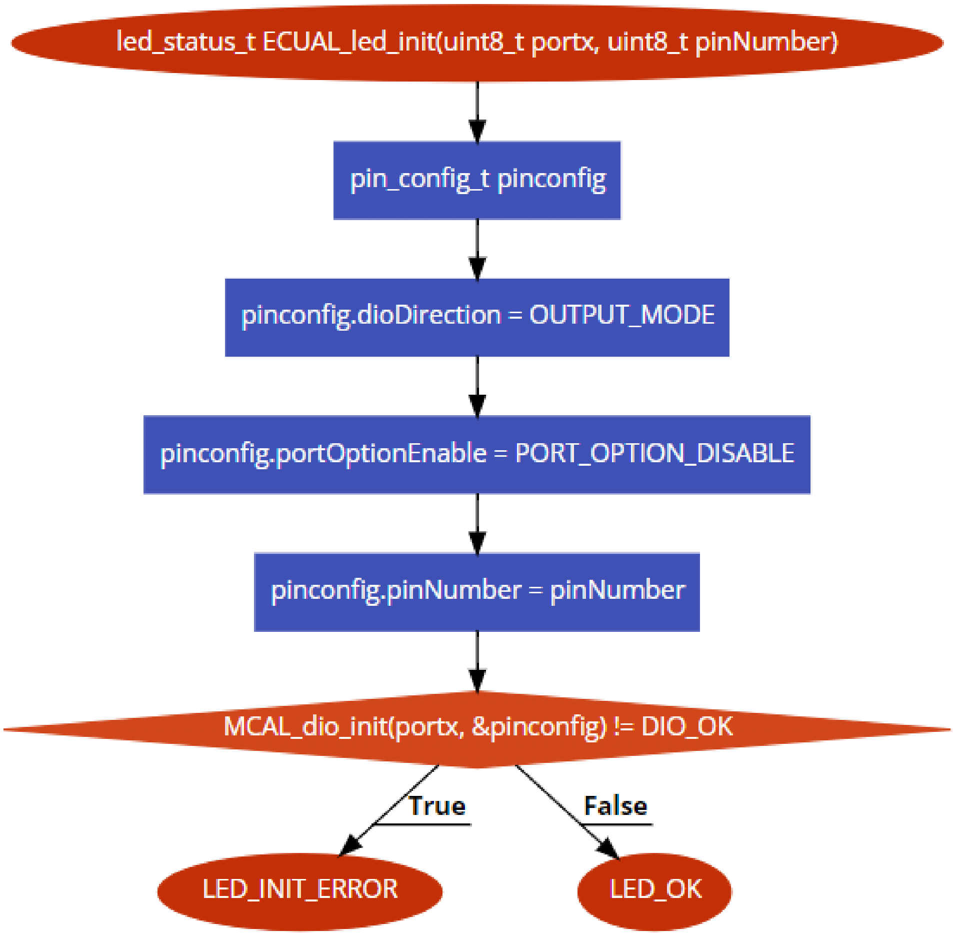

led_status_t ECUAL_led_init(uint8_t portx, uint8_t pinNumber)
pin_config_t pinconfig
pinconfig.dioDirection = OUTPUT_MODE
pinconfig.portOptionEnable = PORT_OPTION_DISABLE
pinconfig.pinNumber = pinNumber
MCAL_dio_init(portx, &pinconfig) != DIO_OK
True
False
LED_INIT_ERROR
LED_OK

Figure 2: ECUAL_led_init function flowchart.

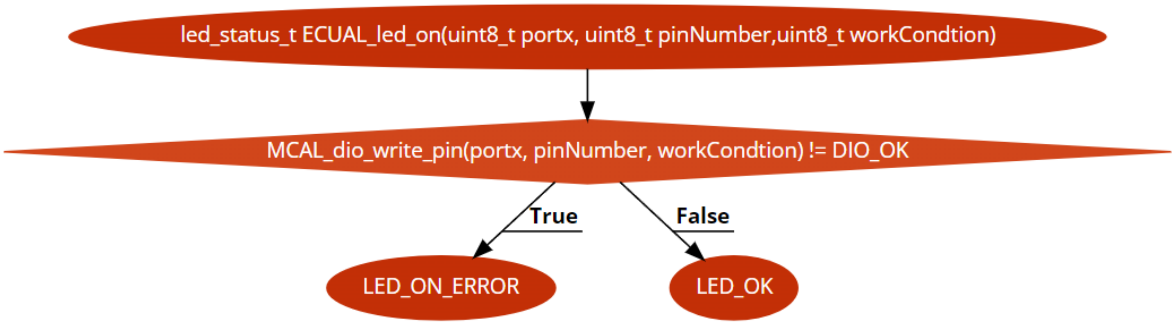

led_status_t ECUAL_led_on(uint8_t portx, uint8_t pinNumber,uint8_t workCondtion)
MCAL_dio_write_pin(portx, pinNumber, workCondtion) != DIO_OK
True
False
LED_ON_ERROR
LED_OK

Figure 3: ECUAL_led_on function flowchart.

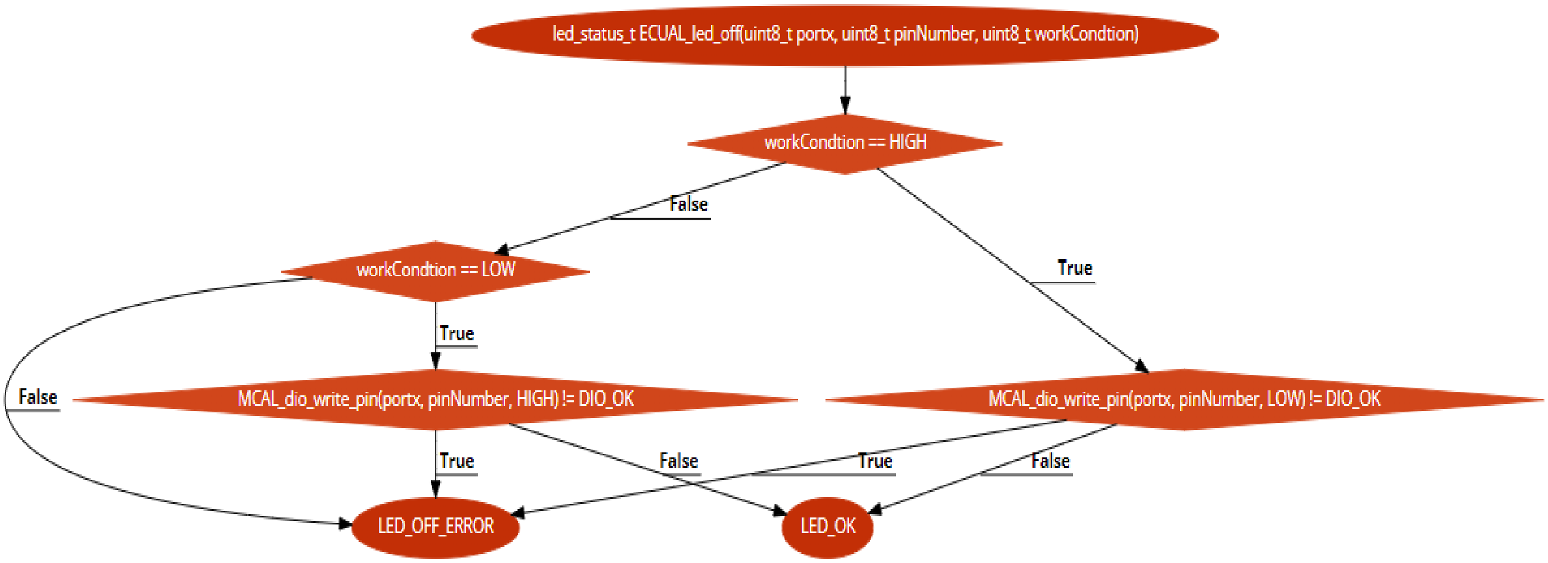

led_status_t ECUAL_led_off(uint8_t portx, uint8_t pinNumber, uint8_t workCondtion)
workCondtion == HIGH
False
True
workCondtion == LOW
True
False
MCAL_dio_write_pin(portx, pinNumber, HIGH) != DIO_OK
MCAL_dio_write_pin(portx, pinNumber, LOW) != DIO_OK
True
False
True
False
LED_OFF_ERROR
LED_OK

Figure 4: ECUAL_led_off function flowchart.

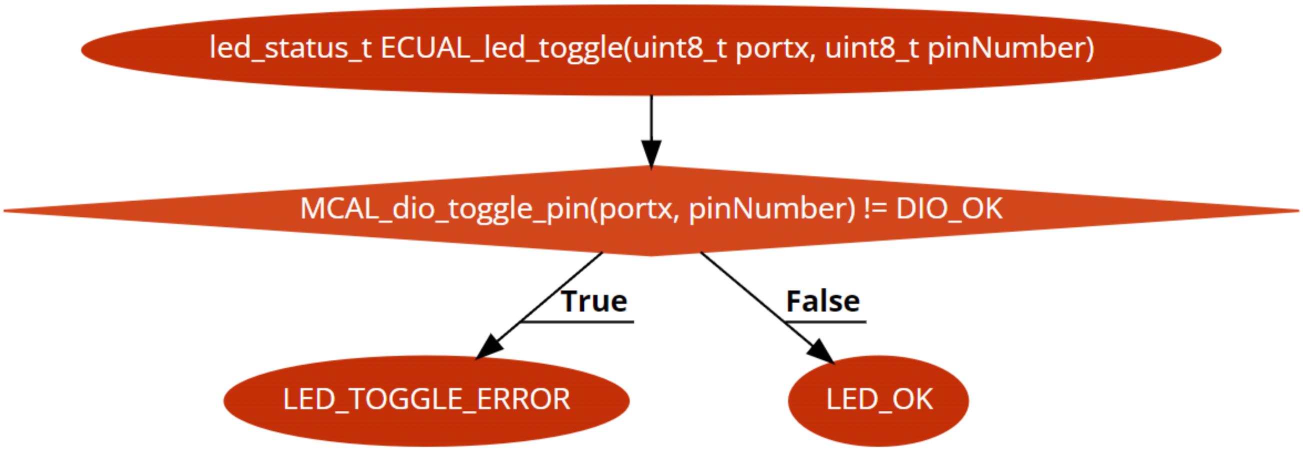

led_status_t ECUAL_led_toggle(uint8_t portx, uint8_t pinNumber)
MCAL_dio_toggle_pin(portx, pinNumber) != DIO_OK
True
False
LED_TOGGLE_ERROR
LED_OK

Figure 5: ECUAL_led_toggle function flowchart.

3.2-Button

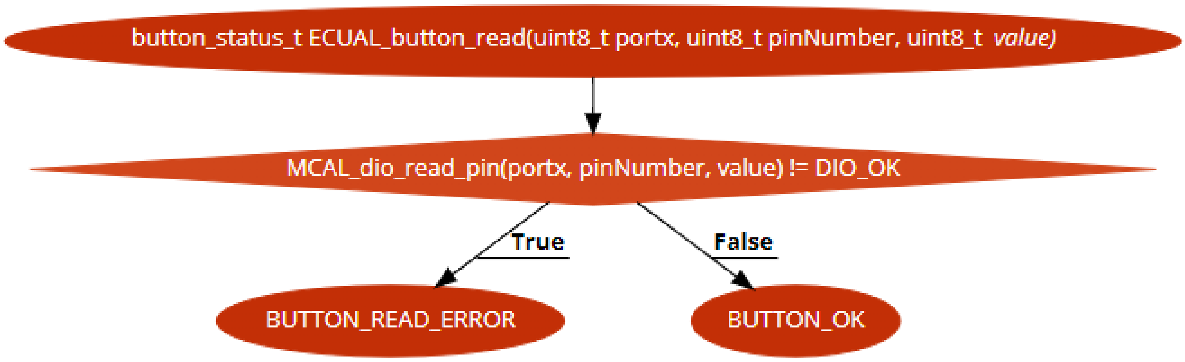

button_status_t ECUAL_button_read(uint8_t portx, uint8_t pinNumber, uint8_t value)
MCAL_dio_read_pin(portx, pinNumber, value) != DIO_OK
True
False
BUTTON_READ_ERROR
BUTTON_OK

Figure 6: ECUAL_button_read function flowchart.

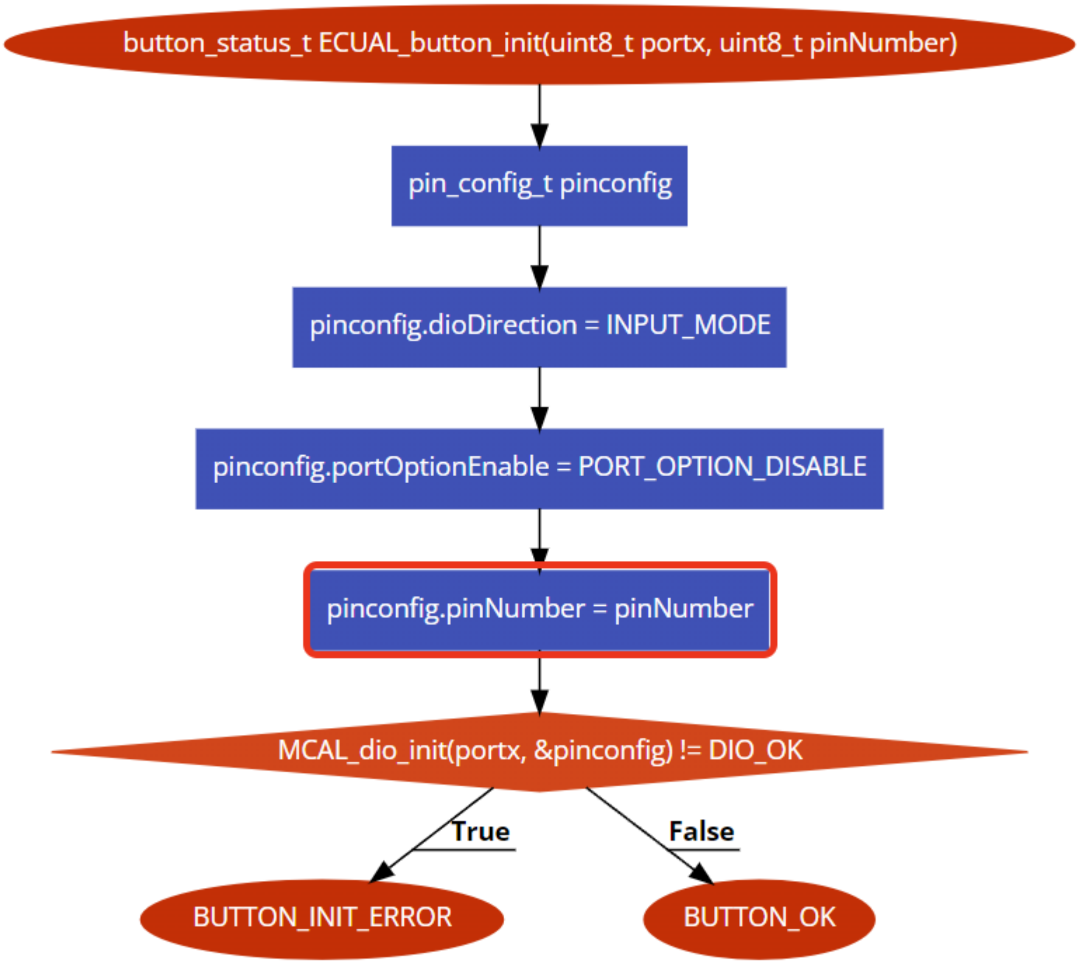

button_status_t ECUAL_button_init(uint8_t portx, uint8_t pinNumber)
pin_config_t pinconfig
pinconfig.dioDirection = INPUT_MODE
pinconfig.portOptionEnable = PORT_OPTION_DISABLE
pinconfig.pinNumber = pinNumber
MCAL_dio_init(portx, &pinconfig) != DIO_OK
True
False
BUTTON_INIT_ERROR
BUTTON_OK

Figure 7: ECUAL_button_init function flowchart.

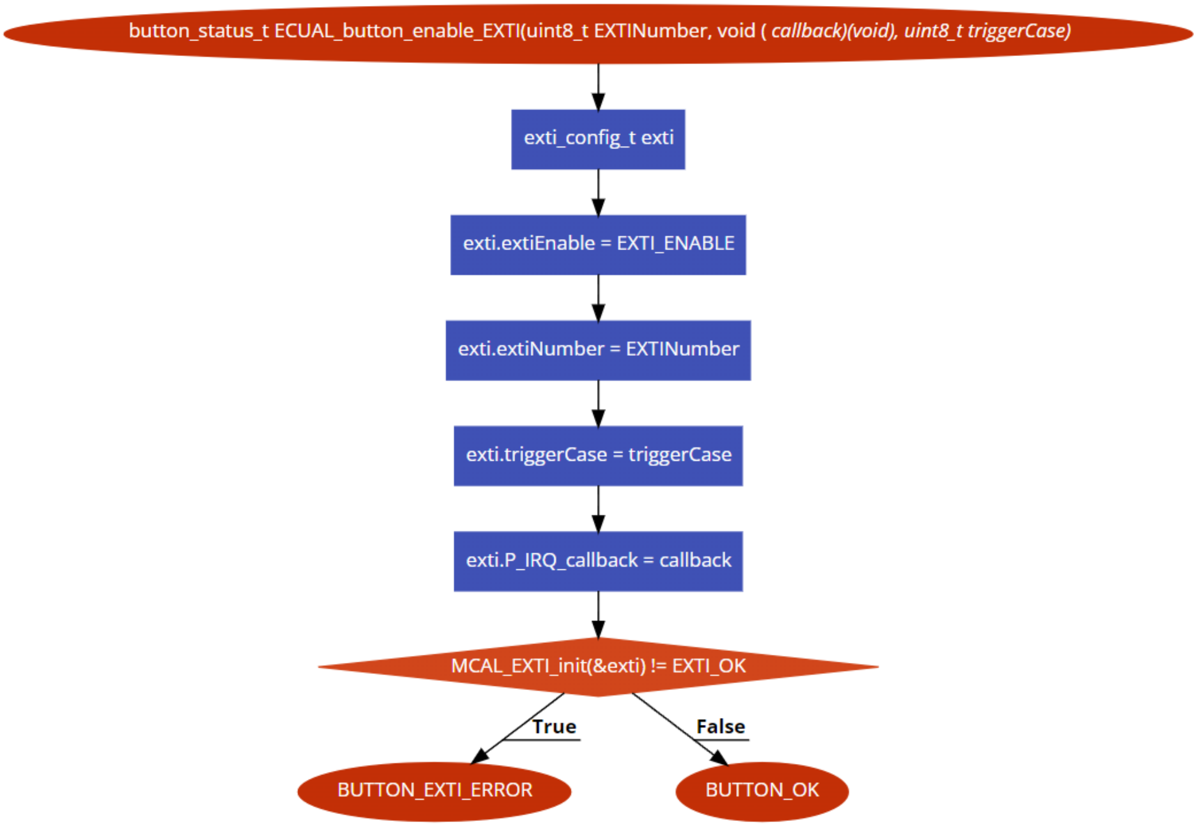
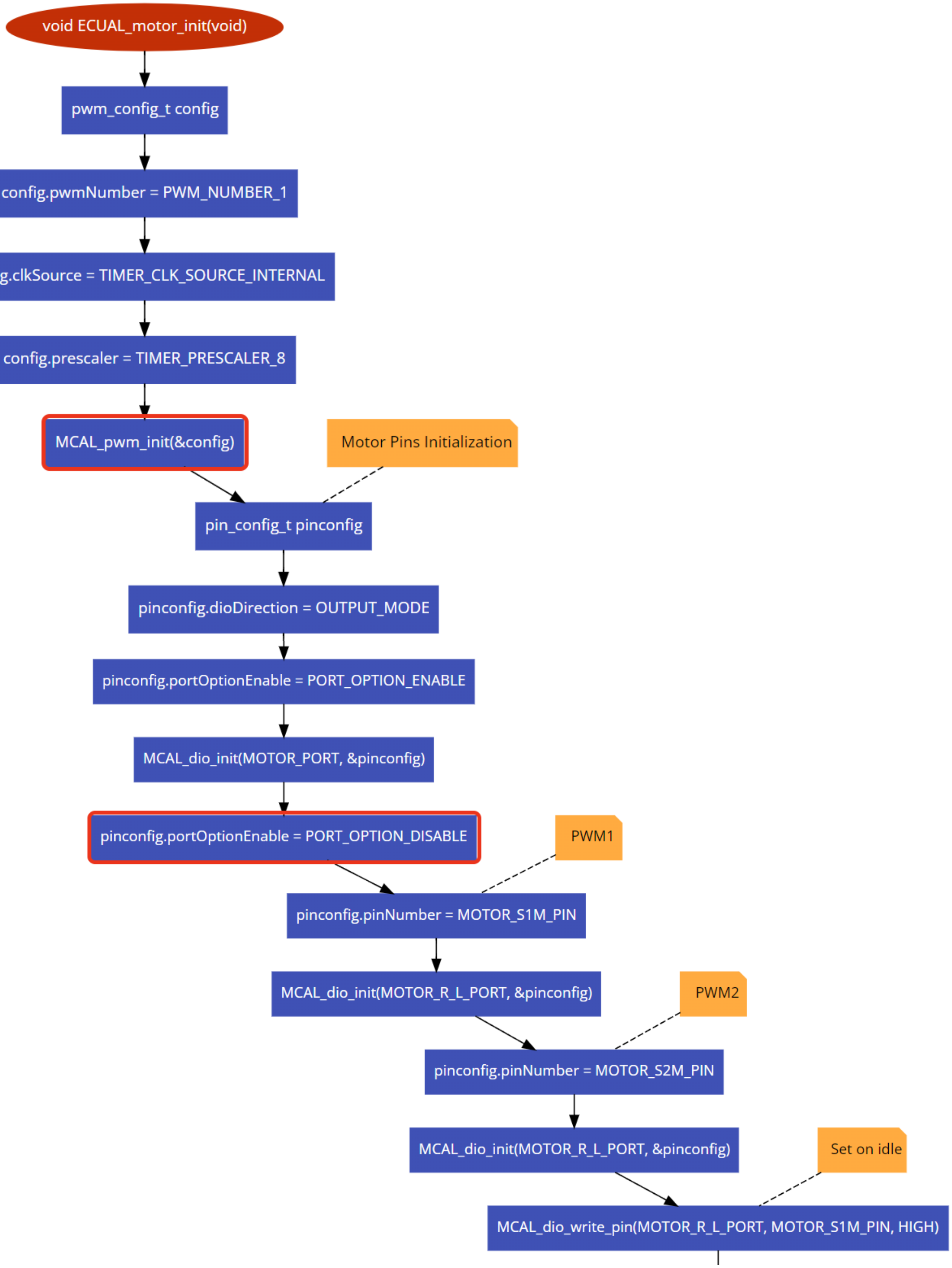

button_status_t ECUAL_button_enable_EXTI(uint8_t EXTINumber, void (*callback)(void), uint8_t triggerCase)*
exti_config_t exti
exti.extiEnable = EXTI_ENABLE
exti.extiNumber = EXTINumber
exti.triggerCase = triggerCase
exti.P_IRQ_callback = callback
MCAL_EXTI_init(&exti) != EXTI_OK
True
False
BUTTON_EXTI_ERROR
BUTTON_OK

Figure 8: ECUAL_button_enable_EXTI function flowchart.

2.3-Motor


void ECUAL_motor_init(void)
pwm_config_t config
config.pwmNumber = PWM_NUMBER_1
config.clkSource = TIMER_CLK_SOURCE_INTERNAL
config.prescaler = TIMER_PRESCALER_8
MCAL_pwm_init(&config)
Motor Pins Initialization
pin_config_t pinconfig
pinconfig.dioDirection = OUTPUT_MODE
pinconfig.portOptionEnable = PORT_OPTION_ENABLE
MCAL_dio_init(MOTOR_PORT, &pinconfig)
pinconfig.portOptionEnable = PORT_OPTION_DISABLE
PWM1
pinconfig.pinNumber = MOTOR_S1M_PIN
MCAL_dio_init(MOTOR_R_L_PORT, &pinconfig)
PWM2
pinconfig.pinNumber = MOTOR_S2M_PIN
MCAL_dio_init(MOTOR_R_L_PORT, &pinconfig)
Set on idle
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S1M_PIN, HIGH)

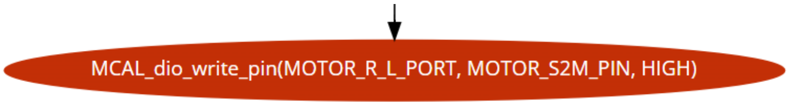


Figure 9: ECUAL_motor_init function flowchart.

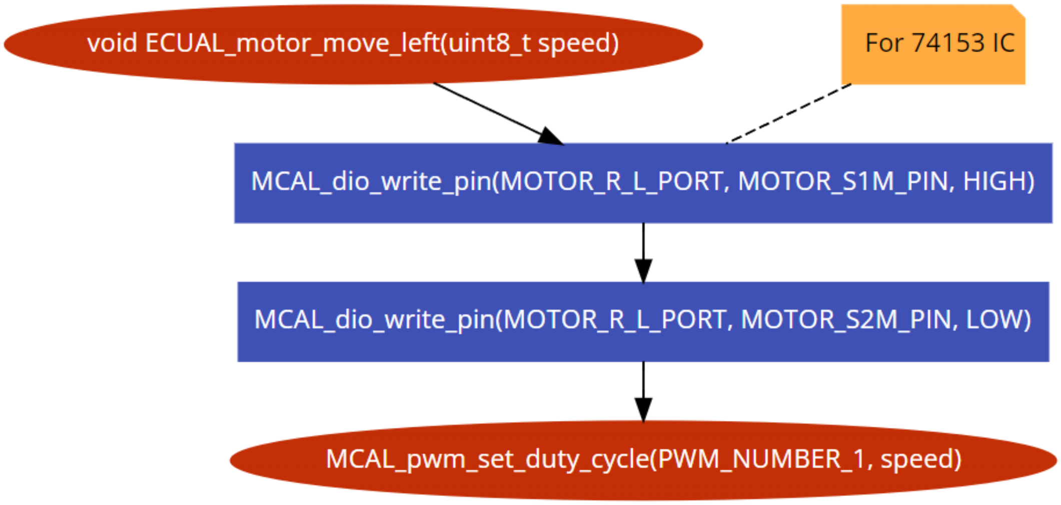

void ECUAL_motor_move_left(uint8_t speed)
For 74153 IC
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S1M_PIN, HIGH)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S2M_PIN, LOW)
MCAL_pwm_set_duty_cycle(PWM_NUMBER_1, speed)

Figure 10: ECUAL_motor_move_left function flowchart.

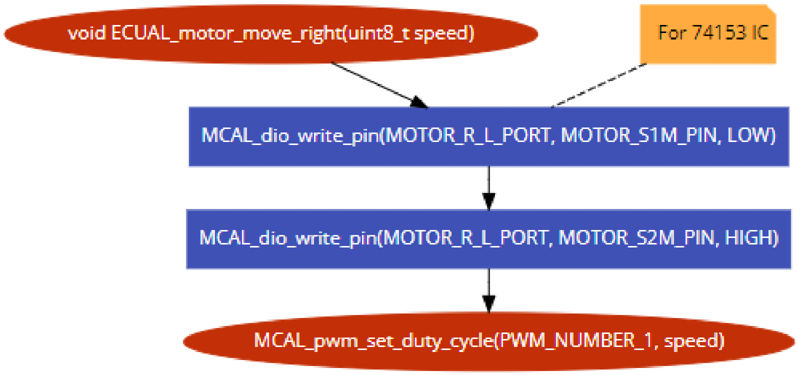

void ECUAL_motor_move_right(uint8_t speed)
For 74153 IC
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S1M_PIN, LOW)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S2M_PIN, HIGH)
MCAL_pwm_set_duty_cycle(PWM_NUMBER_1, speed)

Figure 11: ECUAL_motor_move_right function flowchart.

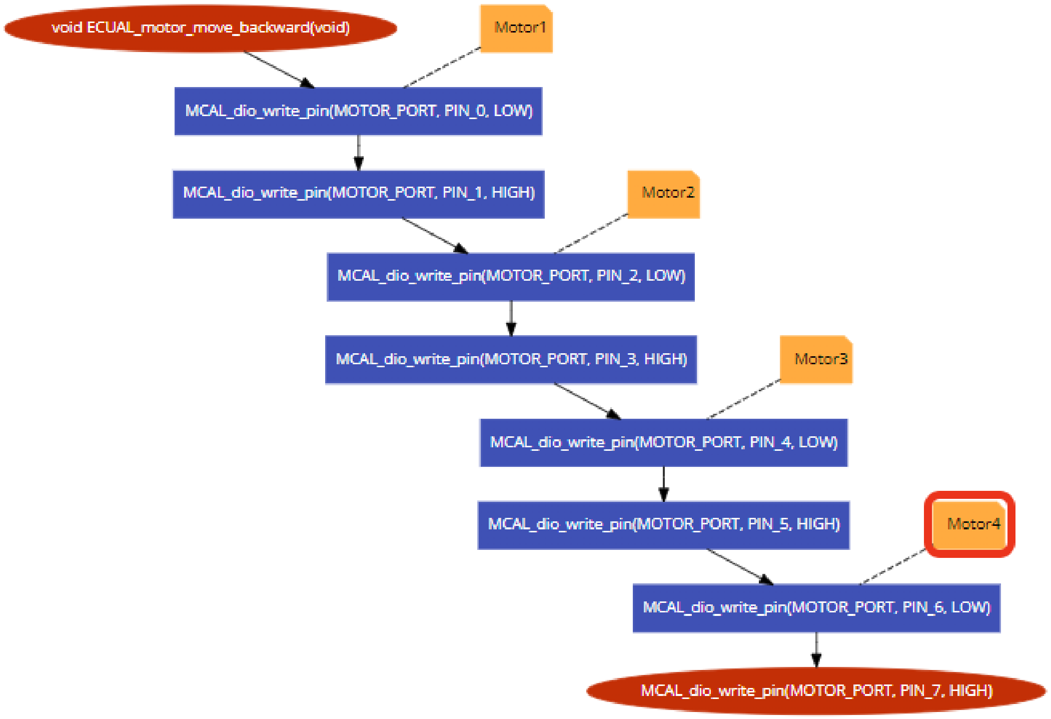

void ECUAL_motor_move_backward(void)
Motor1
MCAL_dio_write_pin(MOTOR_PORT, PIN_0, LOW)
MCAL_dio_write_pin(MOTOR_PORT, PIN_1, HIGH)
Motor2
MCAL_dio_write_pin(MOTOR_PORT, PIN_2, LOW)
MCAL_dio_write_pin(MOTOR_PORT, PIN_3, HIGH)
Motor3
MCAL_dio_write_pin(MOTOR_PORT, PIN_4, LOW)
MCAL_dio_write_pin(MOTOR_PORT, PIN_5, HIGH)
Motor4
MCAL_dio_write_pin(MOTOR_PORT, PIN_6, LOW)
MCAL_dio_write_pin(MOTOR_PORT, PIN_7, HIGH)

Figure 12: ECUAL_motor_move_backward function flowchart.

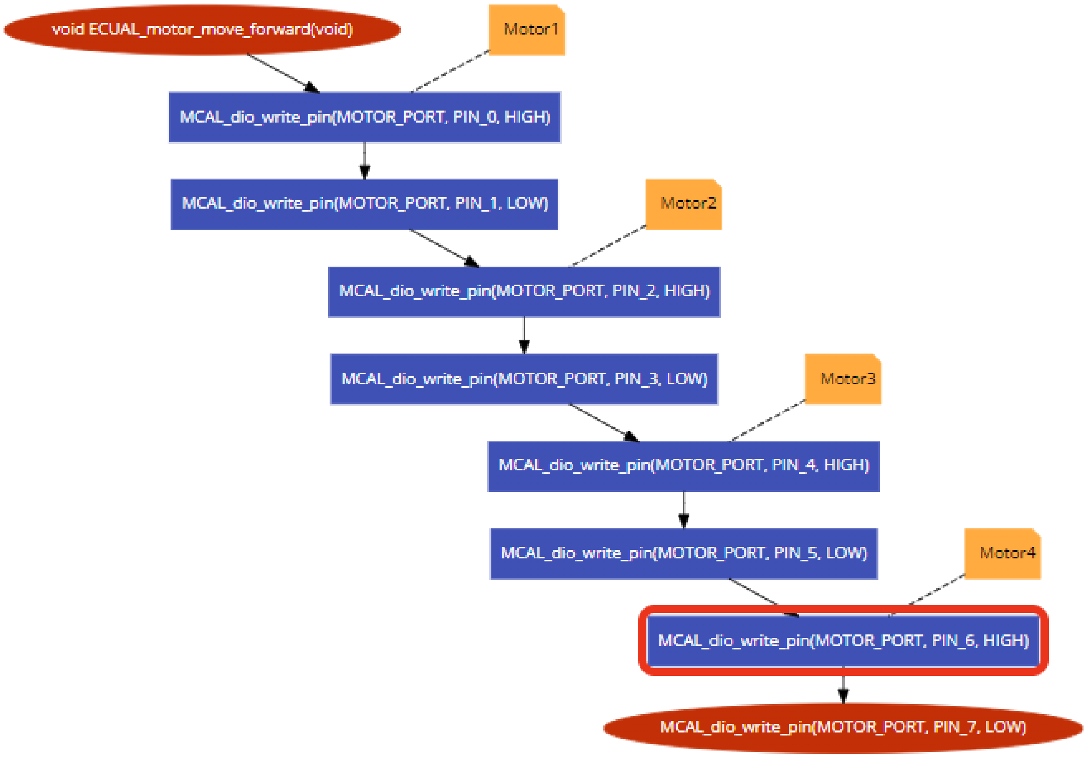

void ECUAL_motor_move_forward(void)
Motor1
MCAL_dio_write_pin(MOTOR_PORT, PIN_0, HIGH)
MCAL_dio_write_pin(MOTOR_PORT, PIN_1, LOW)
Motor2
MCAL_dio_write_pin(MOTOR_PORT, PIN_2, HIGH)
MCAL_dio_write_pin(MOTOR_PORT, PIN_3, LOW)
Motor3
MCAL_dio_write_pin(MOTOR_PORT, PIN_4, HIGH)
MCAL_dio_write_pin(MOTOR_PORT, PIN_5, LOW)
Motor4
MCAL_dio_write_pin(MOTOR_PORT, PIN_6, HIGH)
MCAL_dio_write_pin(MOTOR_PORT, PIN_7, LOW)

Figure 13: ECUAL_motor_move_forward function flowchart.

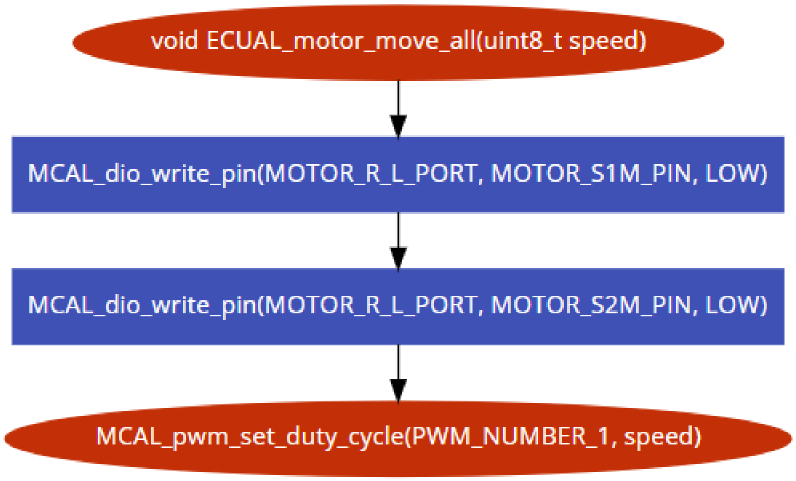

void ECUAL_motor_move_all(uint8_t speed)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S1M_PIN, LOW)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S2M_PIN, LOW)
MCAL_pwm_set_duty_cycle(PWM_NUMBER_1, speed)

Figure 14: ECUAL_motor_move_all function flowchart.

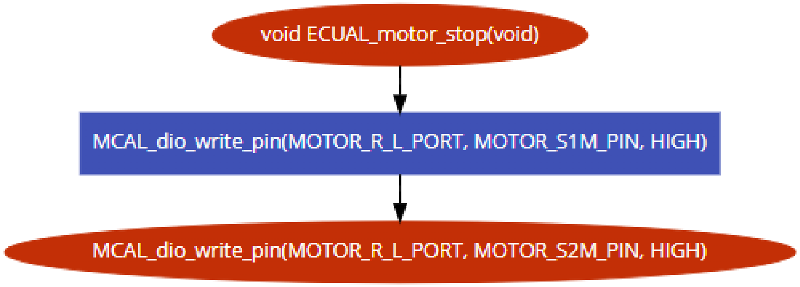

void ECUAL_motor_stop(void)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S1M_PIN, HIGH)
MCAL_dio_write_pin(MOTOR_R_L_PORT, MOTOR_S2M_PIN, HIGH)

Figure 15: ECUAL_motor_stop function flowchart.

3.4-DIO

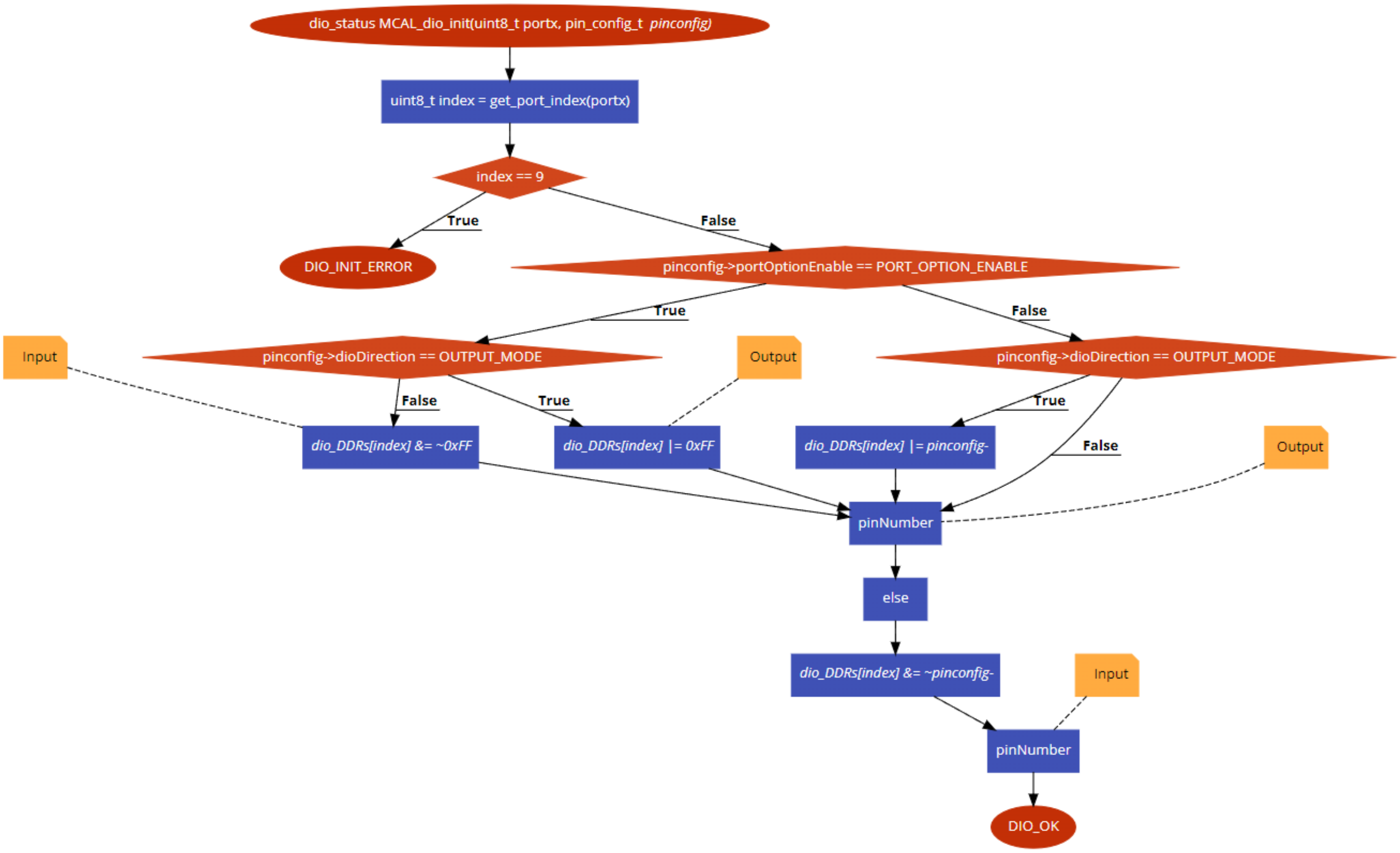

dio_status MCAL_dio_init(uint8_t portx, pin_config_t pinconfig)
uint8_t index = get_port_index(portx)
index == 9
True
False
DIO_INIT_ERROR
pinconfig->portOptionEnable == PORT_OPTION_ENABLE
True
False
Input
pinconfig->dioDirection == OUTPUT_MODE
Output
pinconfig->dioDirection == OUTPUT_MODE
False
True
True
False
dio_DDRs[index] &= ~0xFF
dio_DDRs[index] |= 0xFF
dio_DDRs[index] |= pinconfig-
Output
pinNumber
else
dio_DDRs[index] &= ~pinconfig-
Input
pinNumber
DIO_OK

Figure 16: MCAL_dio_init function flowchart.

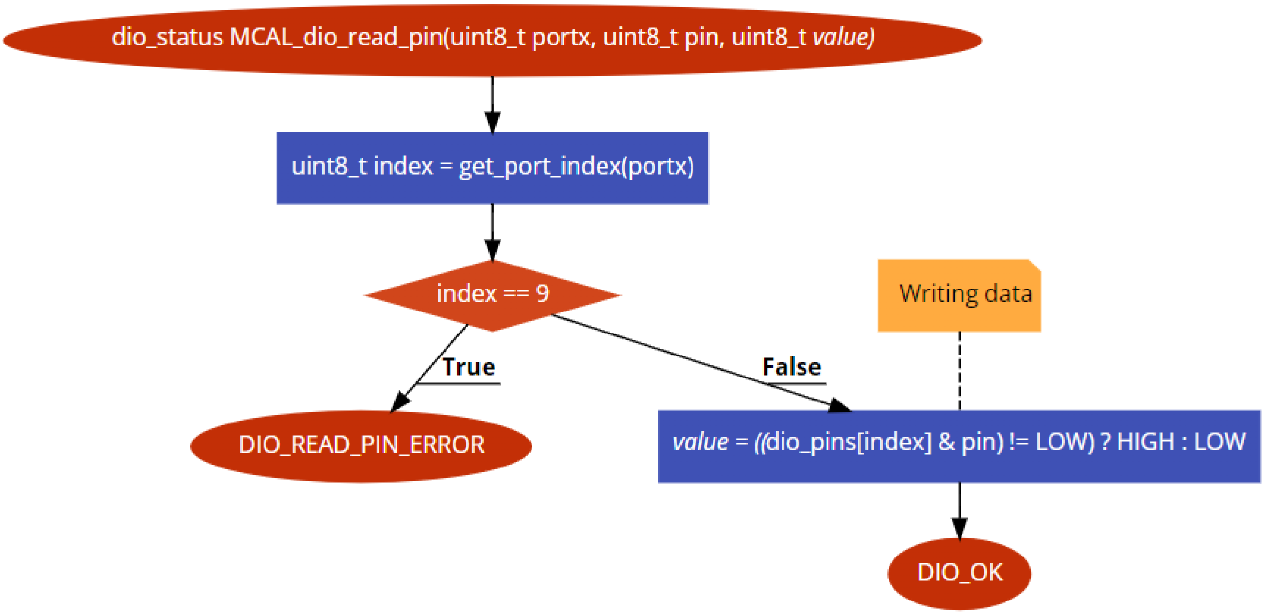

dio_status MCAL_dio_read_pin(uint8_t portx, uint8_t pin, uint8_t value)
uint8_t index = get_port_index(portx)
index == 9
True
False
Writing data
DIO_READ_PIN_ERROR
value = ((dio_pins[index] & pin) != LOW) ? HIGH : LOW
DIO_OK

Figure 18: MCAL_dio_read_pin function flowchart.

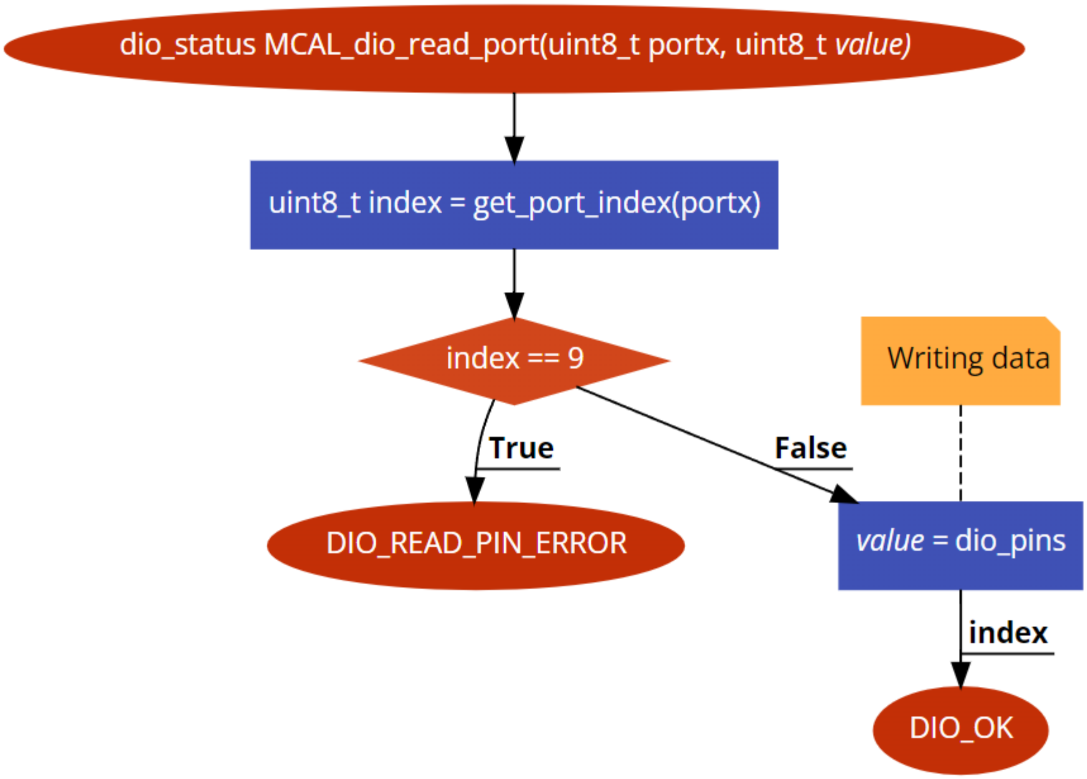

dio_status MCAL_dio_read_port(uint8_t portx, uint8_t value)
uint8_t index = get_port_index(portx)
index == 9
True
False
Writing data
DIO_READ_PIN_ERROR
value = dio_pins
index
DIO_OK

Figure 17: MCAL_dio_read_port function flowchart.

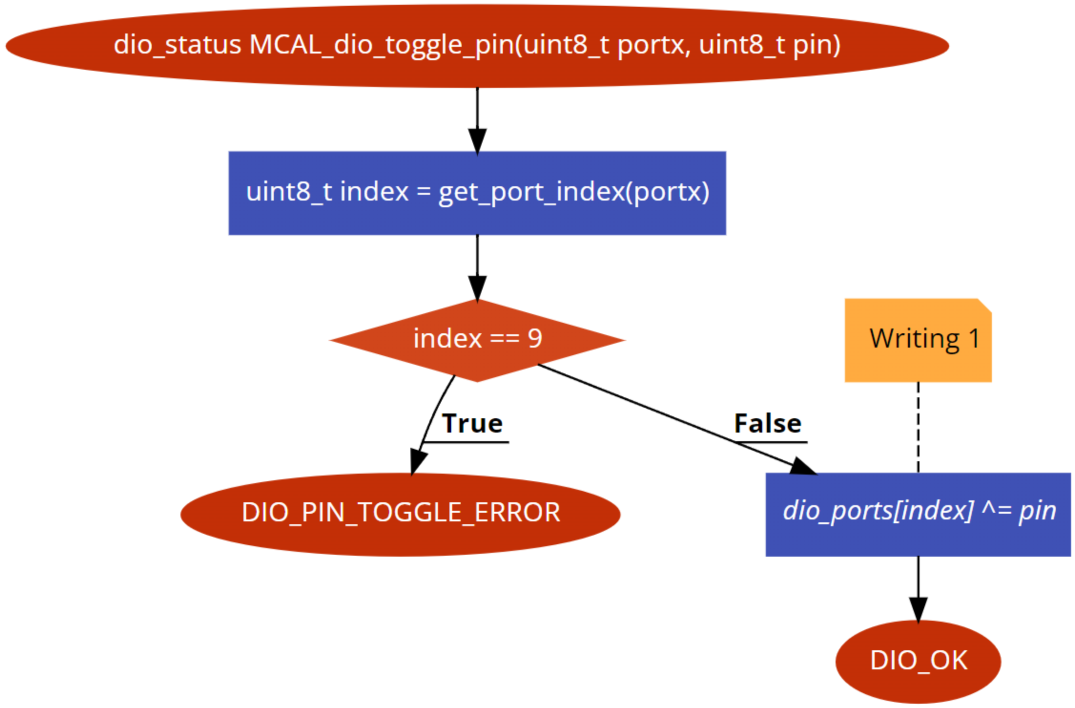

dio_status MCAL_dio_toggle_pin(uint8_t portx, uint8_t pin)
uint8_t index = get_port_index(portx)
index == 9
Writing 1
True
False
DIO_PIN_TOGGLE_ERROR
dio_ports[index] ^= pin
DIO_OK

Figure 19: MCAL_dio_toggle_pin function flowchart.

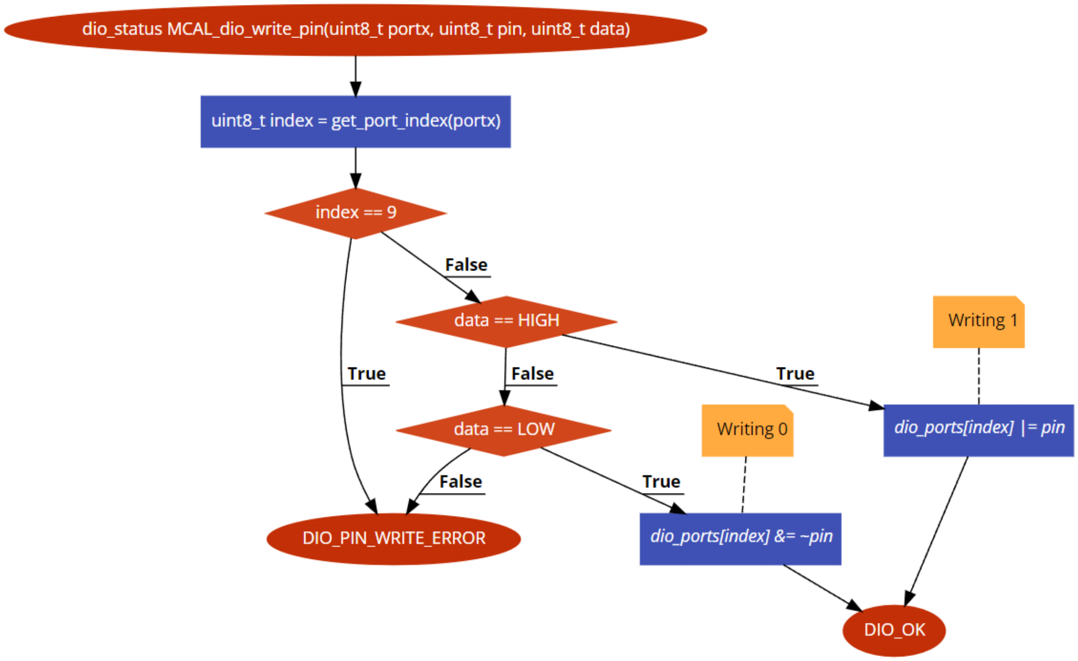


Figure 20: MCAL_dio_write_pin function flowchart.

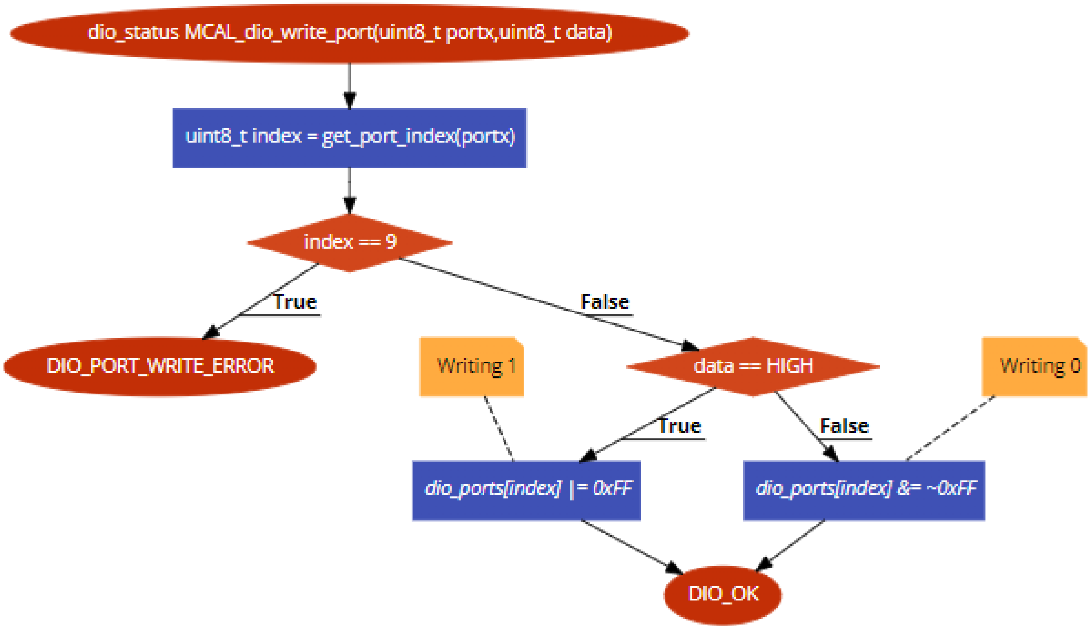

dio_status MCAL_dio_write_port(uint8_t portx,uint8_t data)
uint8_t index = get_port_index(portx)
index == 9
True
False
DIO_PORT_WRITE_ERROR
Writing 1
data == HIGH
Writing 0
True
False
dio_ports[index] |= 0xFF
dio_ports[index] &= ~0xFF
DIO_OK

Figure 21: MCAL_dio_write_port function flowchart.

3.5-EXTI

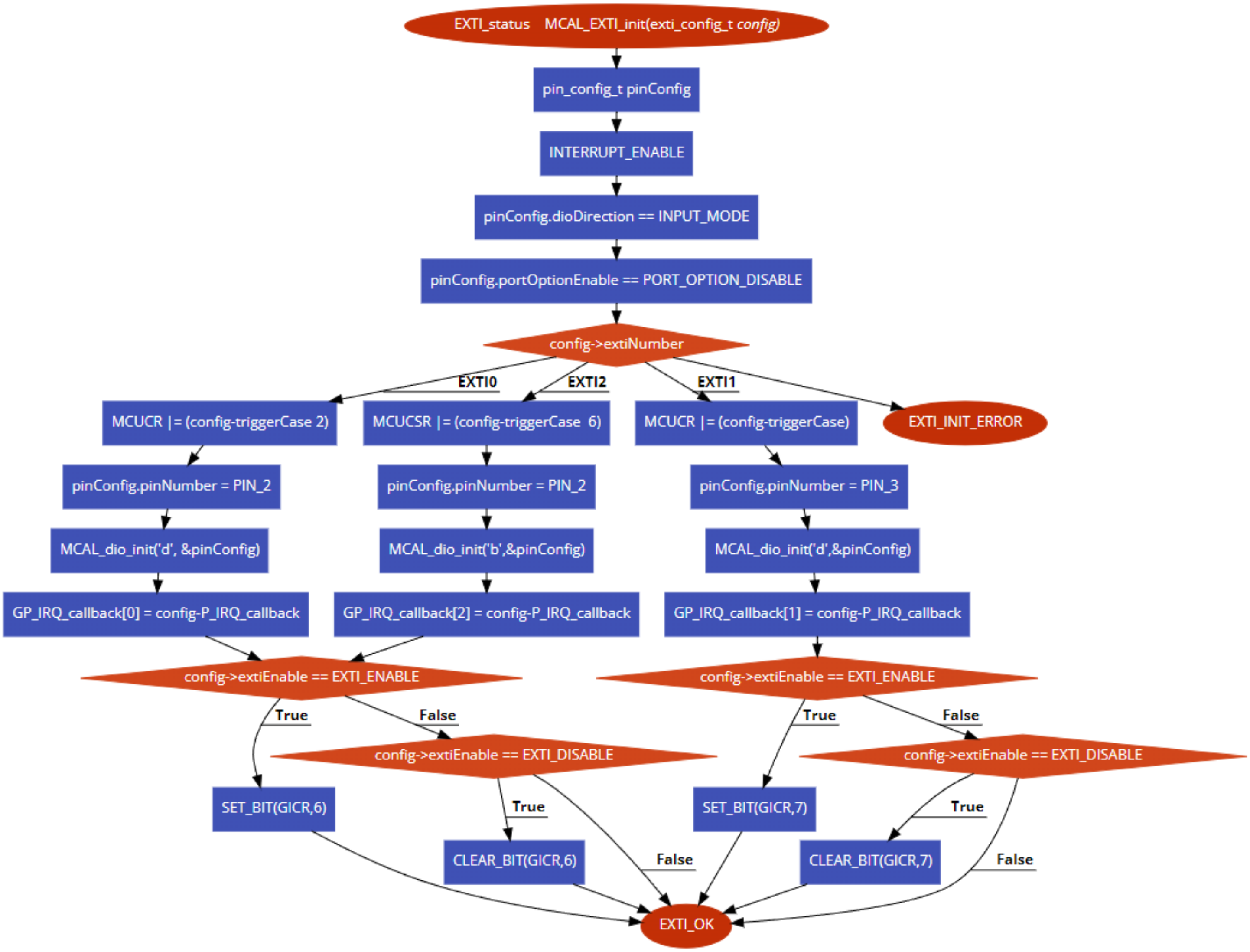

EXTI_status MCAL_EXTI_init(exti_config_t config)
pin_config_t pinConfig
INTERRUPT_ENABLE
pinConfig.dioDirection == INPUT_MODE
pinConfig.portOptionEnable == PORT_OPTION_DISABLE
config->extiNumber
EXTI0
EXTI2
EXTI1
MCUCR |= (config-triggerCase 2)
MCUCSR |= (config-triggerCase 6)
MCUCR |= (config-triggerCase)
EXTI_INIT_ERROR
pinConfig.pinNumber = PIN_2
pinConfig.pinNumber = PIN_2
pinConfig.pinNumber = PIN_3
MCAL_dio_init('d', &pinConfig)
MCAL_dio_init('b',&pinConfig)
MCAL_dio_init('d',&pinConfig)
GP_IRQ_callback[0] = config-P_IRQ_callback
GP_IRQ_callback[2] = config-P_IRQ_callback
GP_IRQ_callback[1] = config-P_IRQ_callback
config->extiEnable == EXTI_ENABLE
config->extiEnable == EXTI_ENABLE
True
False
True
False
config->extiEnable == EXTI_DISABLE
config->extiEnable == EXTI_DISABLE
SET_BIT(GICR,6)
True
SET_BIT(GICR,7)
True
CLEAR_BIT(GICR,6)
False
CLEAR_BIT(GICR,7)
False
EXTI_OK

Figure 22: MCAL_EXTI_init function flowchart.

3.6-Timer

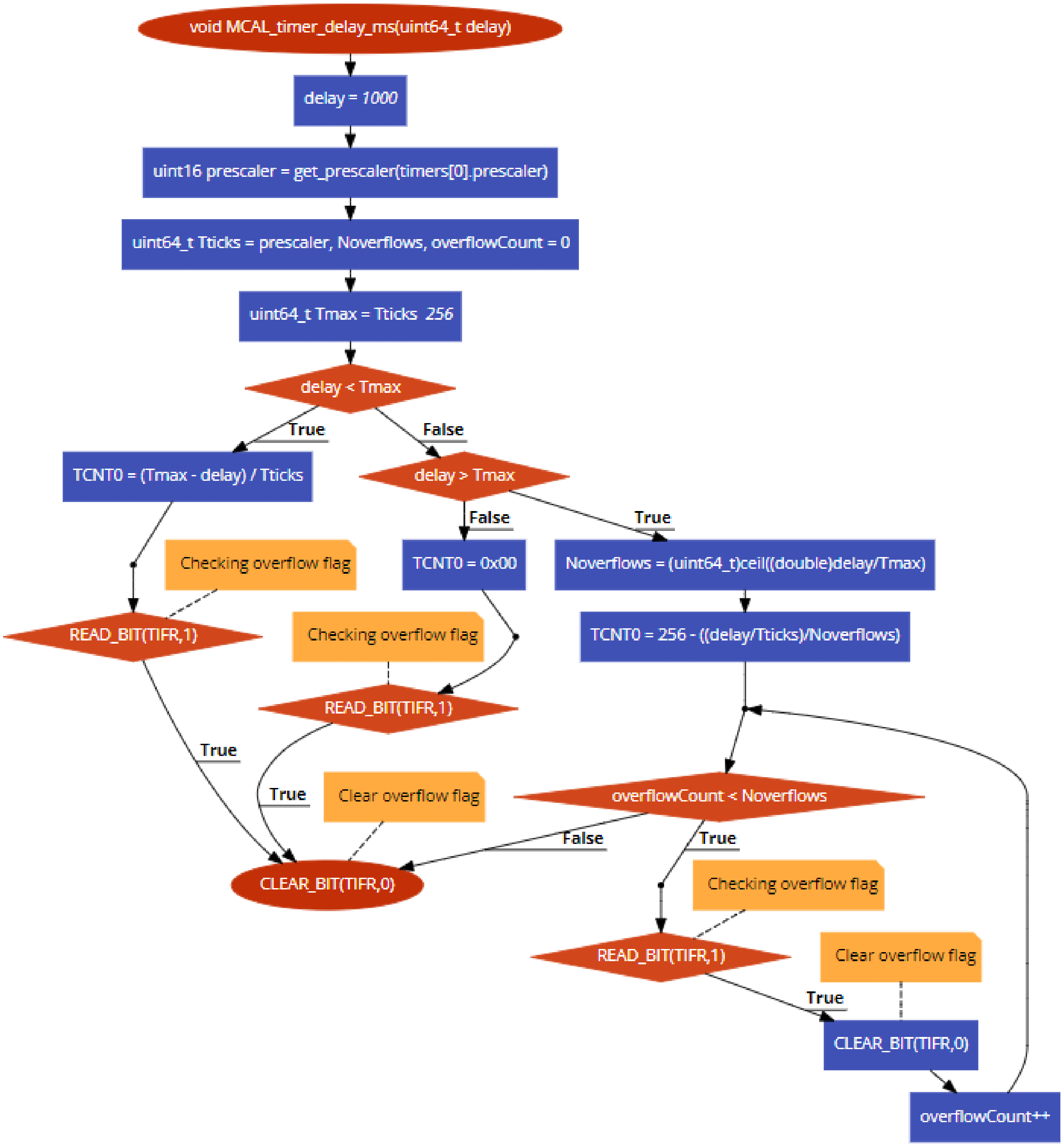

void MCAL_timer_delay_ms(uint64_t delay)
delay = 1000
uint16 prescaler = get_prescaler(timers[0].prescaler)
uint64_t Tticks = prescaler, Noverflows, overflowCount = 0
uint64_t Tmax = Tticks 256
delay < Tmax
True
False
TCNT0 = (Tmax - delay) / Tticks
delay > Tmax
False
True
Checking overflow flag
TCNT0 = 0x00
Noverflows = (uint64_t)ceil((double)delay/Tmax)
READ_BIT(TIFR,1)
Checking overflow flag
TCNT0 = 256 - ((delay/Tticks)/Noverflows)
READ_BIT(TIFR,1)
True
True
Clear overflow flag
overflowCount < Noverflows
False
True
CLEAR_BIT(TIFR,0)
Checking overflow flag
READ_BIT(TIFR,1)
Clear overflow flag
True
CLEAR_BIT(TIFR,0)
overflowCount++

Figure 23: MCAL_timer_delay_ms function flowchart.

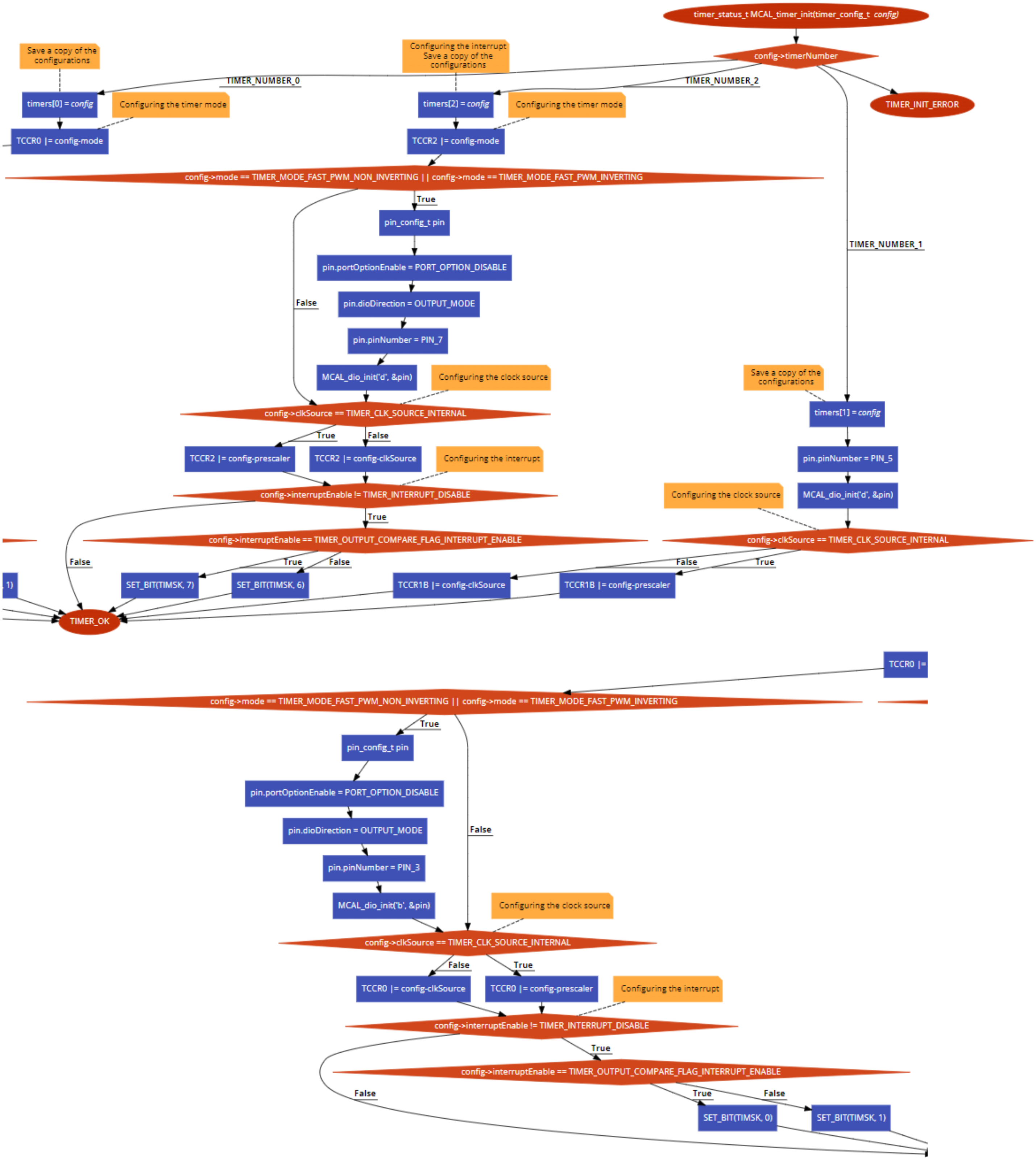

timer_status_t MCAL_timer_init(timer_config_t config)
config->timerNumber
TIMER_NUMBER_0
TIMER_NUMBER_2
TIMER_NUMBER_1
TIMER_INIT_ERROR
Save a copy of the configurations
Configuring the interrupt Save a copy of the configurations
timers[0] = config
Configuring the timer mode
TCCR0 |= config-mode
timers[2] = config
Configuring the timer mode
TCCR2 |= config-mode
config->mode == TIMER_MODE_FAST_PWM_NON_INVERTING || config->mode == TIMER_MODE_FAST_PWM_INVERTING
True
False
pin_config_t pin
pin.portOptionEnable = PORT_OPTION_DISABLE
pin.dioDirection = OUTPUT_MODE
pin.pinNumber = PIN_7
MCAL_dio_init('d', &pin)
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
True
False
TCCR2 |= config-prescaler
TCCR2 |= config-clkSource
Configuring the interrupt
config->interruptEnable != TIMER_INTERRUPT_DISABLE
True
config->interruptEnable == TIMER_OUTPUT_COMPARE_FLAG_INTERRUPT_ENABLE
False
True
False
1)
SET_BIT(TIMSK, 7)
SET_BIT(TIMSK, 6)
TIMER_OK
Save a copy of the configurations
timers[1] = config
pin.pinNumber = PIN_5
MCAL_dio_init('d', &pin)
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
False
True
TCCR1B |= config-clkSource
TCCR1B |= config-prescaler
TCCR0 |=
config->mode == TIMER_MODE_FAST_PWM_NON_INVERTING || config->mode == TIMER_MODE_FAST_PWM_INVERTING
True
False
pin_config_t pin
pin.portOptionEnable = PORT_OPTION_DISABLE
pin.dioDirection = OUTPUT_MODE
pin.pinNumber = PIN_3
MCAL_dio_init('b', &pin)
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
False
True
TCCR0 |= config-clkSource
TCCR0 |= config-prescaler
Configuring the interrupt
config->interruptEnable != TIMER_INTERRUPT_DISABLE
True
config->interruptEnable == TIMER_OUTPUT_COMPARE_FLAG_INTERRUPT_ENABLE
False
True
False
SET_BIT(TIMSK, 0)
SET_BIT(TIMSK, 1)

Figure 24: MCAL_timer_init function flowchart.

3.7-PWM

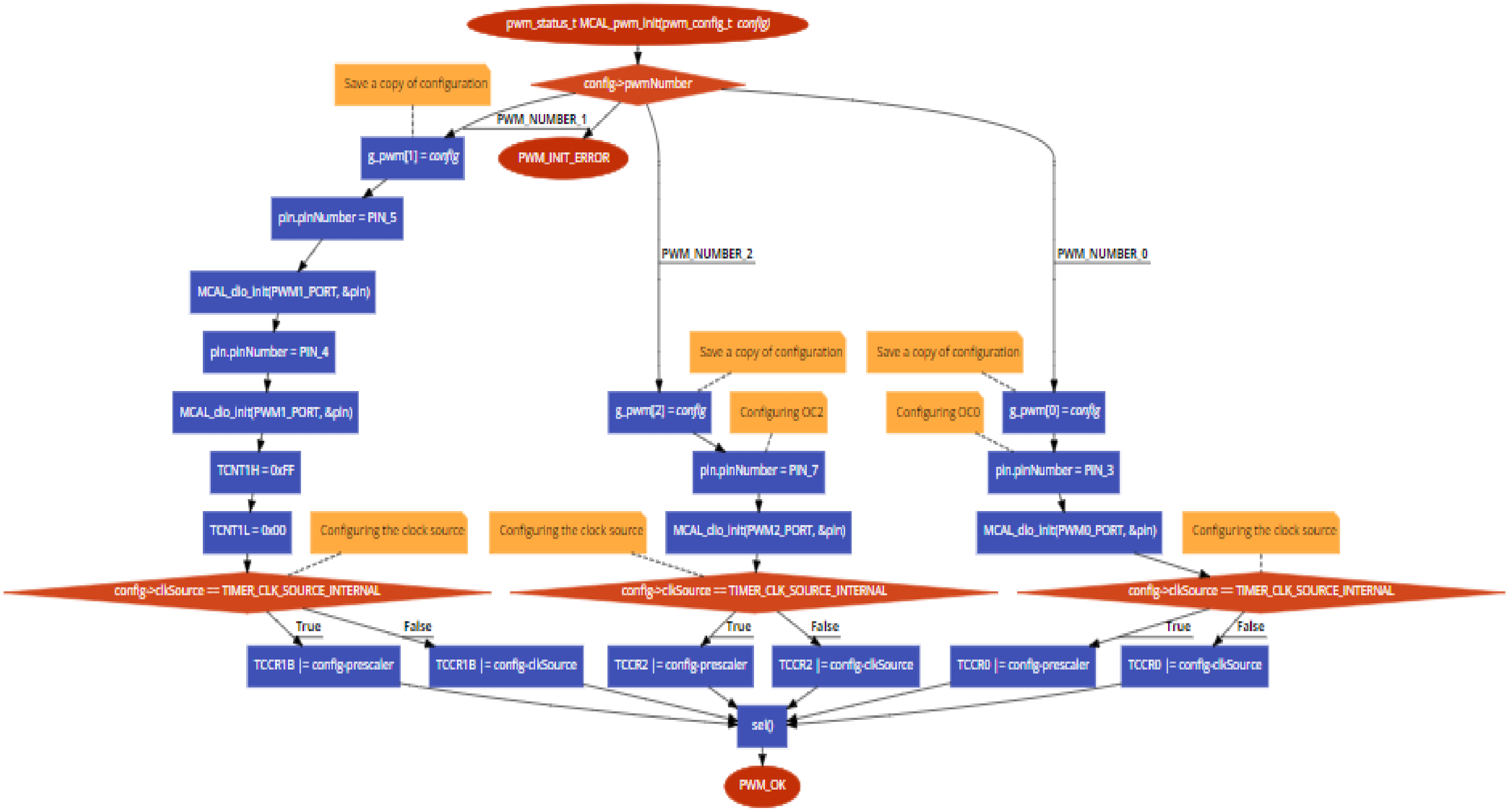

pwm_status_t MCAL_pwm_init(pwm_config_t config)
config->pwmNumber
Save a copy of configuration
PWM_NUMBER_1
g_pwm[1] = config
PWM_INIT_ERROR
pin.pinNumber = PIN_5
MCAL_dio_init(PWM1_PORT, &pin)
pin.pinNumber = PIN_4
MCAL_dio_init(PWM1_PORT, &pin)
TCNT1H = 0xFF
TCNT1L = 0x00
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
True
False
TCCR1B |= config-prescaler
TCCR1B |= config-clkSource
PWM_NUMBER_2
Save a copy of configuration
g_pwm[2] = config
Configuring OC2
pin.pinNumber = PIN_7
MCAL_dio_init(PWM2_PORT, &pin)
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
True
False
TCCR2 |= config-prescaler
TCCR2 |= config-clkSource
PWM_NUMBER_0
Save a copy of configuration
g_pwm[0] = config
Configuring OC0
pin.pinNumber = PIN_3
MCAL_dio_init(PWM0_PORT, &pin)
Configuring the clock source
config->clkSource == TIMER_CLK_SOURCE_INTERNAL
True
False
TCCR0 |= config-prescaler
TCCR0 |= config-clkSource
sei()
PWM_OK

Figure 25: MCAL_pwm_init function flowchart.

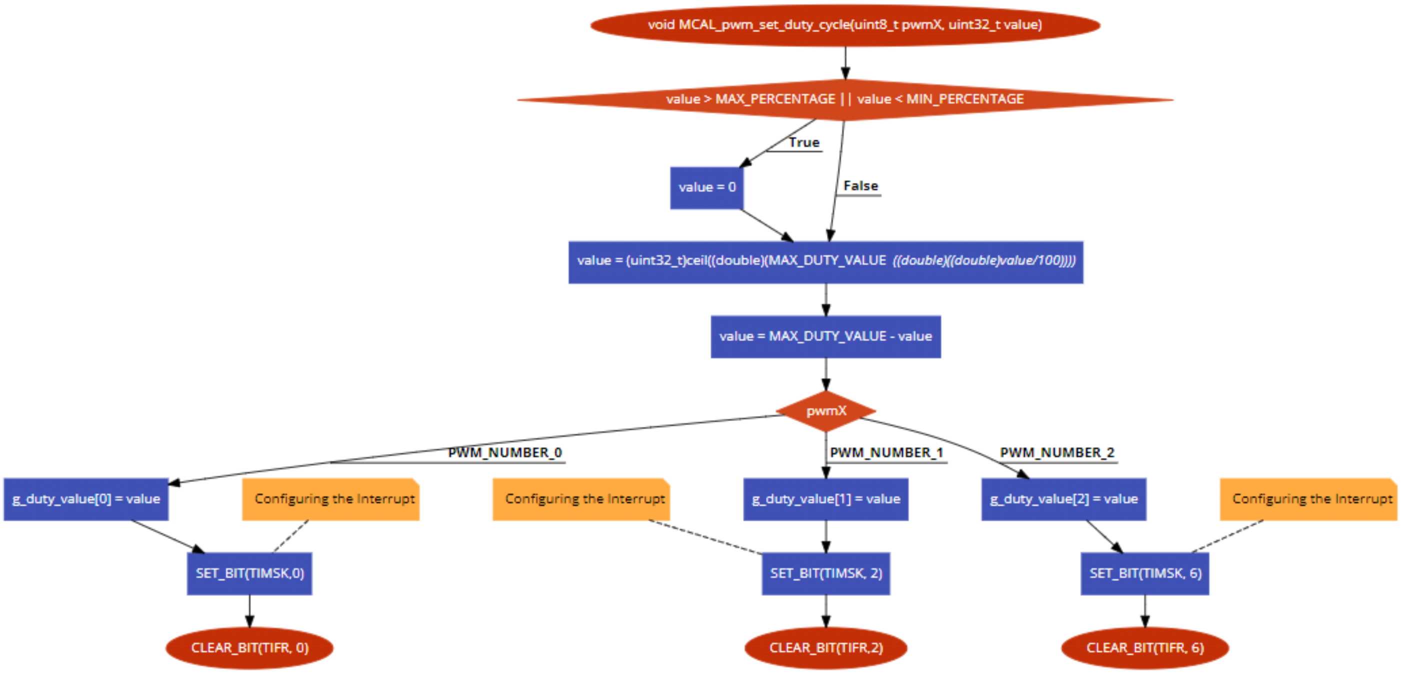

void MCAL_pwm_set_duty_cycle(uint8_t pwmX, uint32_t value)
value > MAX_PERCENTAGE || value < MIN_PERCENTAGE
True
value = 0
False
value = (uint32_t)ceil((double)(MAX_DUTY_VALUE ((double)((double)value/100))))
value = MAX_DUTY_VALUE - value
pwmX
PWM_NUMBER_0
PWM_NUMBER_1
PWM_NUMBER_2
g_duty_value[0] = value
Configuring the Interrupt
Configuring the Interrupt
g_duty_value[1] = value
g_duty_value[2] = value
Configuring the Interrupt
SET_BIT(TIMSK,0)
SET_BIT(TIMSK, 2)
SET_BIT(TIMSK, 6)
CLEAR_BIT(TIFR, 0)
CLEAR_BIT(TIFR,2)
CLEAR_BIT(TIFR, 6)

Figure 26: MCAL_pwm_set_duty_cycle function flowchart.

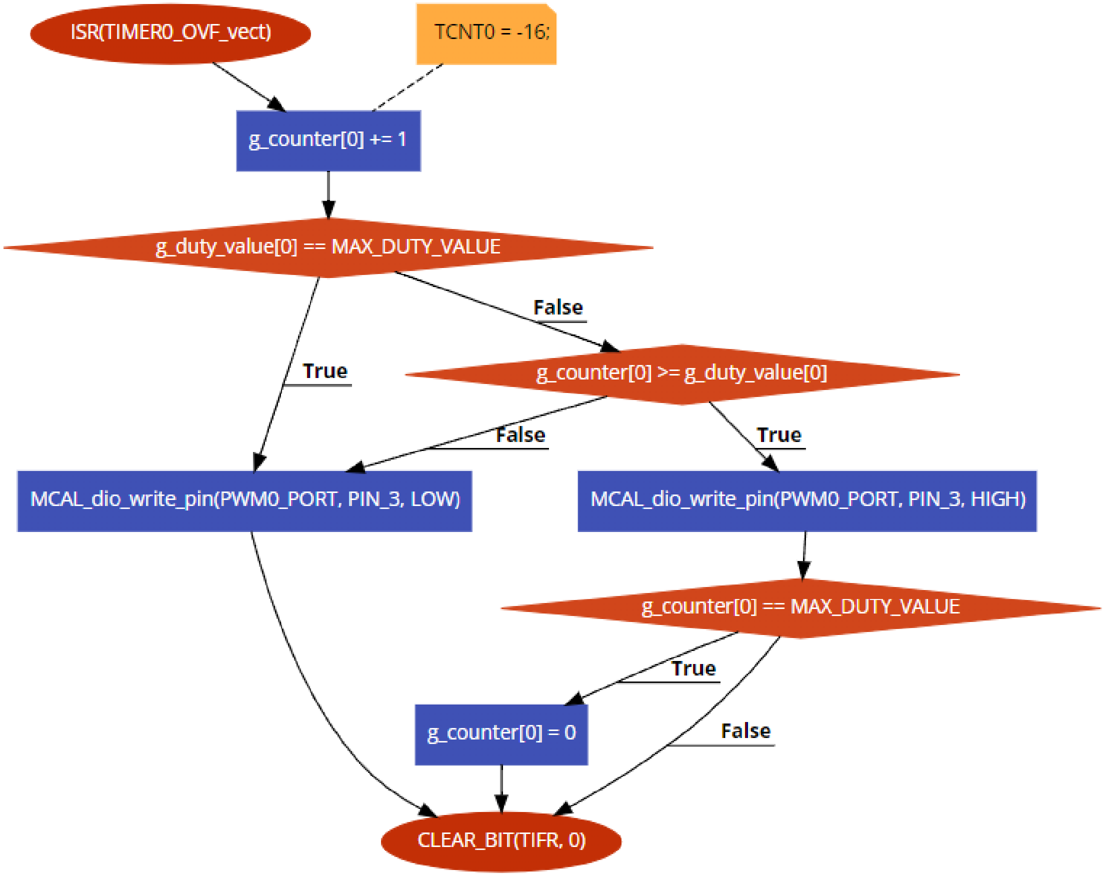

ISR(TIMER0_OVF_vect)
TCNT0 = -16;
g_counter[0] += 1
g_duty_value[0] == MAX_DUTY_VALUE
False
True
g_counter[0] >= g_duty_value[0]
False
True
MCAL_dio_write_pin(PWM0_PORT, PIN_3, LOW)
MCAL_dio_write_pin(PWM0_PORT, PIN_3, HIGH)
g_counter[0] == MAX_DUTY_VALUE
True
False
g_counter[0] = 0
CLEAR_BIT(TIFR, 0)

Figure 27: ISR(TIMER0_OVF_vect) handler flowchart.

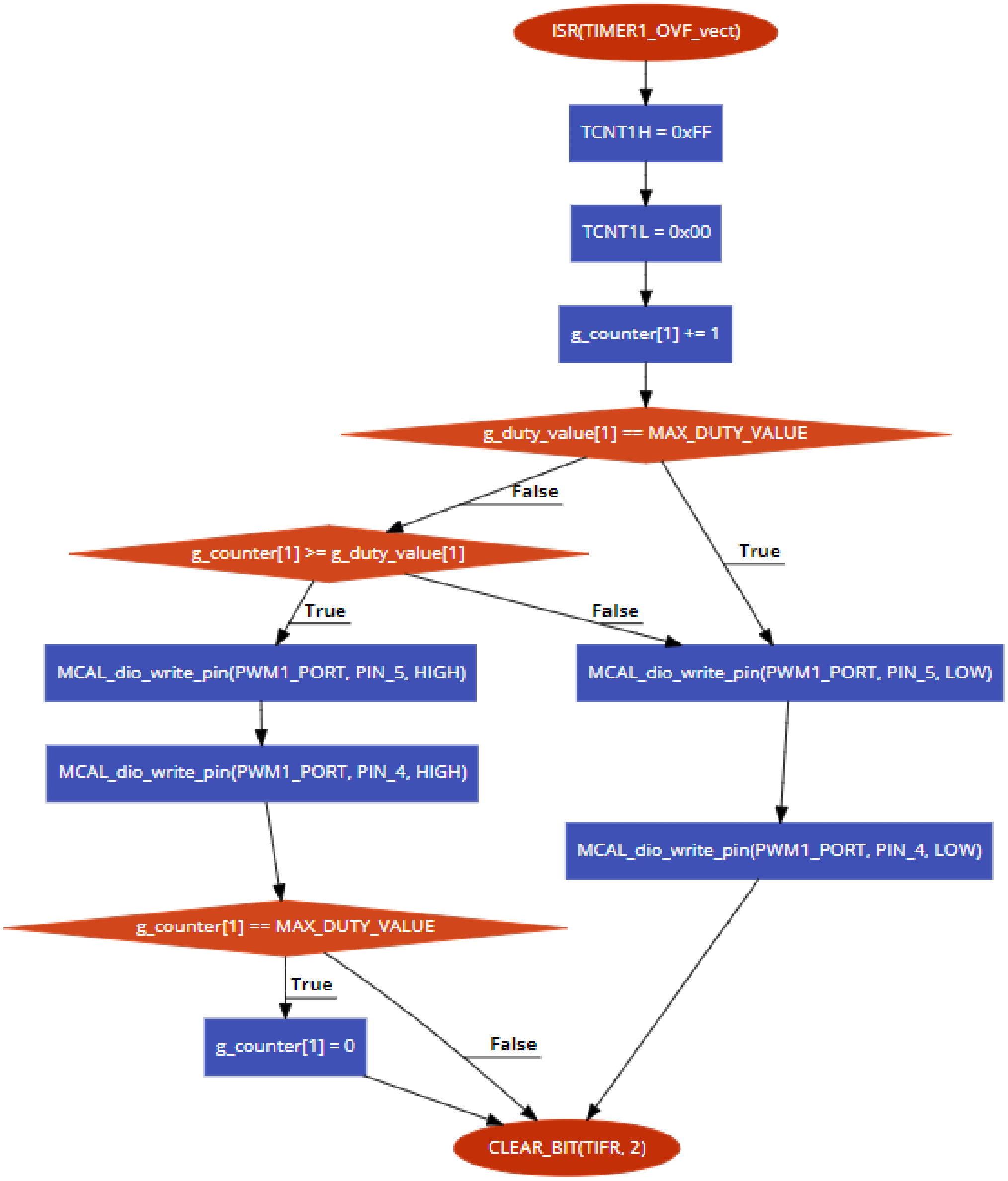

ISR(TIMER1_OVF_vect)
TCNT1H = 0xFF
TCNT1L = 0x00
g_counter[1] += 1
g_duty_value[1] == MAX_DUTY_VALUE
False
True
g_counter[1] >= g_duty_value[1]
True
False
MCAL_dio_write_pin(PWM1_PORT, PIN_5, HIGH)
MCAL_dio_write_pin(PWM1_PORT, PIN_5, LOW)
MCAL_dio_write_pin(PWM1_PORT, PIN_4, HIGH)
MCAL_dio_write_pin(PWM1_PORT, PIN_4, LOW)
g_counter[1] == MAX_DUTY_VALUE
True
False
g_counter[1] = 0
CLEAR_BIT(TIFR, 2)

Figure 28: ISR(TIMER1_OVF_vect) handler flowchart.

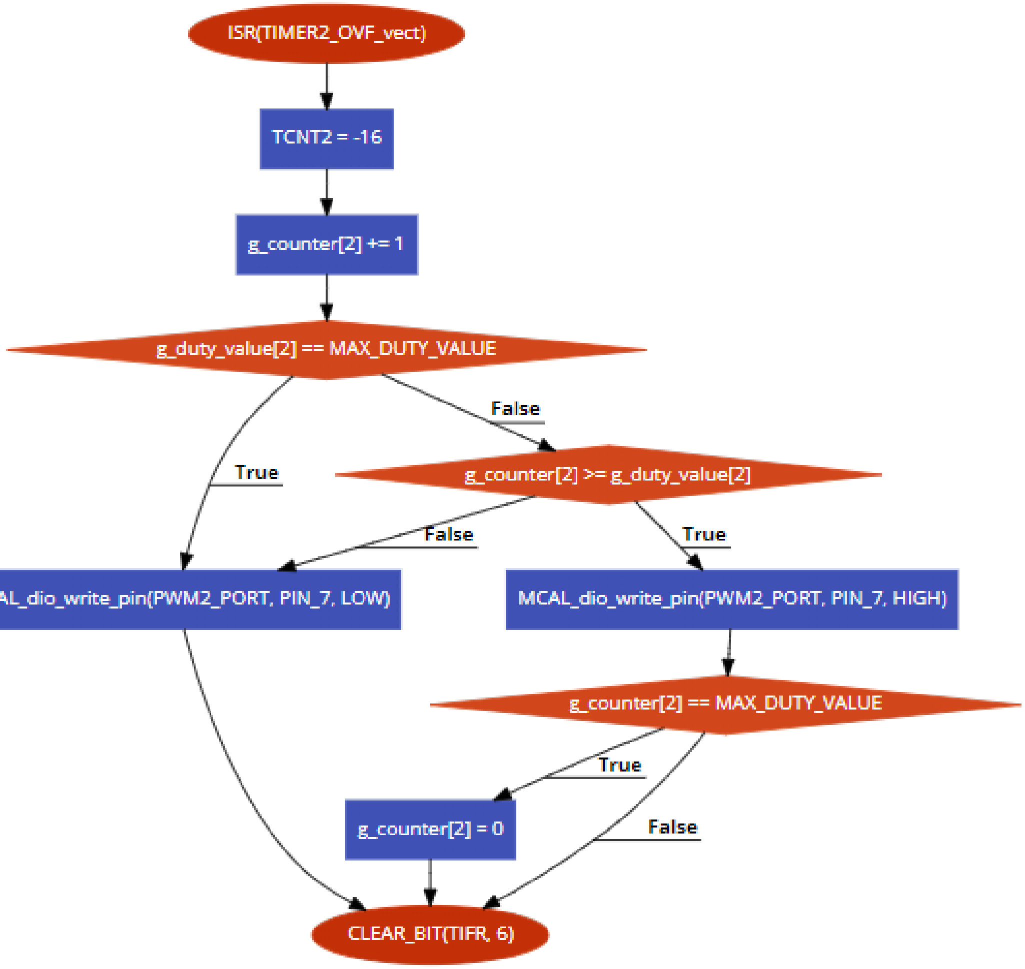
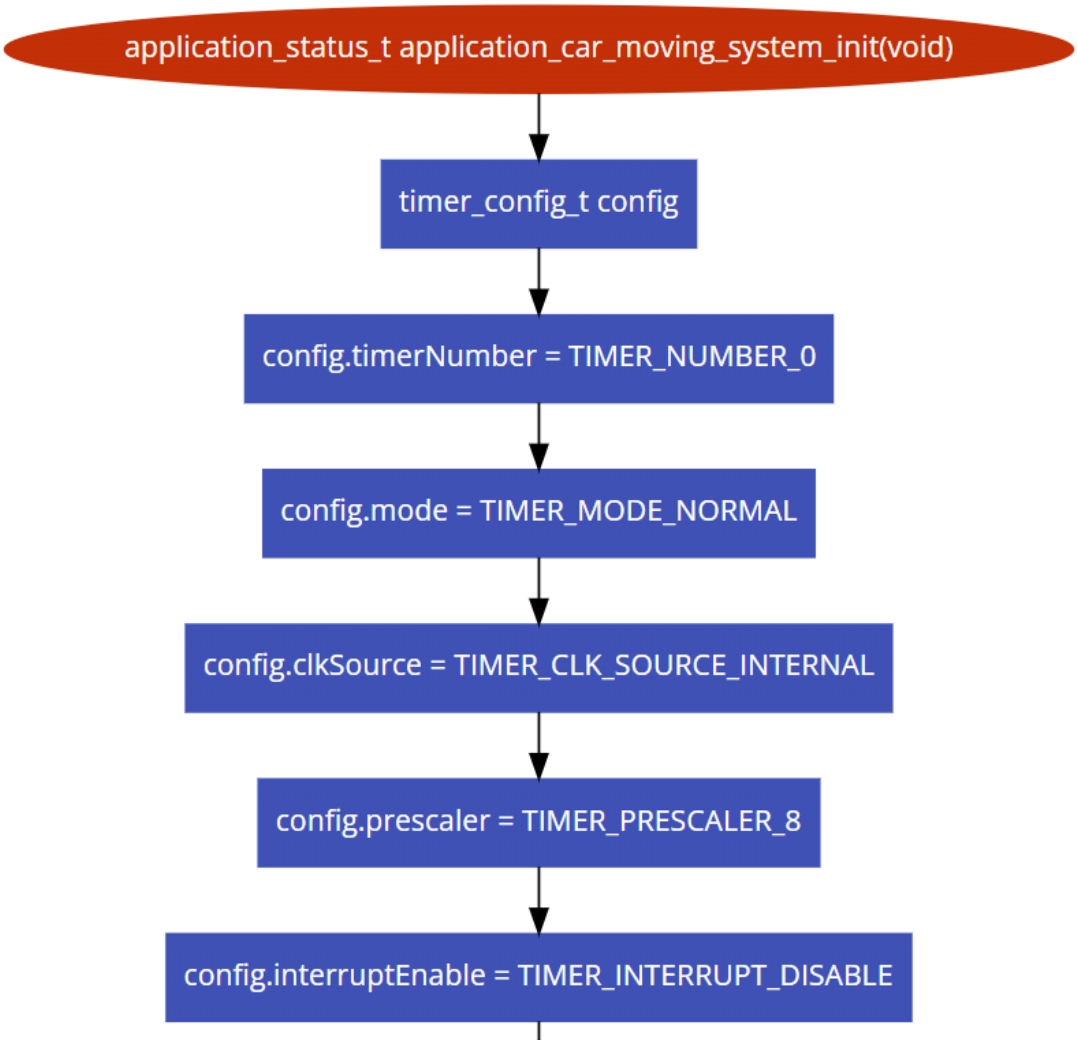
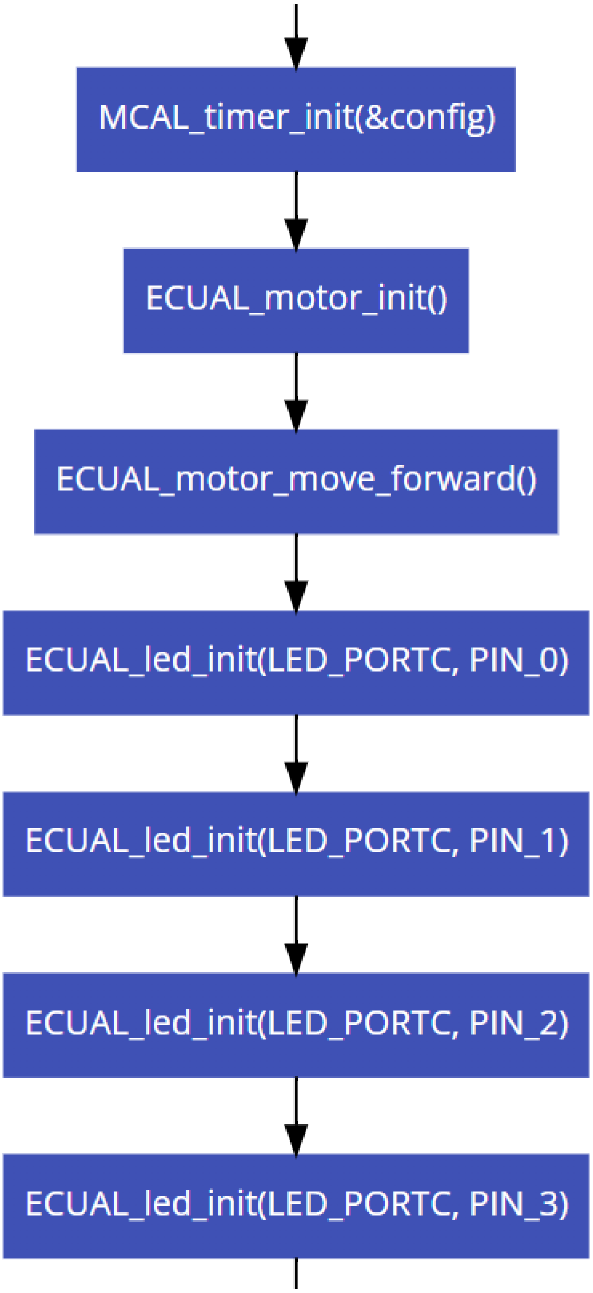

ISR(TIMER2_OVF_vect)
TCNT2 = -16
g_counter[2] += 1
g_duty_value[2] == MAX_DUTY_VALUE
False
True
g_counter[2] >= g_duty_value[2]
False
True
MCAL_dio_write_pin(PWM2_PORT, PIN_7, LOW)
MCAL_dio_write_pin(PWM2_PORT, PIN_7, HIGH)
g_counter[2] == MAX_DUTY_VALUE
True
g_counter[2] = 0
False
CLEAR_BIT(TIFR, 6)

Figure 29: ISR(TIMER2_OVF_vect) handler flowchart.

3.8-Application


application_status_t application_car_moving_system_init(void)
timer_config_t config
config.timerNumber = TIMER_NUMBER_0
config.mode = TIMER_MODE_NORMAL
config.clkSource = TIMER_CLK_SOURCE_INTERNAL
config.prescaler = TIMER_PRESCALER_8
config.interruptEnable = TIMER_INTERRUPT_DISABLE


MCAL_timer_init(&config)
ECUAL_motor_init()
ECUAL_motor_move_forward()
ECUAL_led_init(LED_PORTC, PIN_0)
ECUAL_led_init(LED_PORTC, PIN_1)
ECUAL_led_init(LED_PORTC, PIN_2)
ECUAL_led_init(LED_PORTC, PIN_3)

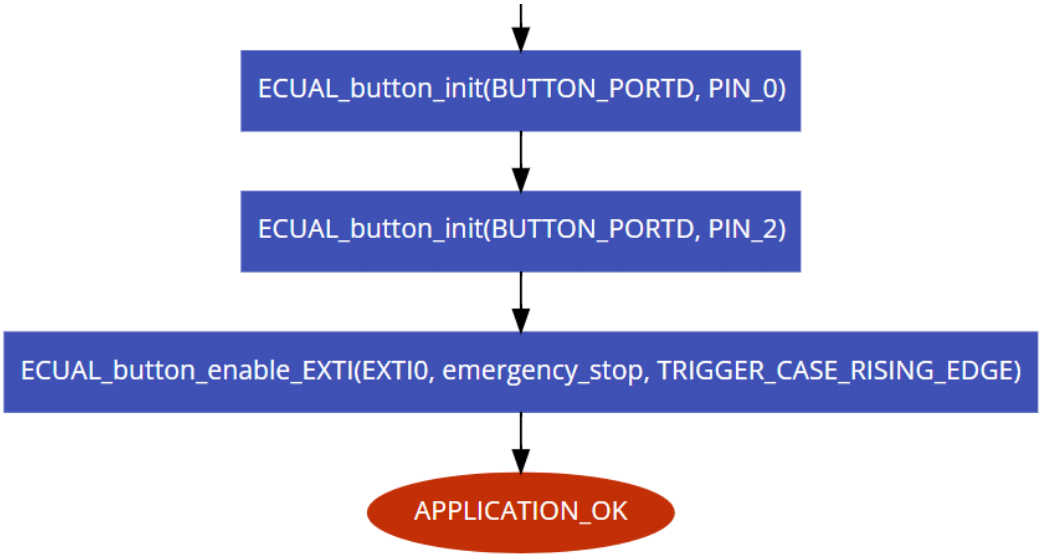

ECUAL_button_init(BUTTON_PORTD, PIN_0)
ECUAL_button_init(BUTTON_PORTD, PIN_2)
ECUAL_button_enable_EXTI(EXTI0, emergency_stop, TRIGGER_CASE_RISING_EDGE)
APPLICATION_OK

Figure 30: application_car_moving_system_init function flowchart.

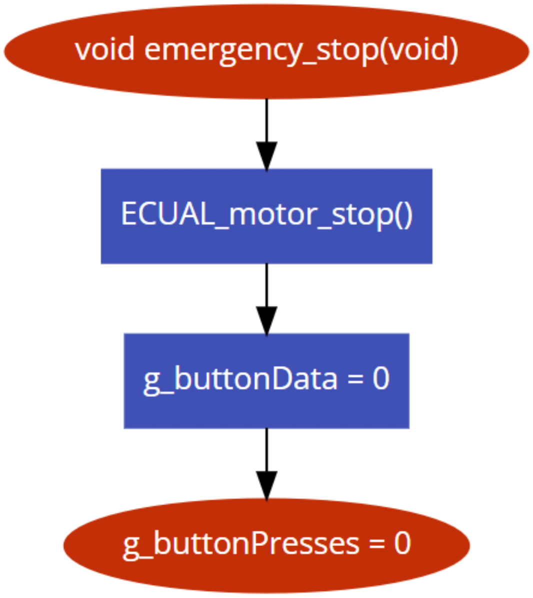
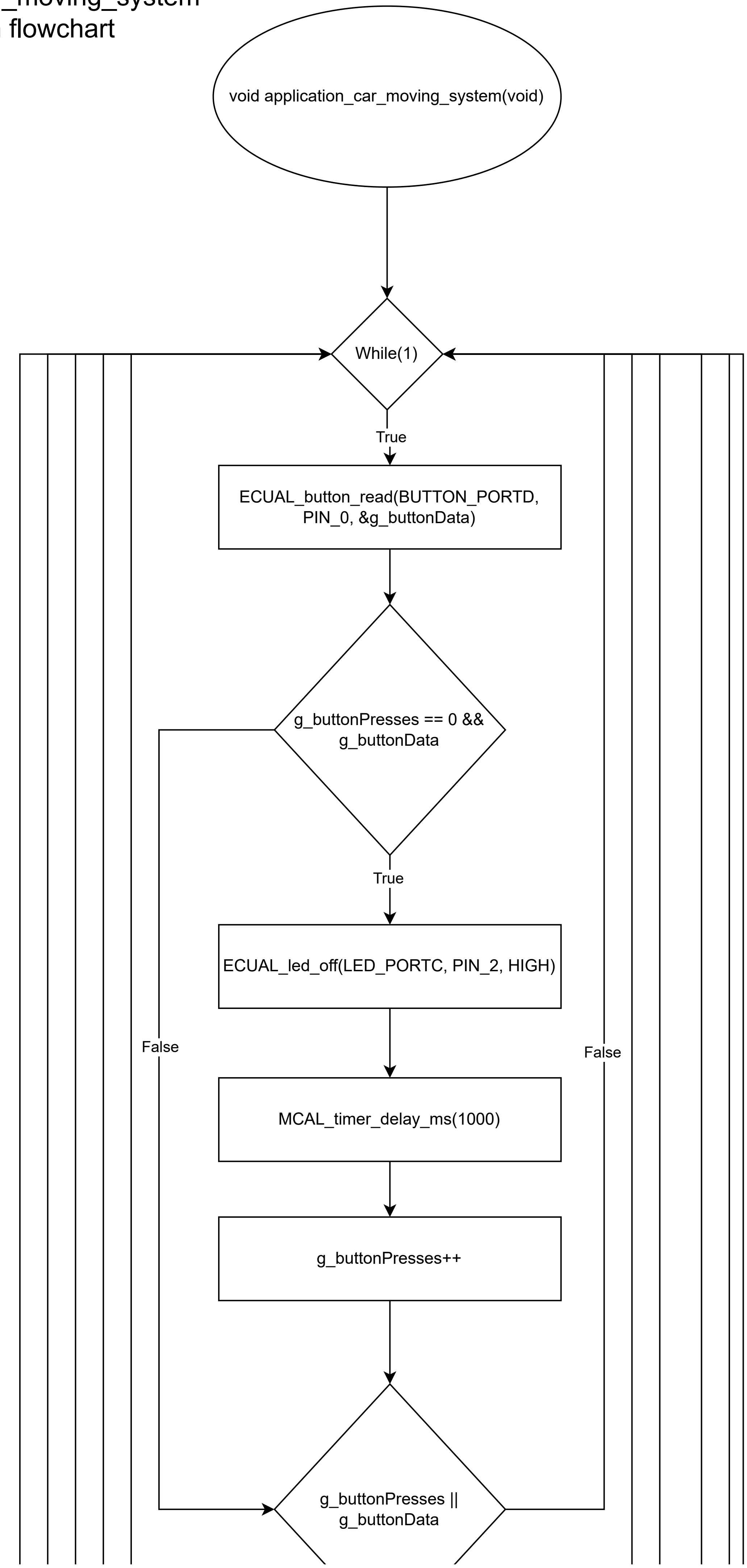
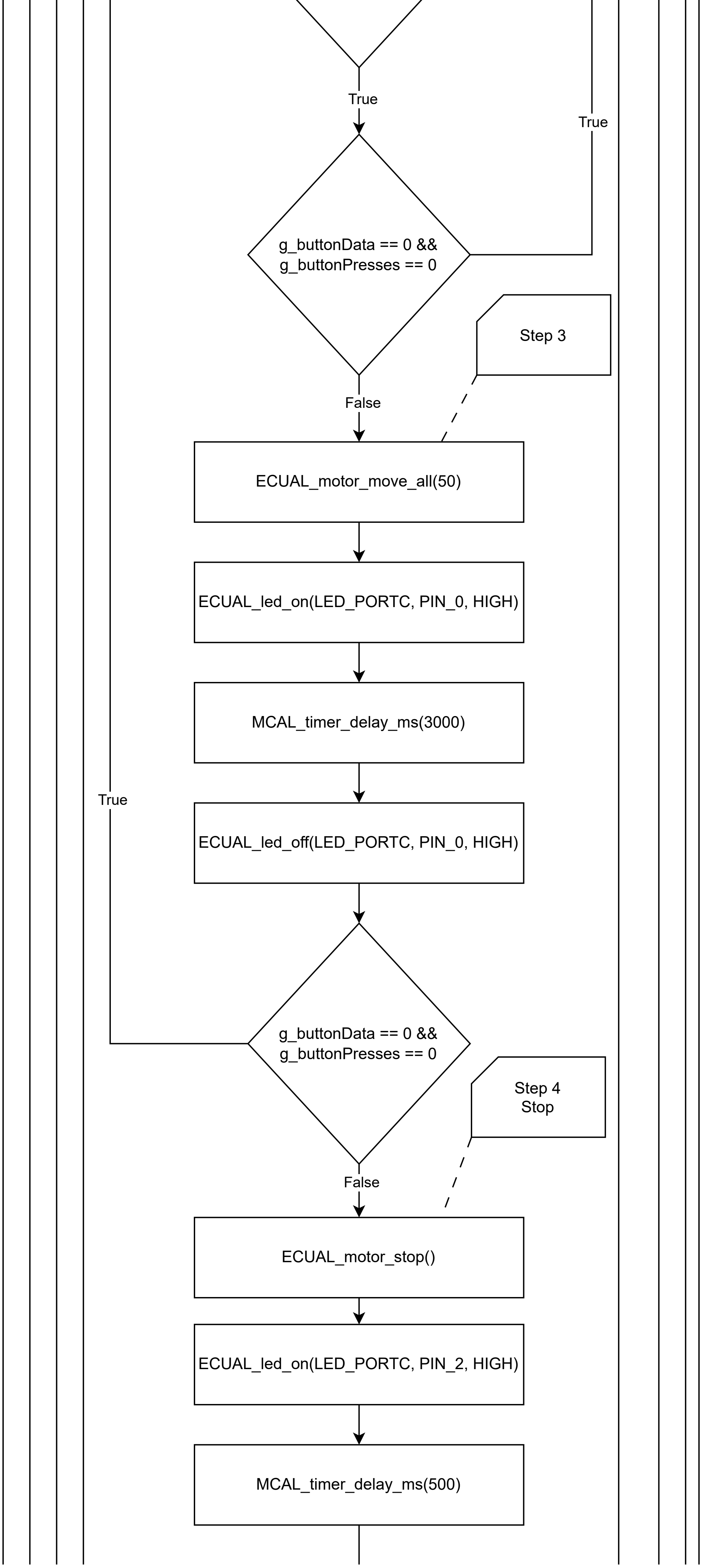
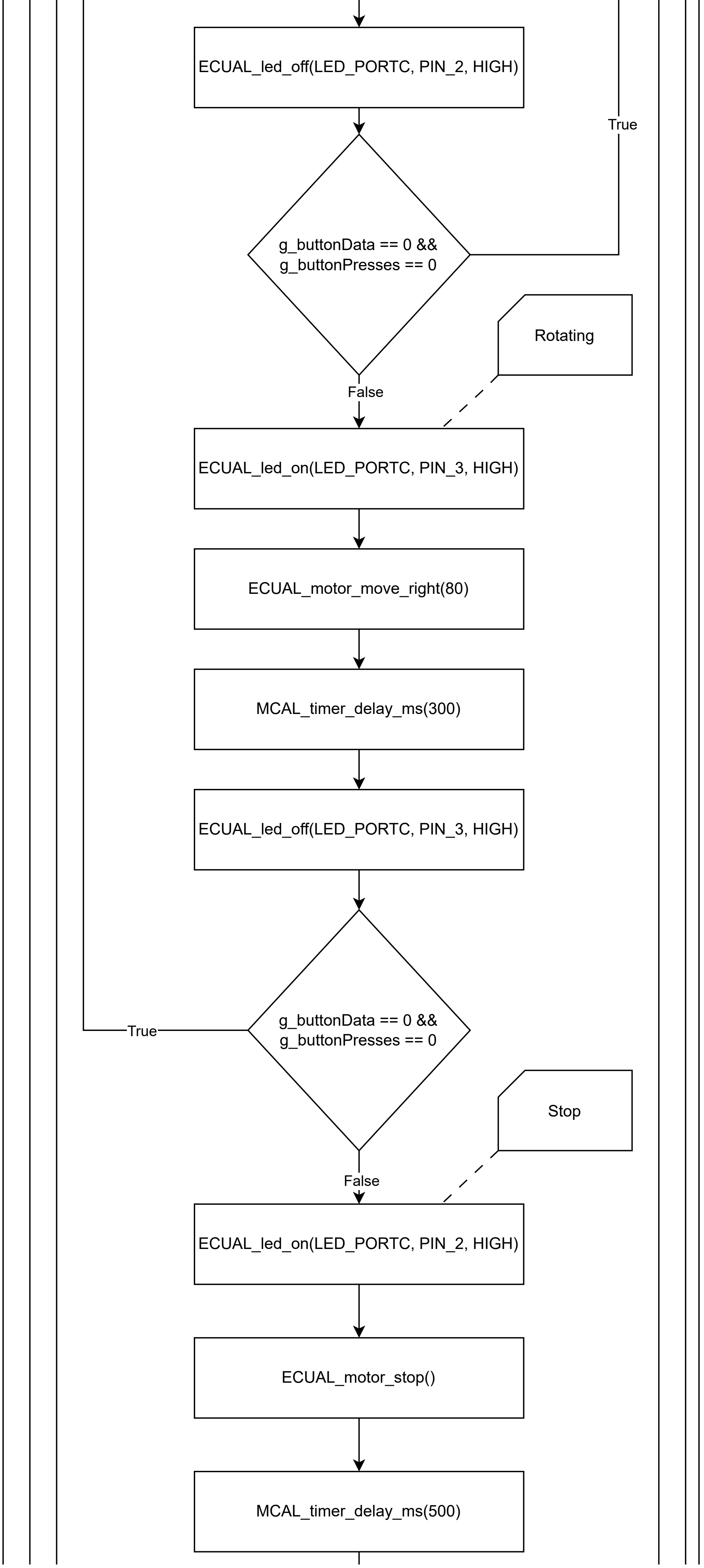


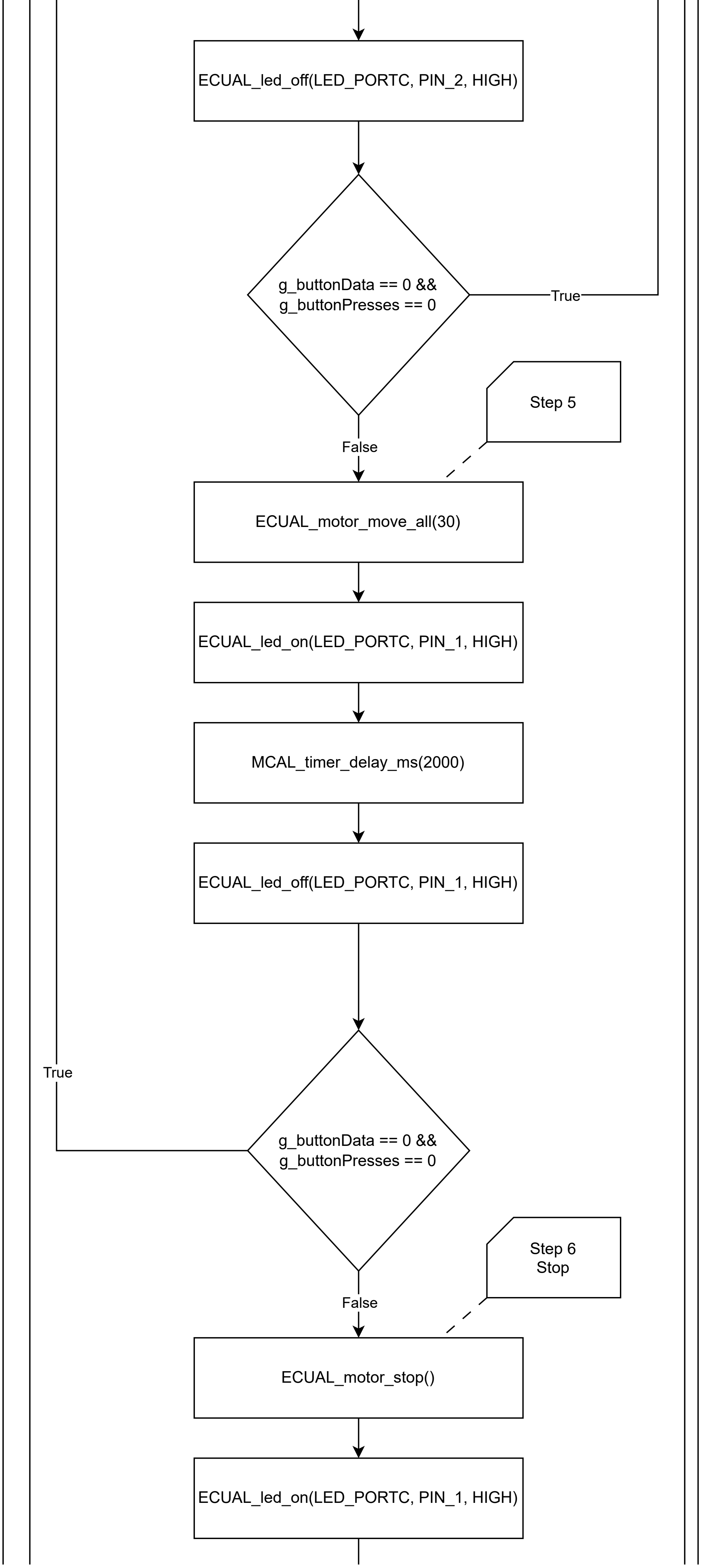
Figure 31: emergency_stop function flowchart.

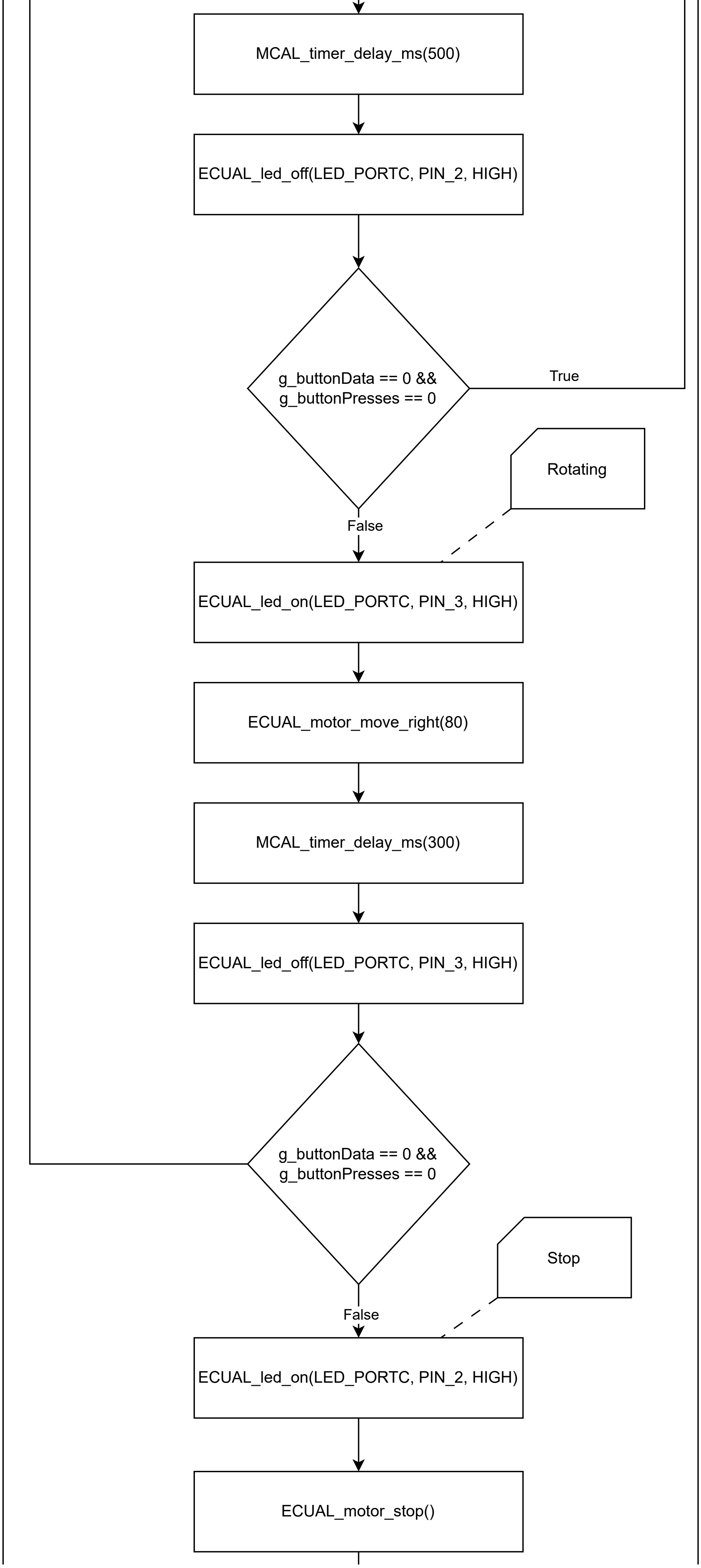
application_car_moving_system function flowchart

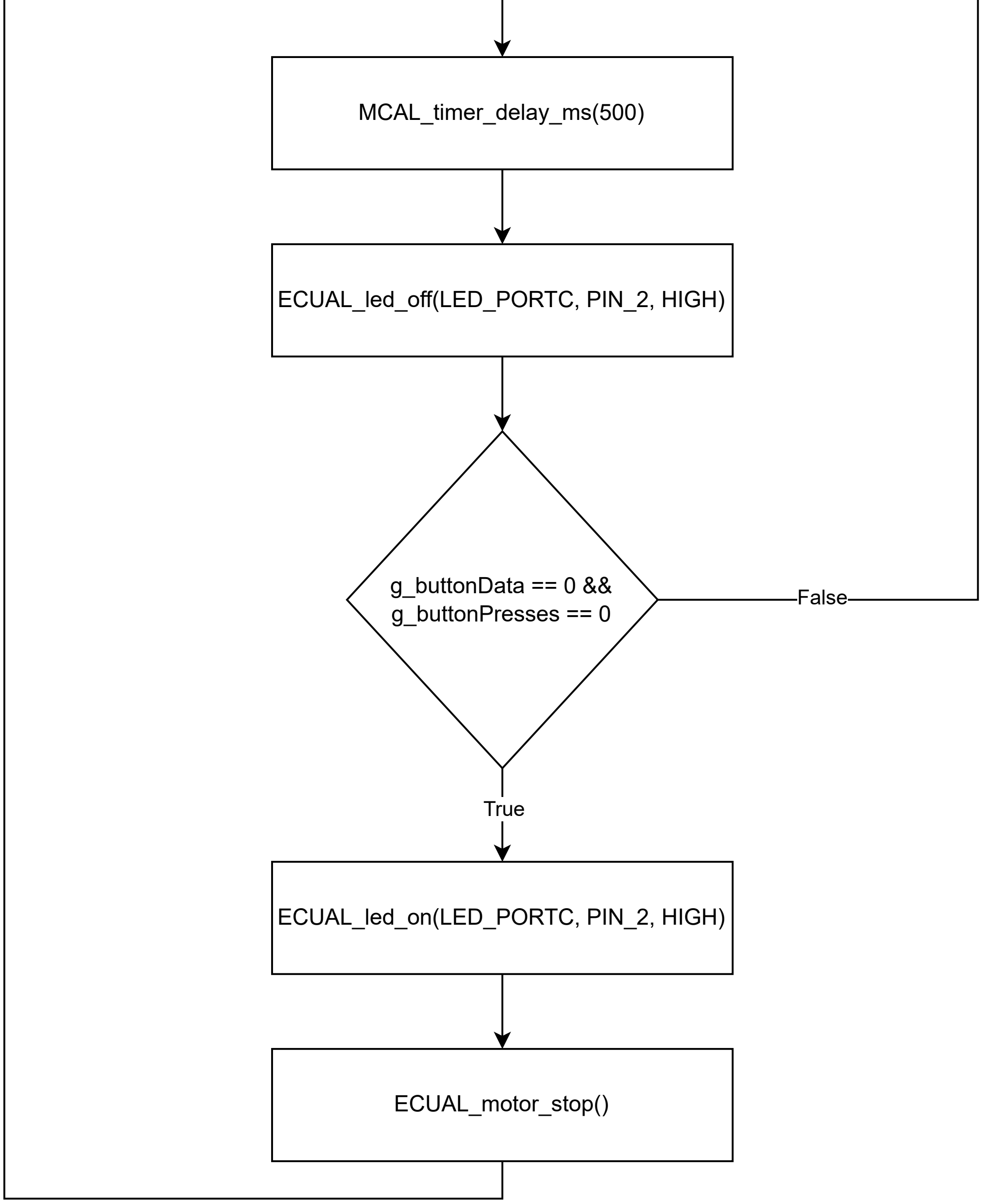

void application_car_moving_system(void)
While(1)
True
ECUAL_button_read(BUTTON_PORTD, PIN_0, &g_buttonData)
g_buttonPresses == 0 && g_buttonData
True
False
ECUAL_led_off(LED_PORTC, PIN_2, HIGH)
False
MCAL_timer_delay_ms(1000)
g_buttonPresses++
g_buttonPresses || g_buttonData


True
True
g_buttonData == 0 &&
g_buttonPresses == 0
Step 3
False
ECUAL_motor_move_all(50)
ECUAL_led_on(LED_PORTC, PIN_0, HIGH)
MCAL_timer_delay_ms(3000)
True
ECUAL_led_off(LED_PORTC, PIN_0, HIGH)
g_buttonData == 0 &&
g_buttonPresses == 0
Step 4
Stop
False
ECUAL_motor_stop()
ECUAL_led_on(LED_PORTC, PIN_2, HIGH)
MCAL_timer_delay_ms(500)


ECUAL_led_off(LED_PORTC, PIN_2, HIGH)
True
g_buttonData == 0 &&
g_buttonPresses == 0
Rotating
False
ECUAL_led_on(LED_PORTC, PIN_3, HIGH)
ECUAL_motor_move_right(80)
MCAL_timer_delay_ms(300)
ECUAL_led_off(LED_PORTC, PIN_3, HIGH)
True
g_buttonData == 0 &&
g_buttonPresses == 0
Stop
False
ECUAL_led_on(LED_PORTC, PIN_2, HIGH)
ECUAL_motor_stop()
MCAL_timer_delay_ms(500)


ECUAL_led_off(LED_PORTC, PIN_2, HIGH)
g_buttonData == 0 &&
g_buttonPresses == 0
True
False
Step 5
ECUAL_motor_move_all(30)
ECUAL_led_on(LED_PORTC, PIN_1, HIGH)
MCAL_timer_delay_ms(2000)
ECUAL_led_off(LED_PORTC, PIN_1, HIGH)
True
g_buttonData == 0 &&
g_buttonPresses == 0
Step 6
Stop
False
ECUAL_motor_stop()
ECUAL_led_on(LED_PORTC, PIN_1, HIGH)


MCAL_timer_delay_ms(500)
ECUAL_led_off(LED_PORTC, PIN_2, HIGH)
g_buttonData == 0 &&
g_buttonPresses == 0
True
Rotating
False
ECUAL_led_on(LED_PORTC, PIN_3, HIGH)
ECUAL_motor_move_right(80)
MCAL_timer_delay_ms(300)
ECUAL_led_off(LED_PORTC, PIN_3, HIGH)
g_buttonData == 0 &&
g_buttonPresses == 0
Stop
False
ECUAL_led_on(LED_PORTC, PIN_2, HIGH)
ECUAL_motor_stop()


MCAL_timer_delay_ms(500)
ECUAL_led_off(LED_PORTC, PIN_2, HIGH)
g_buttonData == 0 &&
g_buttonPresses == 0
False
True
ECUAL_led_on(LED_PORTC, PIN_2, HIGH)
ECUAL_motor_stop()