# 

[**1. Project Introduction 5**](#_z6v9fbnn765e)

[1.1. Project Components 5](#_czzplntdsr6s)

[**2. High Level Design 5**](#_qjle1zuq5ci8)

[2.1. System Architecture 5](#_d0qan0fnsle7)

[2.1.1. Definition 5](#_8ch9pqptg6ow)

[2.1.2. Layered Architecture 6](#_s0iw5nurzj4o)

[2.1.3. Design 7](#_4m61cnt1miyg)

[2.2. Modules Description 8](#_cpa505bqgcm)

[2.2.1. DIO (Digital Input/Output) Module 8](#_hnjxsalfo7r1)

[2.2.2. EXTINT Module 8](#_3sux29wdcgxa)

[2.2.3. Timer Module 8](#_czvvv9odb1tq)

[2.2.4. ICU Module 8](#_otjqn4if7itb)

[2.2.5. LCD Module 8](#_59lsgd4qziy9)

[2.2.6. Button Module 8](#_v3o1jlha9qz6)

[2.2.7. DCM Module 9](#_gvr36vhor878)

[2.2.8. Keypad Module 9](#_kb47cw1vprzv)

[2.2.9. Delay Module 9](#_alddapoycdma)

[2.2.10. PWM Module 9](#_v79u55rxtk9d)

[2.2.11. Ultrasonic Module 9](#_g0iduevurchh)

[2.3. Drivers’ Documentation (APIs) 10](#_c6zmo2sgkvt5)

[2.3.1. Definition 10](#_caglhjniy5nw)

[2.3.2. MCAL APIs 10](#_ldgx3ow0urv)

[2.3.2.1. DIO Driver 10](#_semcnkaqaxsr)

[2.3.2.2. EXTINT Driver 12](#_7euq96x6p5jq)

[2.3.2.3. TIMER Driver 13](#_wh0pgylns092)

[2.3.3. HAL APIs 18](#_71wwpvn2lhrt)

[2.3.3.1. LCD APIs 18](#_g8ans492cifz)

[2.3.3.2. BUTTON APIs 20](#_btwdn2re26jf)

[2.3.3.3. DCM APIs 20](#_9naemocgye6o)

[2.3.3.4. KEYPAD APIs 22](#_v96mp9e5p0bv)

[2.3.3.5. ICU APIs 22](#_b881u1sy0n3i)

[2.3.3.6. Ultrasonic APIs 23](#_pq6uiycgwu9v)

[2.3.3.7. Delay APIs 24](#_xjqks6n0ud4b)

[2.3.3.8. PWM APIs 25](#_yskllqba3lus)

[**3. Low Level Design 27**](#_phtdxn3ttdmz)

[3.1. MCAL Layer 27](#_s58865paflhl)

[3.1.1. DIO Module 27](#_c2rsi8ahde9g)

[3.1.1.1. DIO\_init 27](#_r53q3elrqgry)

[3.1.1.2. DIO\_write 28](#_rdr31cc6vxlz)

[3.1.1.3. DIO\_read 29](#_x95erzeku47f)

[3.1.1.4. DIO\_toggle 30](#_yr8rnyviy6ty)

[3.1.1.5. DIO\_pinPullUp 31](#_oxhtnvj73bbn)

[3.1.2. EXTINT Module 32](#_j5tqziyk5jto)

[3.1.2.1. EXTINT\_init 32](#_kr7wne9o8jpm)

[3.1.2.2. EXTINT\_setCallBack 33](#_e0itjo8q0qr1)

[3.1.2.3. ISR(EXT\_INT\_#) 34](#_nz6rutt65p3n)

[3.1.3. Timer Module 35](#_gsaip9mwk6p)

[3.1.3.1. sub process 35](#_65bzi1o6gx7g)

[3.1.3.2. TIMER\_init 36](#_xl2i1cspdtvo)

[3.1.3.3. TIMER\_setTime 37](#_u3dz52omwtfc)

[3.1.3.4. TIMER\_pwmGenerator 38](#_bmbo8tcs0rmn)

[3.1.3.5. TIMER\_resume 39](#_o5vus0xe32mn)

[3.1.3.6. TIMER\_pause 39](#_i5vpc7qszj88)

[3.1.3.7. TIMER\_reset 40](#_dhbw0oxa6yii)

[3.1.3.8. TIMER\_enableInterrupt 41](#_m0q06r65t5tb)

[3.1.3.9. TIMER\_enableInterrupt 41](#_cdp4v45jkzh1)

[3.1.3.10. TIMER\_setCallBack 42](#_ic4vi6947eg2)

[3.1.3.11. TIMER\_setPwmOnCallBack 43](#_3233l744cunk)

[3.1.3.12. TIMER\_setPwmOffCallBack 44](#_6brp5lt2ywwc)

[3.1.3.13. TIMER\_getElapsedTime 45](#_uqnat0n9b30i)

[3.1.3.14. TIMER\_setDelayTime 46](#_emtoh70fl2i)

[3.2. HAL Layer 48](#_4cgysolo5f1)

[3.2.1. ICU Module (Input Capture Unit) 48](#_1wg7tomwust7)

[3.2.1.1. ICU\_init 48](#_xv4dqus84jzu)

[3.2.1.2. ICU\_getElapsedTime 49](#_wmllv545kc4o)

[3.2.1.3. ICU\_task 50](#_52p5i1x16do3)

[3.2.2. LCD Module 51](#_nx76v9s479lc)

[3.2.2.1. LCD\_init 51](#_hkrozmwkixbn)

[3.2.2.2. LCD\_writeChar 53](#_f0wznukrusoa)

[3.2.2.3. LCD\_writeCmd 54](#_dvvn5bhe05kl)

[3.2.2.4. LCD\_clr 55](#_nb1fkq2es2i5)

[3.2.2.5. LCD\_goTo 56](#_u8yt5qp6zhl0)

[3.2.3. BTN Module 57](#_i430gknpgkin)

[3.2.3.1. BUTTON\_init 57](#_i3hkdqd053fm)

[3.2.3.2. BUTTON\_read 58](#_so15r7bed303)

[3.2.4. DCM Module 59](#_31v3lx9uszos)

[3.2.4.1. DCM\_init 59](#_lmgna0w9dnne)

[3.2.4.2. DCM\_setDirection 60](#_8xnxyx1v98zg)

[3.2.4.3. DCM\_speed 61](#_scgwa253zdun)

[3.2.4.4. DCM\_start 62](#_dttrived2zay)

[3.2.4.5. DCM\_stop 63](#_fx8inh3dfkzj)

[3.2.5. Ultrasonic Module 64](#_jnrv0yw5nkz6)

[3.2.5.1. US\_init 64](#_eymet1o9bdc7)

[3.2.5.2. US\_getDistance 65](#_64u8hyrlqxm9)

[3.2.6. Delay Module 66](#_bsusxyu6jpu7)

[3.2.6.1. DELAY\_init 66](#_yc3f00s4t8av)

[3.2.6.2. DELAY\_setTime 67](#_iy1f5znj1840)

[3.2.6.3. DELAY\_setTimeBlocking 68](#_b8rxfeaiegt1)

[3.2.6.4. DELAY\_setTimeBlocking 69](#_psmh2mnmzffj)

[3.2.7. PWM Module 70](#_kmqm9azffsqd)

[3.2.7.1. PWM\_init 70](#_vjfwnz7ki9f)

[3.2.7.2. PWM\_setDutyCycle 71](#_1zdwpqk05fk5)

[3.2.7.3. PWM\_start 72](#_mf4y0jwn2kvv)

[3.2.7.4. PWM\_stop 73](#_nsb0zk56wyt)

[3.2.7.5. PWM\_onTask 74](#_217mj0cxadsh)

[3.2.7.6. PWM\_offTask 74](#_afjbxrxy9kr9)

[3.3. APP Layer 75](#_y3c3qkldz1ay)

[3.3.1. App State Diagram 75](#_y4tp1ksaglw8)

[**4. Pre-compiling and linking configurations 77**](#_39o97bt561y)

[4.1. EXTINT Driver 77](#_jfvqn9e8fadd)

[4.1.1. Linking Configurations 77](#_uuq0wbl9xuzp)

[4.1.2. Pre\_compiling Configurations 77](#_xwv57oli6fw6)

[4.2. Timer Driver 77](#_3q2gl5bgovs1)

[4.2.1. Linking configurations 77](#_yt3e23o4t51k)

[4.2.2. Pre-compiled Configurations 78](#_vk28ivit1qh1)

[4.3. ICU Driver 78](#_34bkykwpecf1)

[4.3.1. Linking Configuration 78](#_ww5nh763ozgg)

[4.4. PWM Driver 79](#_rd2tgtybk0mz)

[4.4.1. Linking Configurations 79](#_e1js0nova4a8)

[4.4.2. Pre-compiled Configurations 79](#_cfj7fucj5oq6)

[4.5. Delay Driver 80](#_15c3aov03ywo)

[4.5.1. Pre-compiled Configurations 80](#_um6hzh7vuxix)

[4.6. DCM Driver 80](#_umicbblase0d)

[4.6.1. Linking Configurations 80](#_rwee08yhu2cs)

[4.6.2. Pre-compiled Configurations 80](#_9jjpqck4gc0u)

[4.7. US (ultrasonic) Driver 80](#_3gq9khdvacph)

[4.7.1. Linking Configuration 80](#_7dcijf73t59u)

[4.8. LCD Driver 81](#_czuiupkcz5s9)

[4.8.1. pre-compiling configuration 81](#_mn6n3xw7urrj)

[4.9. KEYPAD Driver 81](#_88qejrgia0a4)

[4.9.1. pre-compiling configuration 81](#_kau22bgeg0gr)

[4.9.2. Linking configurations 82](#_mekwi5git8ok)

[4.10. BUTTON Driver 82](#_6vspywpjk9h6)

[4.10.1. pre-compiling configuration 82](#_794t34kmmesz)

[4.10.2. Linking configurations 82](#_c9ku8d31x7f2)

OBSTACLE AVOIDANCE CAR DESIGN

# Project Introduction

Obstacle avoidance car is an embedded systems project which is using Atmega32 microcontroller to control a bunch of sensors and dc motors to make the car avoid crashing when it finds obstacle in front of it and it will control it’s direction to detect the best route to move through it.

## Project Components

* ATmega32 microcontroller
* Four motors (M1, M2, M3, M4)
* One button to change default rotation direction (PBUTTON0)
* Keypad 3x3 (KPD1: start, KPD2: stop)
* One ultrasonic sensor
  + Vcc to 5v
  + GND to GND
  + Trig to PB3 (INT1)
  + Echo to PB2 (INT0)
* LCD 2x16

# High Level Design

## System Architecture

### Definition

Layered Architecture (Figure 1) describes an architectural pattern composed of several separate horizontal layers that function together as a single unit of software.

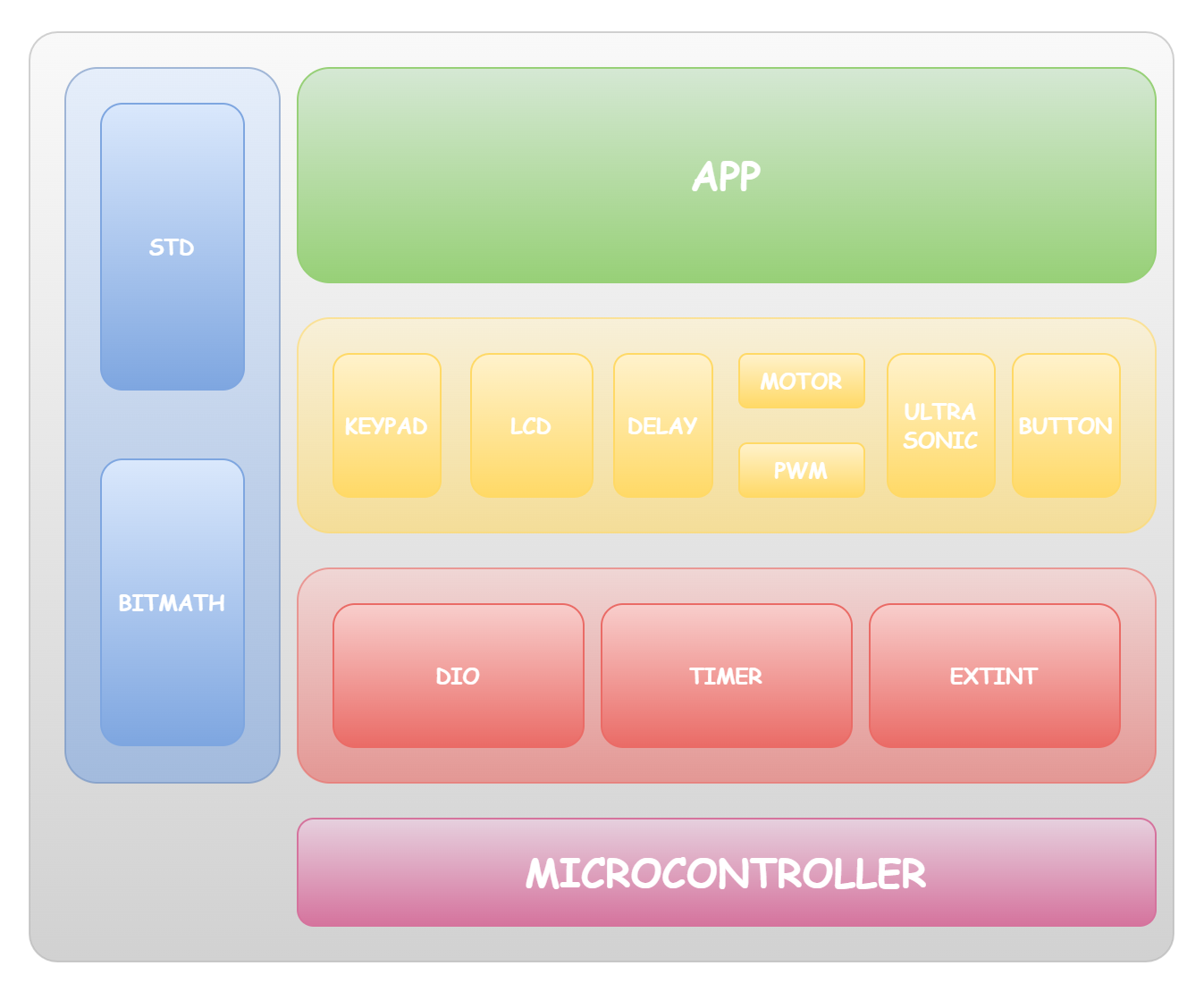
Microcontroller Abstraction Layer (MCAL) is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU.

Hardware Abstraction Layer (HAL) is a layer of programming that allows a computer OS to interact with a hardware device at a general or abstract level rather than at a detailed hardware level.

### Layered Architecture



### Design



## Modules Description

### DIO (Digital Input/Output) Module

The DIO module is responsible for reading input signals from the system's sensors (such as buttons) and driving output signals to the system's actuators (such as LEDs). It provides a set of APIs to configure the direction and mode of each pin (input/output, pull-up/down resistor), read the state of an input pin, and set the state of an output pin.

### EXTINT Module

The EXI (External Interrupt) module is responsible for detecting external events that require immediate attention from the microcontroller, such as a button press. It provides a set of APIs to enable/disable external interrupts for specific pins, set the interrupt trigger edge (rising/falling/both), and define an interrupt service routine (ISR) that will be executed when the interrupt is triggered.

### Timer Module

The TIMER module is responsible for generating timing events that are used by other modules in the system. It provides a set of APIs to configure the timer clock source and prescaler, set the timer mode (count up/down), set the timer period, enable/disable timer interrupts, and define an ISR that will be executed when the timer event occurs.

### ICU Module

The ICU Module is a software component designed to interface with an Ultrasonic sensor in our project. It enables the accurate detection of the distance between the car and surrounding objects in all directions. By leveraging input capture techniques, the ICU Module captures the sensor's readings and provides real-time distance measurements. This crucial information aids in implementing a comprehensive collision avoidance system, ensuring safe navigation for the vehicle in various scenarios.

### LCD Module

LCD stands for "Liquid Crystal Display," which is a type of flat-panel display used in electronic devices to display text and graphics. We're using the 4-bit mode to reduce the number of I/O pins needed to interface with the LCD. The module includes a controller, a display, and a backlight. By interfacing with the LCD module and writing the necessary code, we're able to provide the user with real-time information about the current speed and direction of the car as well as providing an interface to set the default rotation direction.

### Button Module

The BTN (Button) module is responsible for reading the state of the system's buttons. It provides a set of APIs to enable/disable button interrupts, set the button trigger edge (rising/falling/both), and define an ISR that will be executed when a button press is detected.

### DCM Module

The DCM (DC Motor) module is responsible for controlling the speed and direction of the system's DC motors. It provides a set of APIs to set the speed and direction of each motor, and to stop all motors. It also uses the TIMER module to generate PWM (Pulse Width Modulation) signals that control the motor speed.

### Keypad Module

Keypad is an analog switching device which is generally available in matrix structure. It is used in many embedded system applications for allowing the user to perform a necessary task. A matrix Keypad consists of an arrangement of switches connected in matrix format in rows and columns. The rows and columns are connected with a microcontroller such that the rows of switches are connected to one pin and the columns of switches are connected to another pin of a microcontroller.

### Delay Module

Delay is a software-implemented module that uses a normal timer to generate both non-blocking delays and blocking delays. it also sends to the timer call back function to be called when the timer interrupt occurs.

### PWM Module

PWM is a software-implemented module that uses a normal timer to generate pulse width module signals through chosen pins. It has the accessibility to edit the duty cycle of the output signal too.

### Ultrasonic Module

The Ultrasonic module is a key component in our project for distance sensing. It utilizes ultrasonic waves to measure distances between the module and surrounding objects. By emitting ultrasonic pulses and calculating the time taken for the echoes to return, the module provides accurate distance measurements. Its compact size, low power consumption, and wide detection range make it an ideal choice for collision avoidance applications. With its reliable performance and easy integration, the Ultrasonic module enhances the safety and precision of our project's distance sensing capabilities.

### 

## Drivers’ Documentation (APIs)

### Definition

An API is an Application Programming Interface that defines a set of routines, protocols and tools for creating an application. An API defines the high level interface of the behavior and capabilities of the component and its inputs and outputs.

An API should be created so that it is generic and implementation independent. This allows for the API to be used in multiple applications with changes only to the implementation of the API and not the general interface or behavior.

### MCAL APIs

#### DIO Driver

|  |
| --- |
| |  | Syntax : DIO\_init(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 u8\_a\_direction)  | Description : Initializes DIO pin  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_portNumber  | u8 u8\_a\_pinNumber  | u8 u8\_a\_direction  | Parameters (out): None  | Return value: : en\_DIO\_error\_t DIO\_OK = 0  | WRONG\_PORT\_NUMBER = 1  | WRONG\_PIN\_NUMBER = 2  | WRONG\_DIRECTION = 4  |  en\_DIO\_error\_t DIO\_init(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 u8\_a\_direction);   |  | Syntax : DIO\_write(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 u8\_a\_value)  | Description : Assigns DIO pins LEVEL  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_portNumber  | u8 u8\_a\_pinNumber  | u8 u8\_a\_value  | Parameters (out): None  | Return value: : en\_DIO\_error\_t DIO\_OK = 0  | WRONG\_PORT\_NUMBER = 1  | WRONG\_PIN\_NUMBER = 2  | WRONG\_VALUE =3  |  en\_DIO\_error\_t DIO\_write(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 u8\_a\_value);  |  | Syntax : DIO\_toggle(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber)  | Description : Toggles DIO pins LEVEL  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_portNumber  | u8 u8\_a\_pinNumber  | Parameters (out): None  | Return value: : en\_DIO\_error\_t DIO\_OK = 0  | WRONG\_PORT\_NUMBER = 1  | WRONG\_PIN\_NUMBER = 2  |  en\_DIO\_error\_t DIO\_toggle(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber);  |  | Syntax : DIO\_read(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8u8\_a\_value)  | Description : Gets DIO pins LEVEL  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_portNumber  | u8 u8\_a\_pinNumber  | Parameters (out): u8 u8\_a\_value  | Return value: : en\_DIO\_error\_t DIO\_OK = 0  | WRONG\_PORT\_NUMBER = 1  | WRONG\_PIN\_NUMBER = 2  |  en\_DIO\_error\_t DIO\_read(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 \*u8\_a\_value);  |  | Syntax : DIO\_pinPullUp(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 u8\_a\_pullUpState);  | Description : Assigns DIO pin's pullup state  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_portNumber  | u8 u8\_a\_pinNumber  | u8 u8\_a\_pullUpState  | Parameters (out): None  | Return value: : en\_DIO\_error\_t DIO\_OK = 0  | WRONG\_PORT\_NUMBER = 1  | WRONG\_PIN\_NUMBER = 2  | DIO\_NOK = 5  |  en\_DIO\_error\_t DIO\_pinPullUp(u8 u8\_a\_portNumber, u8 u8\_a\_pinNumber, u8 pullUpState); |

#### 

#### 

#### EXTINT Driver

|  |
| --- |
| |  | Syntax : EXTINT\_init (void)  | Description : Initializes External Interrupts  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_EXTINT\_error\_t EXTINT\_OK = 0  | EXTINT\_NOK = 1  | en\_EXTINT\_error\_t EXTINT\_init (void);  |  | Syntax : EXTINT\_setCallBackInt (u8 u8\_a\_intNumber, void (\*funPtr) (void))  | Description : Sets callback function for a specific external interrupt  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_intNumber  | void (\*funPtr) (void)  | Parameters (out): None  | Return value: : en\_EXTINT\_error\_t EXTINT\_OK = 0  | EXTINT\_NOK = 1  | en\_EXTINT\_error\_t EXTINT\_setCallBackInt (u8 u8\_a\_intNumber, void (\*funPtr) (void)); |

#### 

#### TIMER Driver

|  |
| --- |
| |  | **Syntax** : en\_TIMER\_error\_t TIMER\_init( void )  | **Description** : Initialize Timer according to preprocessed  | configured definitions  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_init( void );  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_setTime  | (en\_TIMER\_number\_t en\_a\_timerUsed, f32 f32\_a\_desiredTime)  | **Description** : set the time at which the timer interrupts  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | f32 f32\_a\_desiredTime  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_setTime(en\_TIMER\_number\_t en\_a\_timerUsed, f32 f32\_a\_desiredTime);  |  | **Syntax**  : en\_TIMER\_error\_t TIMER\_pwmGenerator  | (en\_TIMER\_number\_t en\_a\_timerUsed , u16 u16\_a\_onTime, u16 u16\_a\_offTime)  | **Description** : initialize the timer to generates pwm signal using  | normal mode  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | u16 u16\_a\_onTime  | u16 u16\_a\_offTime  | **Parameters (out)**: None  | **Return value:**  : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_pwmGenerator(en\_TIMER\_number\_t en\_a\_timerUsed , u16 u16\_a\_onTime, u16 u16\_a\_offTime);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_resume(en\_TIMER\_number\_t en\_a\_timerUsed)  | **Description** : makes the timer to start/resume counting  | **Sync\Async** : Synchronous  | **Reentrancy**  : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_resume(en\_TIMER\_number\_t en\_a\_timerUsed);  |  | **Syntax**  : en\_TIMER\_error\_t TIMER\_reset(en\_TIMER\_number\_t en\_a\_timerUsed)  | **Description** : makes the timer to reset counting from the beginning  | **Sync\Async**  : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: None  | **Return value:**  : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_reset(en\_TIMER\_number\_t en\_a\_timerUsed);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_getElapsedTime  | (en\_TIMER\_number\_t en\_a\_timerUsed, u32\* u32\_a\_elapsedTime)  | **Description** : returns the elapsed time since the timer started  | from the beginning in microseconds  | **Sync\Async**  : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: u32 u32\_a\_elapsedTime  | **Return value:**  : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_getElapsedTime(en\_TIMER\_number\_t en\_a\_timerUsed, u32\* u32\_a\_elapsedTime);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_pause(en\_TIMER\_number\_t en\_a\_timerUsed)  | **Description** : makes the timer to pause counting  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_pause(en\_TIMER\_number\_t en\_a\_timerUsed);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_disableInterrupt(en\_TIMER\_number\_t en\_a\_timerUsed)  | **Description** : Disables timer's interrupts  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_disableInterrupt(en\_TIMER\_number\_t en\_a\_timerUsed);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_enableInterrupt(en\_TIMER\_number\_t en\_a\_timerUsed)  | **Description** : Enables timer's interrupts  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_enableInterrupt(en\_TIMER\_number\_t en\_a\_timerUsed);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_setCallBack  | (en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void))  | **Description** : sets the call back function for a specific timer  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | void (\*funPtr)(void)  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_setCallBack(en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void));  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_setDelayTime  | (en\_TIMER\_number\_t en\_a\_timerUsed, f32 f32\_a\_timeInMS)  | **Description** : Set delay time for blocking delay using a specific timer  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | f32 f32\_a\_timeInMS  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_setDelayTime(en\_TIMER\_number\_t en\_a\_timerUsed, f32 f32\_a\_timeInMS);  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_setPwmOnCallBack  | (en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void))  | **Description** : Set callback function for the task done while signal is high  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | void (\*funPtr)(void)  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  |  en\_TIMER\_error\_t TIMER\_setPwmOnCallBack(en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void));  |  | **Syntax** : en\_TIMER\_error\_t TIMER\_setPwmOffCallBack  | (en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void))  | **Description** : Set callback function for the task done while signal is low  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_TIMER\_number\_t en\_a\_timerUsed  | void (\*funPtr)(void)  | **Parameters (out)**: None  | **Return value:** : en\_TIMER\_error\_t TIMER\_OK = 0  | TIMER\_WRONG\_TIMER\_USED = 1  | TIMER\_WRONG\_DESIRED\_TIME = 2  | TIMER\_NOK = 3  | en\_TIMER\_error\_t TIMER\_setPwmOffCallBack(en\_TIMER\_number\_t en\_a\_timerUsed, void (\*funPtr)(void)); |

### 

### 

### HAL APIs

#### LCD APIs

|  |
| --- |
| |  | Syntax : LCD\_init (void)  | Description : Initializes LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_init (void);  |  | Syntax : LCD\_writeChar (u8 u8\_a\_char)  | Description : writes character on LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_char  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_writeChar (u8 u8\_a\_char);  |  | Syntax : LCD\_writeCmd(u8 u8\_a\_cmd)  | Description : Do command on LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_cmd  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_writeCmd (u8 u8\_a\_cmd);  |  | Syntax : LCD\_writeNum (s32 s32\_a\_num)  | Description : Writes an signed number on LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 s32\_a\_num  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_writeNum (s32 s32\_a\_num);  |  | Syntax : LCD\_clr(void)  | Description : Clears LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_clr (void);  |  | Syntax : LCD\_writeString (u8\* u8\_a\_string)  | Description : Writes a string on LCD  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8\* u8\_a\_string  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_writeString (u8\* u8\_a\_string);  |  | Syntax : LCD\_goTo(u8 u8\_a\_row, u8 u8\_a\_Col)  | Description : Moves cursor on lcd to the desired location  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_row,  | u8 u8\_a\_Col  | Parameters (out): None  | Return value: : en\_LCD\_error\_t LCD\_OK = 0  | LCD\_NOK = 1  |  en\_LCD\_error\_t LCD\_goTo (u8 u8\_a\_row, u8 u8\_a\_Col); |

#### 

#### 

#### BUTTON APIs

|  |
| --- |
| |  | Syntax : BUTTON\_init (void)  | Description : Initializes buttons  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_BUTTON\_error\_t BUTTON\_OK = 0  | WRONG\_BUTTON\_PORT = 1  | WRONG\_BUTTON\_PIN = 2  | WRONG\_BUTTON\_NUMBER = 3  |  en\_BUTTON\_error\_t BUTTON\_init(void);  |  | Syntax : BUTTON\_read(u8 u8\_a\_buttonNumber, u8\* u8\_a\_buttonState)  | Description : Gets a specific button's state  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : u8 u8\_a\_buttonNumber  | Parameters (out): u8\* u8\_a\_buttonState  | Return value: : en\_BUTTON\_error\_t BUTTON\_OK = 0  | WRONG\_BUTTON\_PORT = 1  | WRONG\_BUTTON\_PIN = 2  | WRONG\_BUTTON\_NUMBER = 3  | en\_BUTTON\_error\_t BUTTON\_read(u8 u8\_a\_buttonNumber, u8 \*u8\_a\_buttonState); |

#### DCM APIs

|  |
| --- |
| |  | **Syntax**  : en\_DCM\_error\_t DCM\_init (void)  | **Description**  : Initializes DCM module  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_DCM\_error\_t DCM\_OK = 0  | DCM\_NOK = 1  |  en\_DCM\_error\_t DCM\_init (void);  | **Syntax** : en\_DCM\_error\_t DCM\_setDirection  | (en\_DCM\_number\_t en\_a\_dcmNumber, en\_DCM\_direction\_t en\_a\_direction)  | **Description** : Sets Directions for the a specific DCM  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : en\_DCM\_number\_t en\_a\_dcmNumber  | en\_DCM\_direction\_t en\_a\_direction  | **Parameters (out)**: None  | **Return value:** : en\_DCM\_error\_t DCM\_OK = 0  | DCM\_NOK = 1  |  en\_DCM\_error\_t DCM\_setDirection (en\_DCM\_number\_t en\_a\_dcmNumber, en\_DCM\_direction\_t en\_a\_direction);  |  | **Syntax** : en\_DCM\_error\_t DCM\_speed (u8 u8\_a\_speed)  | **Description** : Sets speed for DCMs  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : u8 u8\_a\_speed  | **Parameters (out)**: None  | **Return value:** : en\_DCM\_error\_t DCM\_OK = 0  | DCM\_NOK = 1  |  en\_DCM\_error\_t DCM\_speed (u8 u8\_a\_speed);  |  | **Syntax** : en\_DCM\_error\_t DCM\_start (void)  | **Description** : Starts DCMs to rotate  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_DCM\_error\_t DCM\_OK = 0  | DCM\_NOK = 1  |  en\_DCM\_error\_t DCM\_start (void);  |  | **Syntax** : en\_DCM\_error\_t DCM\_stop (void)  | **Description** : Stops DCMs from rotating  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_DCM\_error\_t DCM\_OK = 0  | DCM\_NOK = 1  |  en\_DCM\_error\_t DCM\_stop (void); |

#### 

#### KEYPAD APIs

|  |
| --- |
| |  | Syntax : KEYPAD\_init (void)  | Description : Initializes keypad  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_KEYPAD\_error\_t KEYPAD\_OK = 0  | KEYPAD\_NOK = 1  |  en\_KEYPAD\_error\_t KEYPAD\_init (void);  |  | Syntax : KEYPAD\_read (u8\* u8\_a\_keyPressed)  | Description : gets the value of key pressed  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): u8\* u8\_a\_keyPressed  | Return value: : en\_KEYPAD\_error\_t KEYPAD\_OK = 0  | KEYPAD\_NOK = 1  |  en\_KEYPAD\_error\_t KEYPAD\_read (u8 \*u8\_a\_keyPressed); |
|  |

#### ICU APIs

|  |
| --- |
| |  | Syntax : ICU\_init (void)  | Description : Initializes ICU  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_ICU\_error\_t ICU\_OK = 0  | ICU\_NOK = 1  |  en\_ICU\_error\_t ICU\_init (void);  |  | Syntax : ICU\_getElapsedTime (u32 \*u32\_a\_elapsedTime)  | Description : gets the value of the elapsed time from the start to end of signal  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): u32\* u32\_a\_elapsedTime  | Return value: : en\_ICU\_error\_t ICU\_OK = 0  | ICU\_NOK = 1  |  en\_ICU\_error\_t ICU\_getElapsedTime (u32 \*u32\_a\_elapsedTime);  |  | Syntax : ICU\_task (void)  | Description : ICU task sent to interrupt callback function  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : None  |  void ICU\_task (void); |

#### 

#### Ultrasonic APIs

|  |
| --- |
| |  | Syntax : US\_init (void)  | Description : Initializes UltrasSonic  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): None  | Return value: : en\_US\_error\_t US\_OK = 0  | US\_NOK = 1  |  en\_US\_error\_t US\_init (void);  |  | Syntax : US\_getDistance (u16 \*u16\_a\_distance)  | Description : gets the distance between the object and the obstacle  | Sync\Async : Synchronous  | Reentrancy : Reentrant  | Parameters (in) : None  | Parameters (out): u16\* u16\_a\_distance  | Return value: : en\_US\_error\_t US\_OK = 0  | US\_NOK = 1  |  en\_US\_error\_t US\_getDistance (u16 \*u16\_a\_distance); |

#### Delay APIs

|  |
| --- |
| |  | **Syntax** : en\_DELAY\_error\_t DELAY\_init (void)  | **Description** : Initializes delay module  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_DELAY\_error\_t DELAY\_OK = 0  | DELAY\_NOK = 1  |  en\_DELAY\_error\_t DELAY\_init (void);  |  | **Syntax** : en\_DELAY\_error\_t DELAY\_setTime (f32 f32\_a\_timeInMS)  | **Description** : Sets blocking delay time  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : f32 f32\_a\_timeInMS  | **Parameters (out)**: None  | **Return value:** : en\_DELAY\_error\_t DELAY\_OK = 0  | DELAY\_NOK = 1  |  en\_DELAY\_error\_t DELAY\_setTime (f32 f32\_a\_timeInMS);  |  | **Syntax** : en\_DELAY\_error\_t DELAY\_setTime (f32 f32\_a\_timeInMS)  | **Description** : Sets non-blocking delay time  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : f32 f32\_a\_timeInMS  | **Parameters (out)**: None  | **Return value:** : en\_DELAY\_error\_t DELAY\_OK = 0  | DELAY\_NOK = 1  | en\_DELAY\_error\_t DELAY\_setTimeNonBlocking (f32 f32\_a\_timeInMs);  |  | **Syntax** : en\_DELAY\_error\_t DELAY\_setCallBack (void (\*funPtr)(void))  | **Description** : Sets call back for the function which gonna be performed  | when the nonblocking delay times out  | **Sync\Async**  : Synchronous  | **Reentrancy**  : Reentrant  | **Parameters (in)** : void (\*funPtr)(void)  | **Parameters (out)**: None  | **Return value:** : en\_DELAY\_error\_t DELAY\_OK = 0  | DELAY\_NOK = 1  |  en\_DELAY\_error\_t DELAY\_setCallBack (void (\*funPtr)(void)); |

#### PWM APIs

|  |
| --- |
| |  | **Syntax**  : en\_PWM\_error\_t PWM\_init (void)  | **Description**  : Initializes PWM module  | **Sync\Async**  : Synchronous  | **Reentrancy**  : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  en\_PWM\_error\_t PWM\_init (void);  |  | **Syntax** : en\_PWM\_error\_t PWM\_setDutyCycle (u8 u8\_a\_speed)  | **Description** : Sets PWM duty cycle  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : u8 u8\_a\_speed  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  en\_PWM\_error\_t PWM\_setDutyCycle (u8 u8\_a\_speed);  |  | **Syntax** : en\_PWM\_error\_t PWM\_start (void)  | **Description** : Starts PWM  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  en\_PWM\_error\_t PWM\_start (void);  |  | **Syntax** : en\_PWM\_error\_t PWM\_stop (void)  | **Description** : Stops PWM  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  en\_PWM\_error\_t PWM\_stop (void);  |  | **Syntax** : en\_PWM\_error\_t PWM\_onTask (void)  | **Description** : task called by the interrupt when the signal is in HIGH phase  | **Sync\Async** : Synchronous  | **Reentrancy** : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  void PWM\_onTask (void);  |  | **Syntax**  : en\_PWM\_error\_t PWM\_onTask (void)  | **Description**  : task called by the interrupt when the signal is in LOW phase  | **Sync\Async**  : Synchronous  | **Reentrancy**  : Reentrant  | **Parameters (in)** : None  | **Parameters (out)**: None  | **Return value:** : en\_PWM\_error\_t PWM\_OK = 0  | PWM\_NOK = 1  |  void PWM\_offTask (void); |

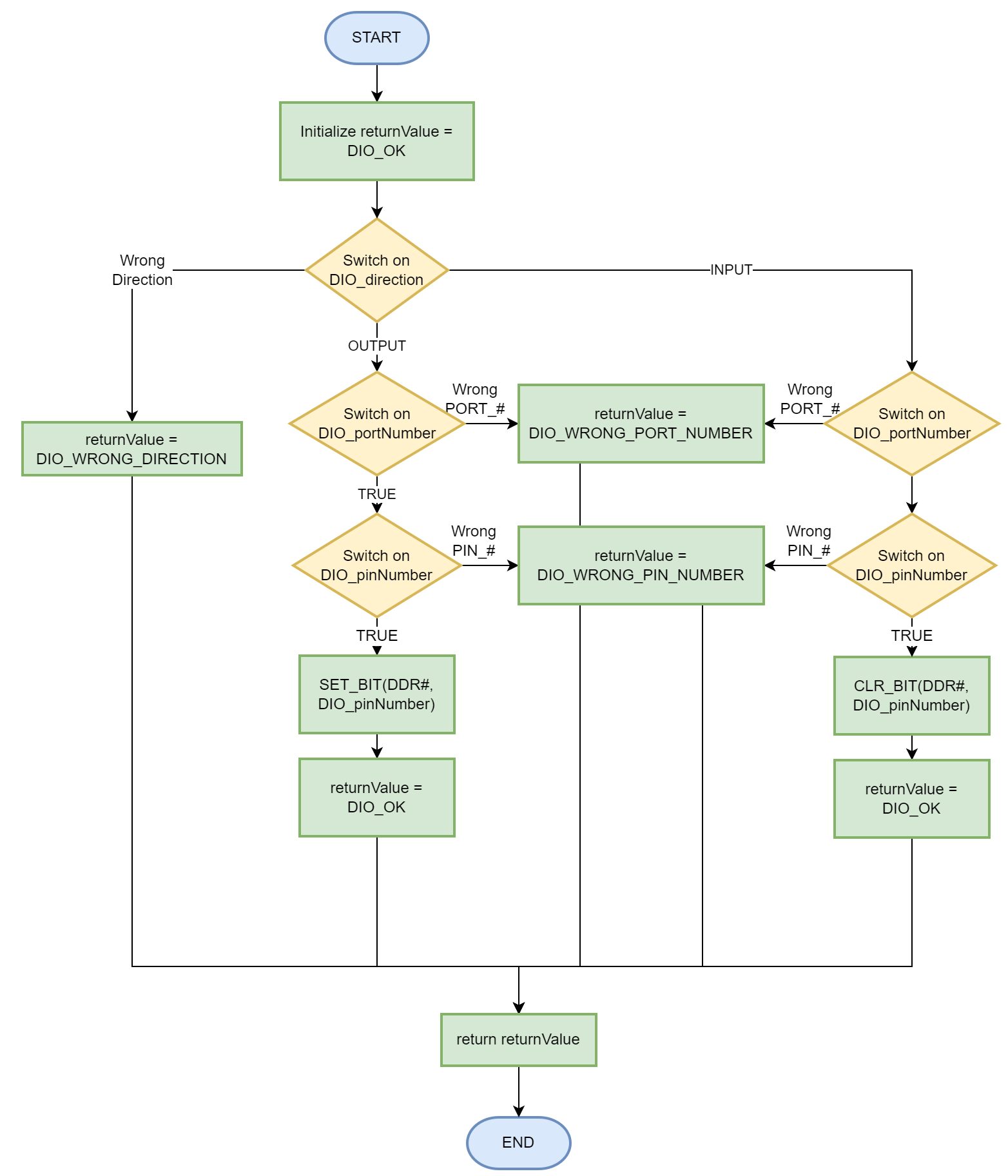
# 

# Low Level Design

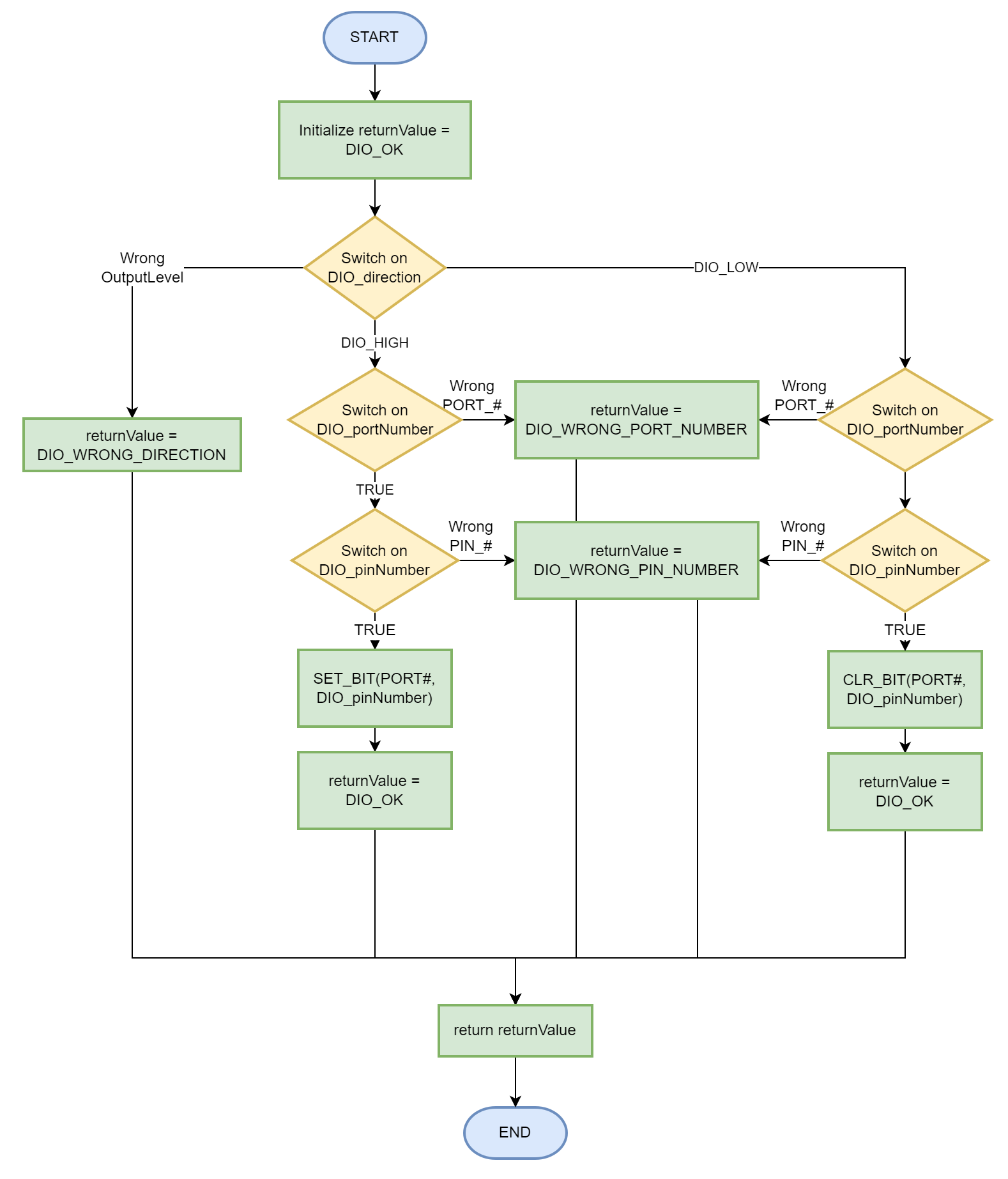
## MCAL Layer

### DIO Module

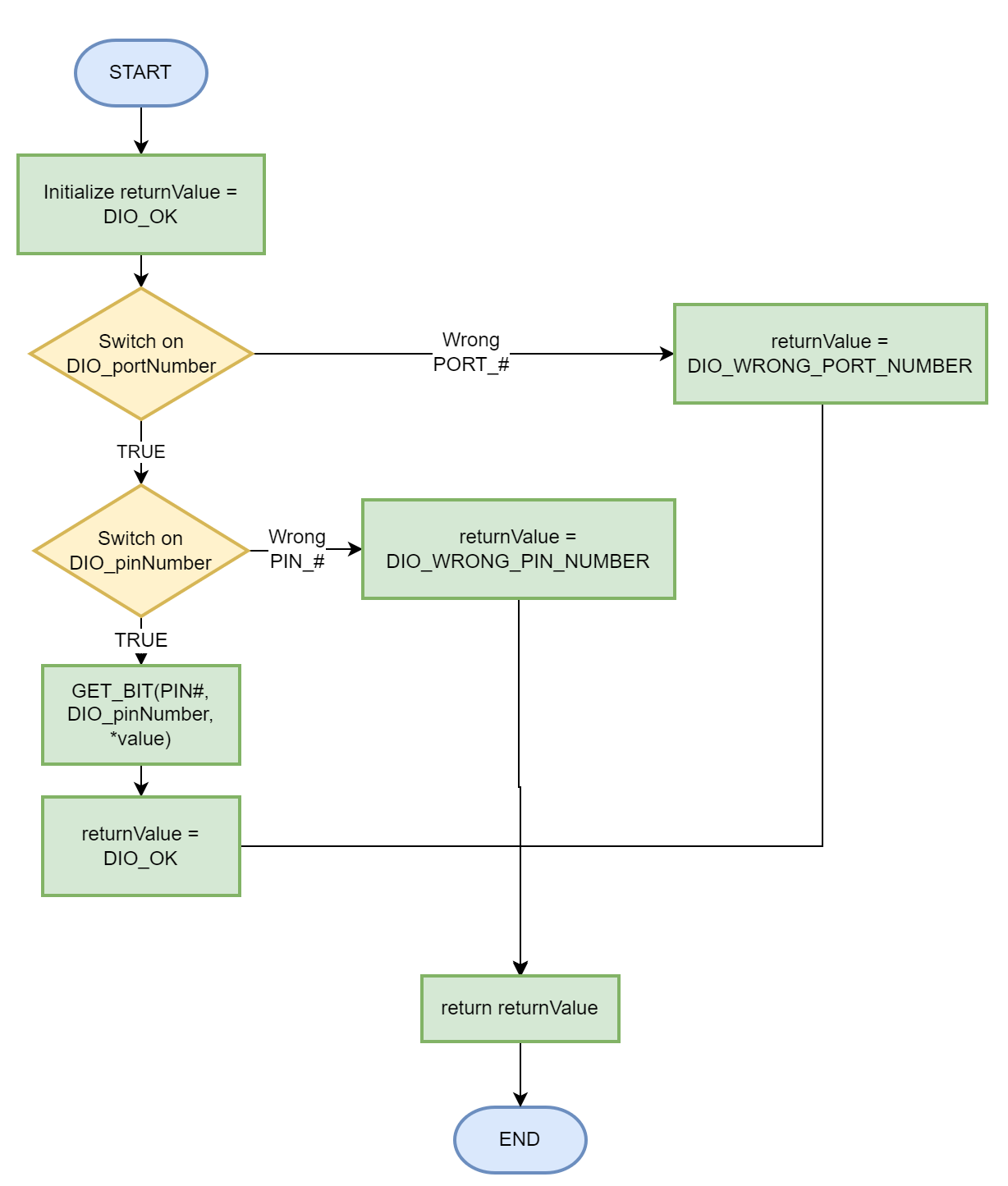
#### DIO\_init



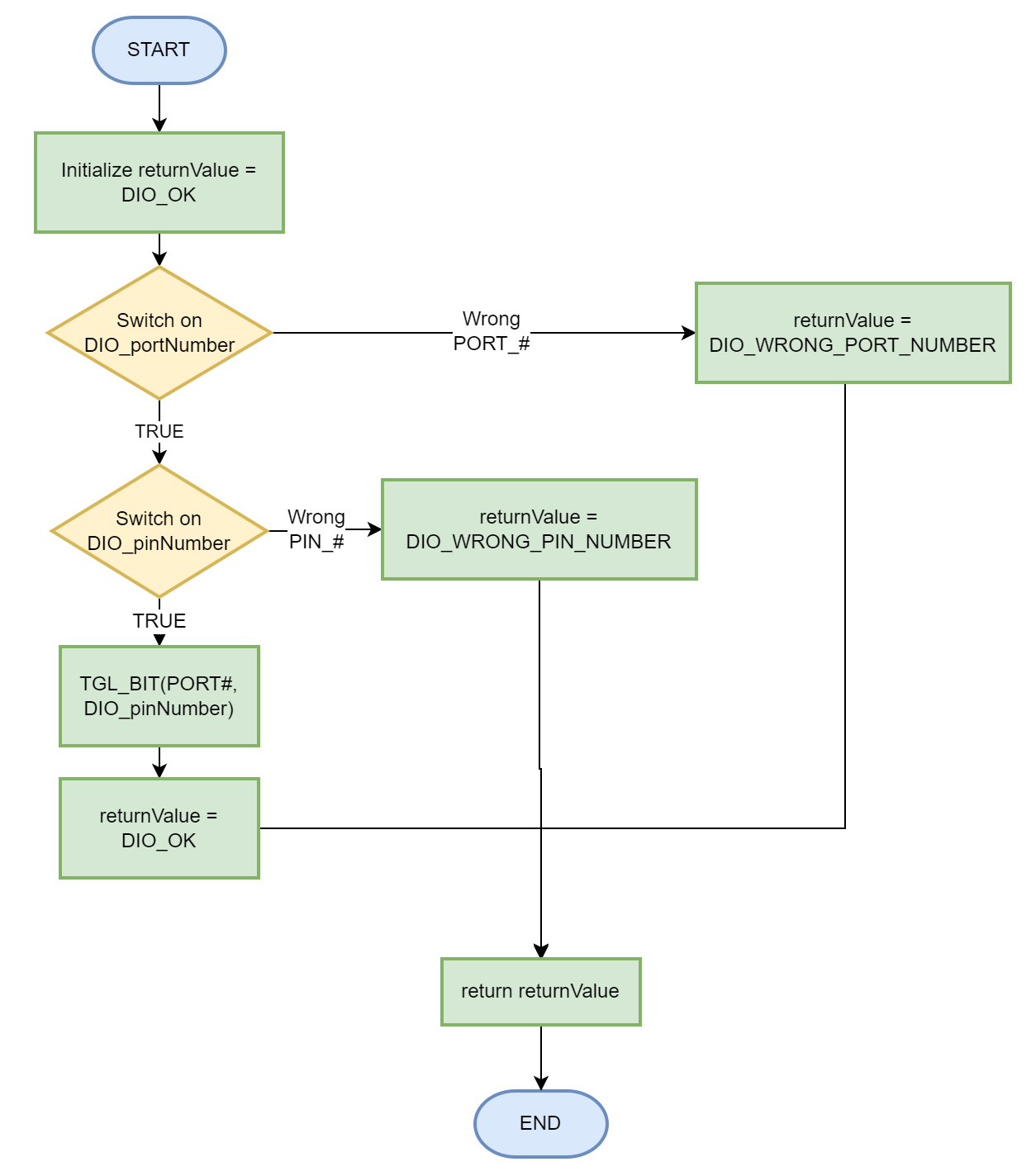
#### DIO\_write



#### DIO\_read



#### DIO\_toggle



#### DIO\_pinPullUp

#### 

### EXTINT Module

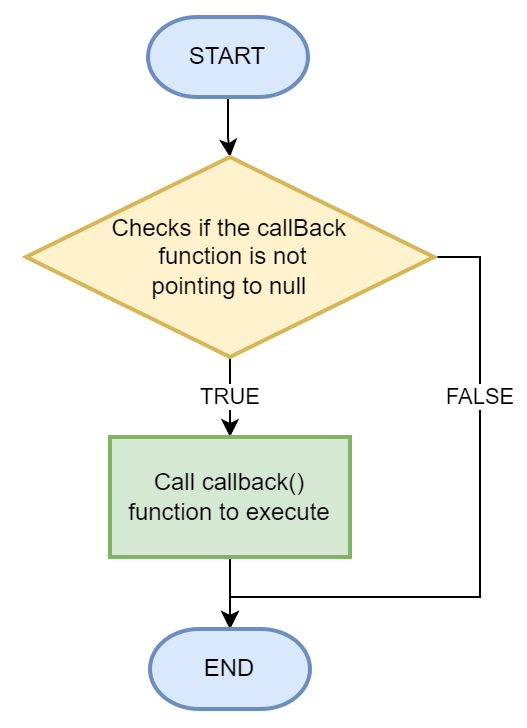
#### EXTINT\_init

#### 

#### EXTINT\_setCallBack

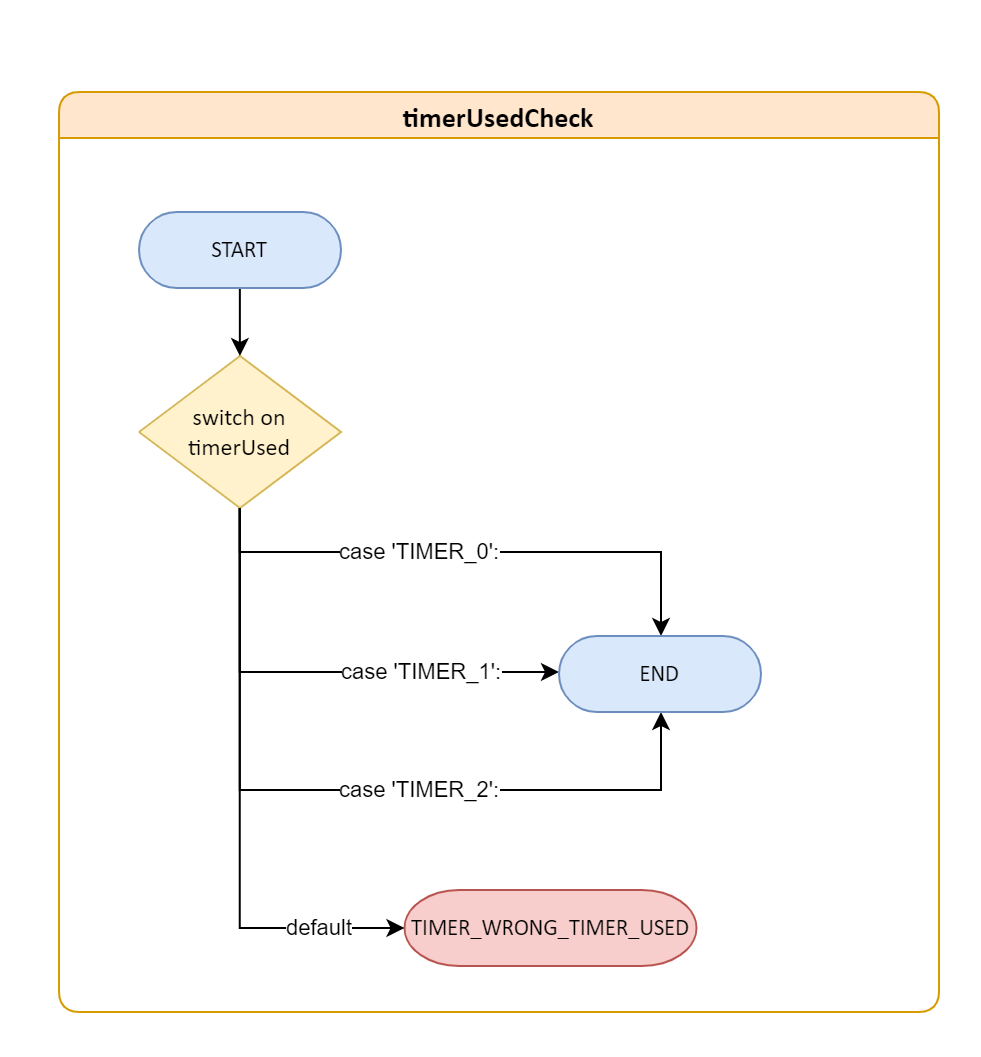
#### 

#### ISR(EXT\_INT\_#)

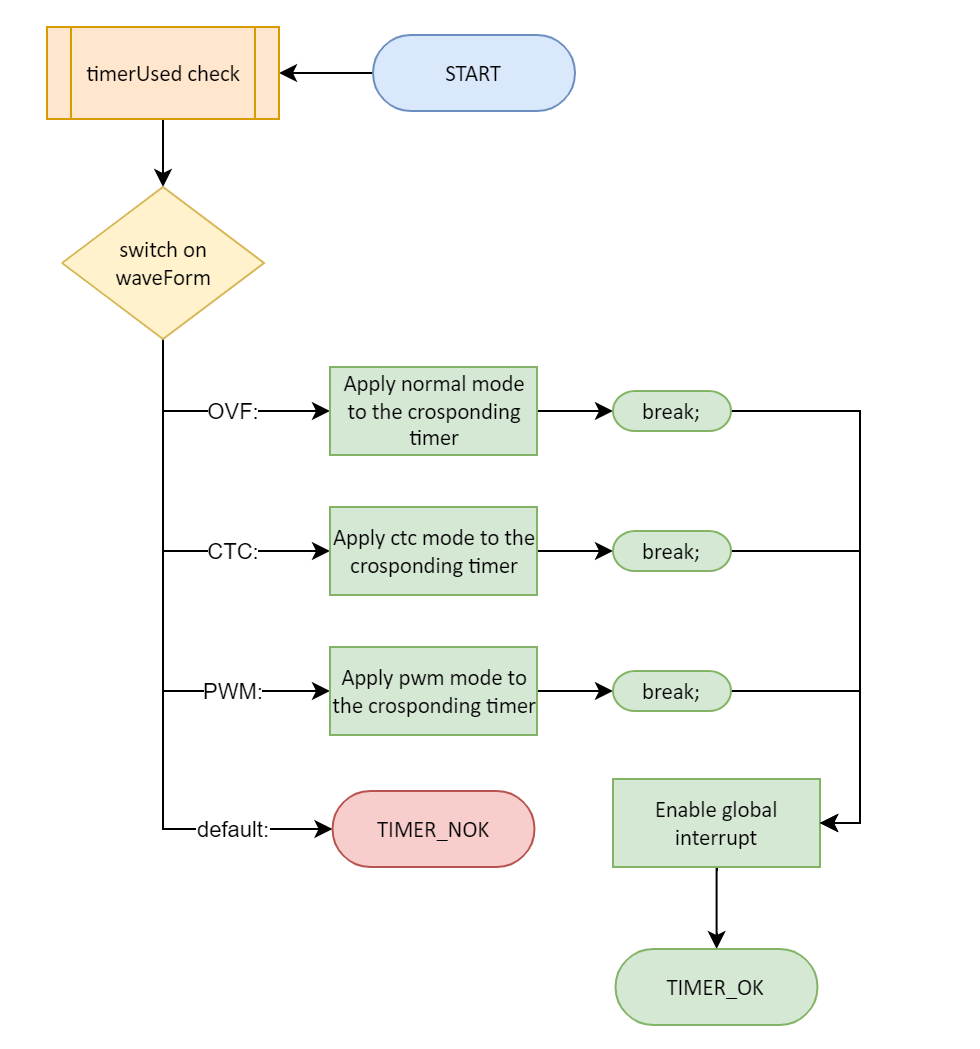


### Timer Module

#### sub process

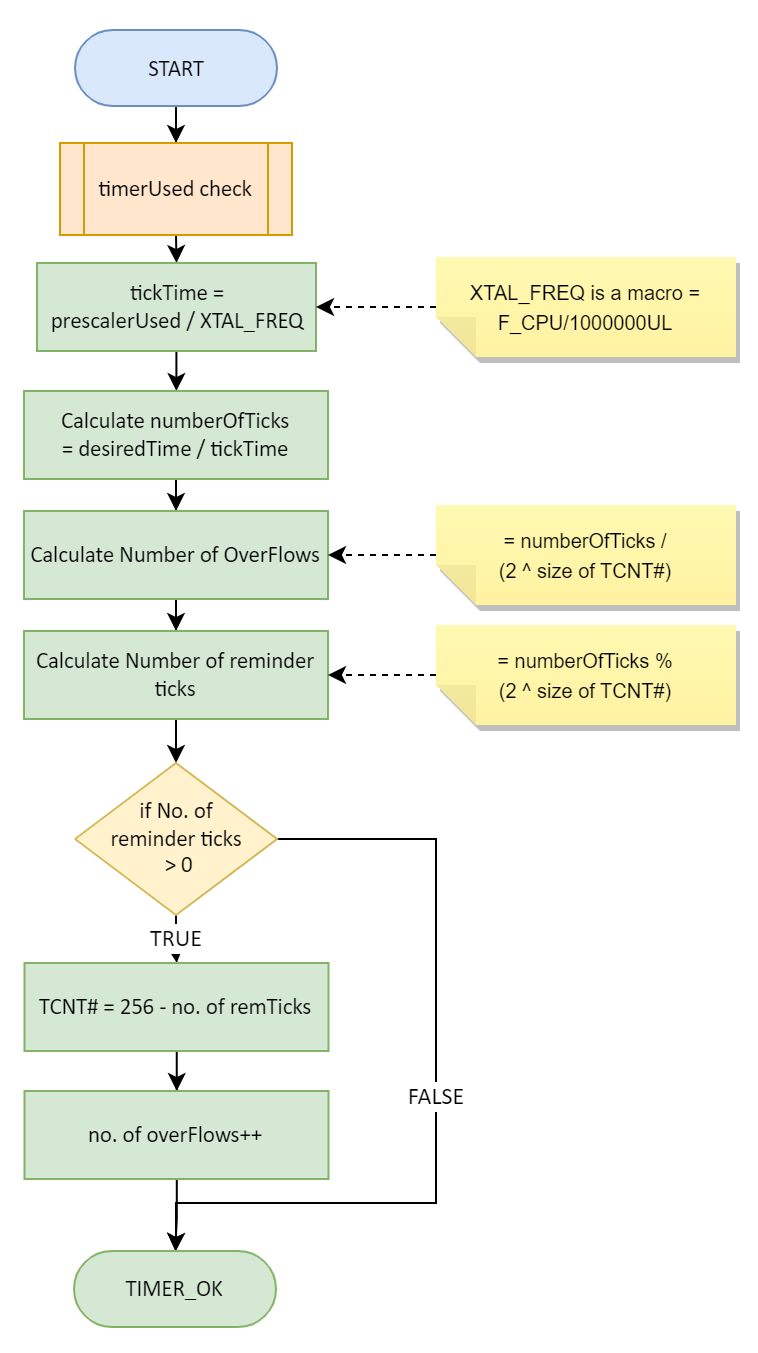


#### TIMER\_init

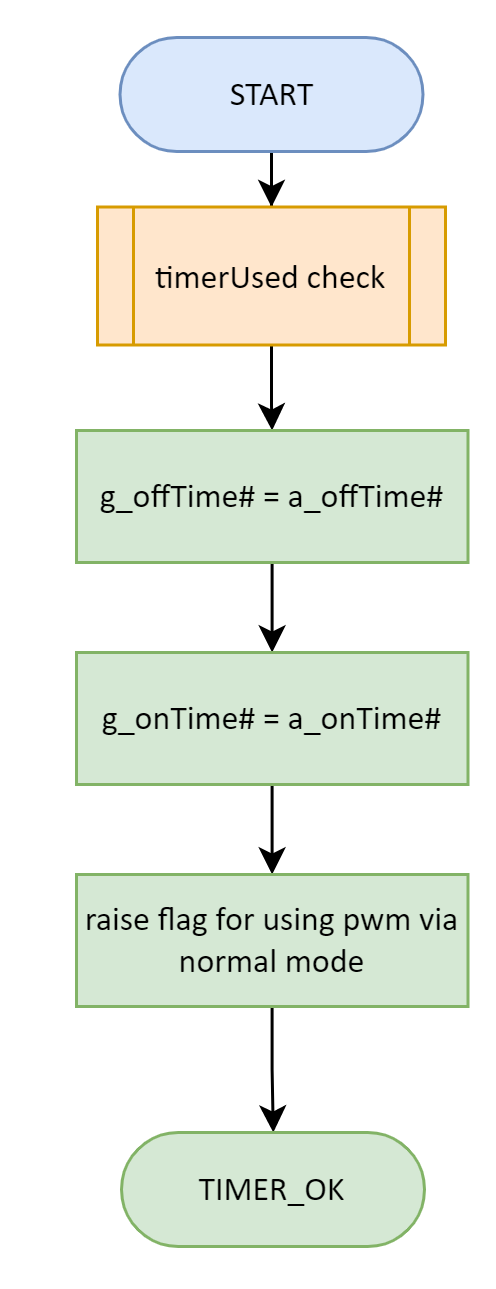


#### 

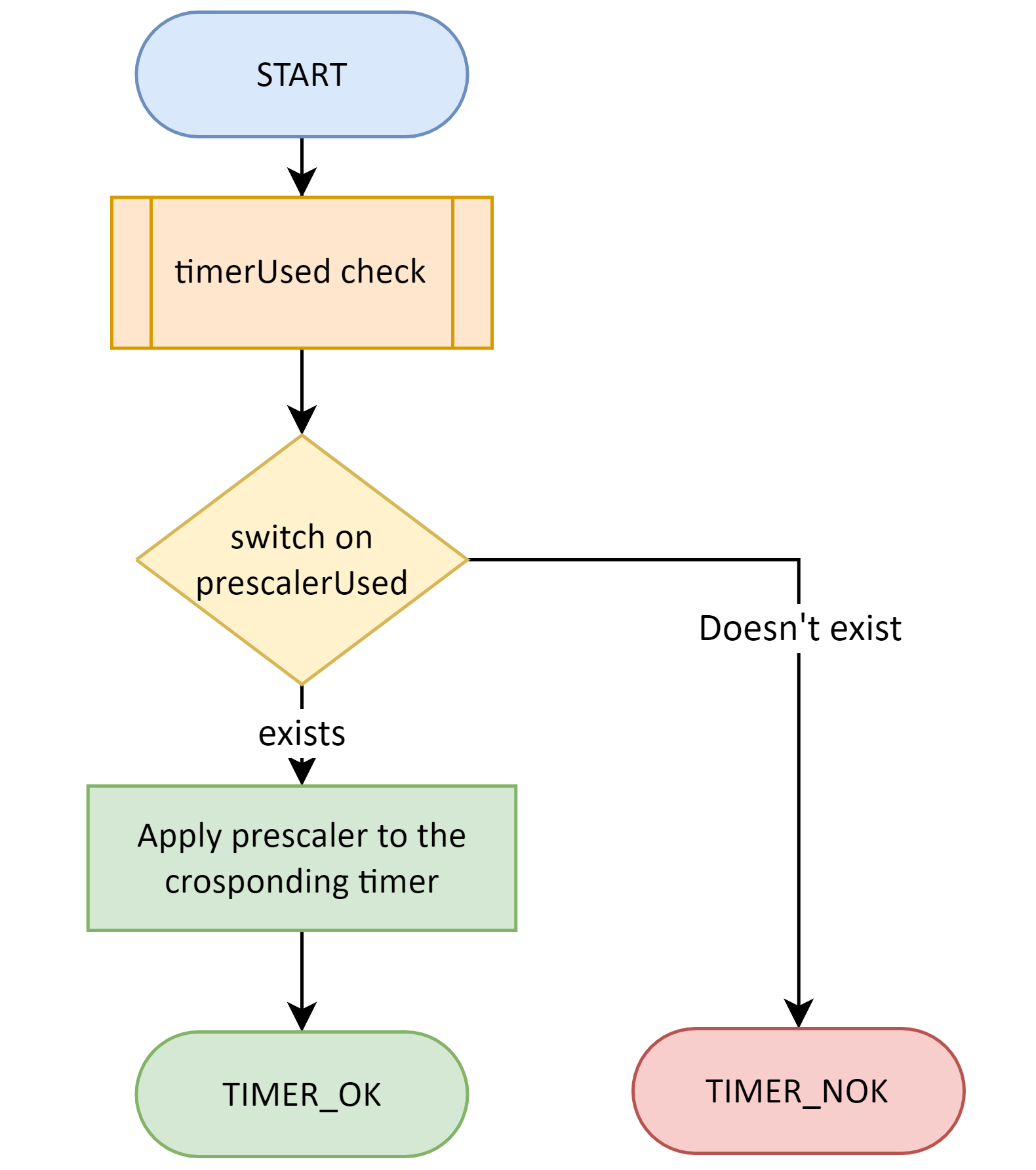
#### TIMER\_setTime



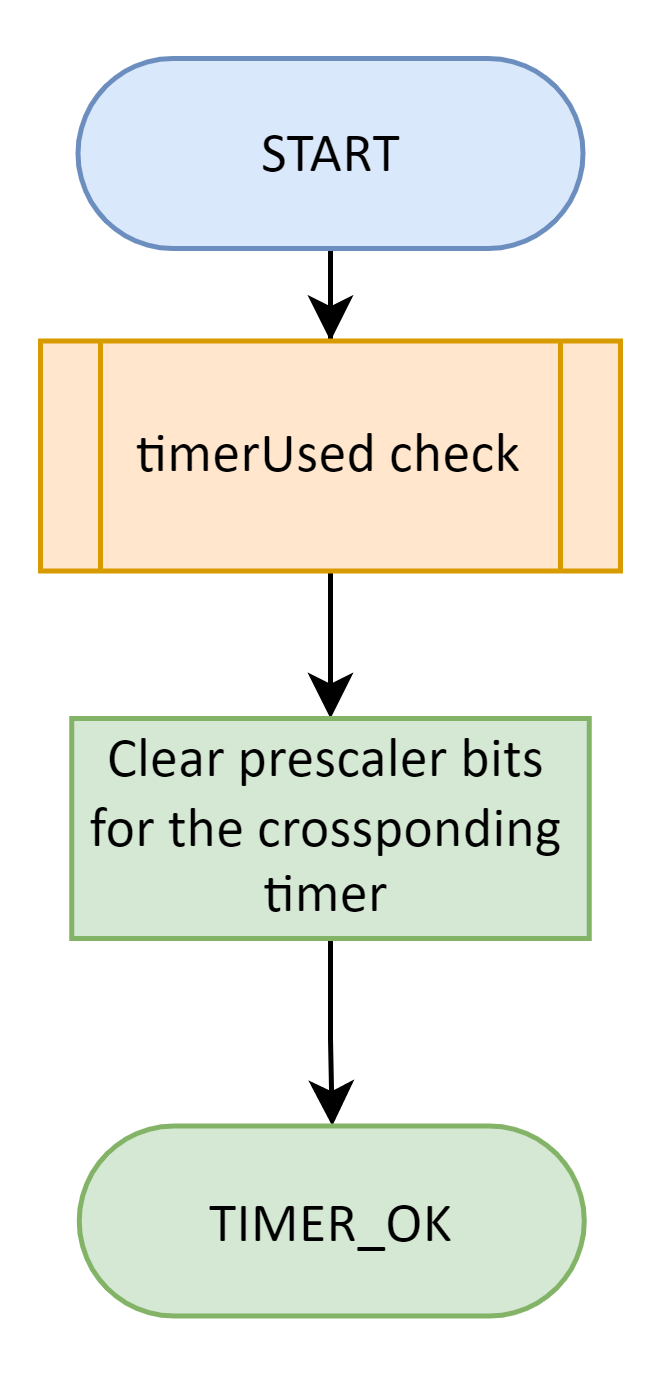
#### TIMER\_pwmGenerator



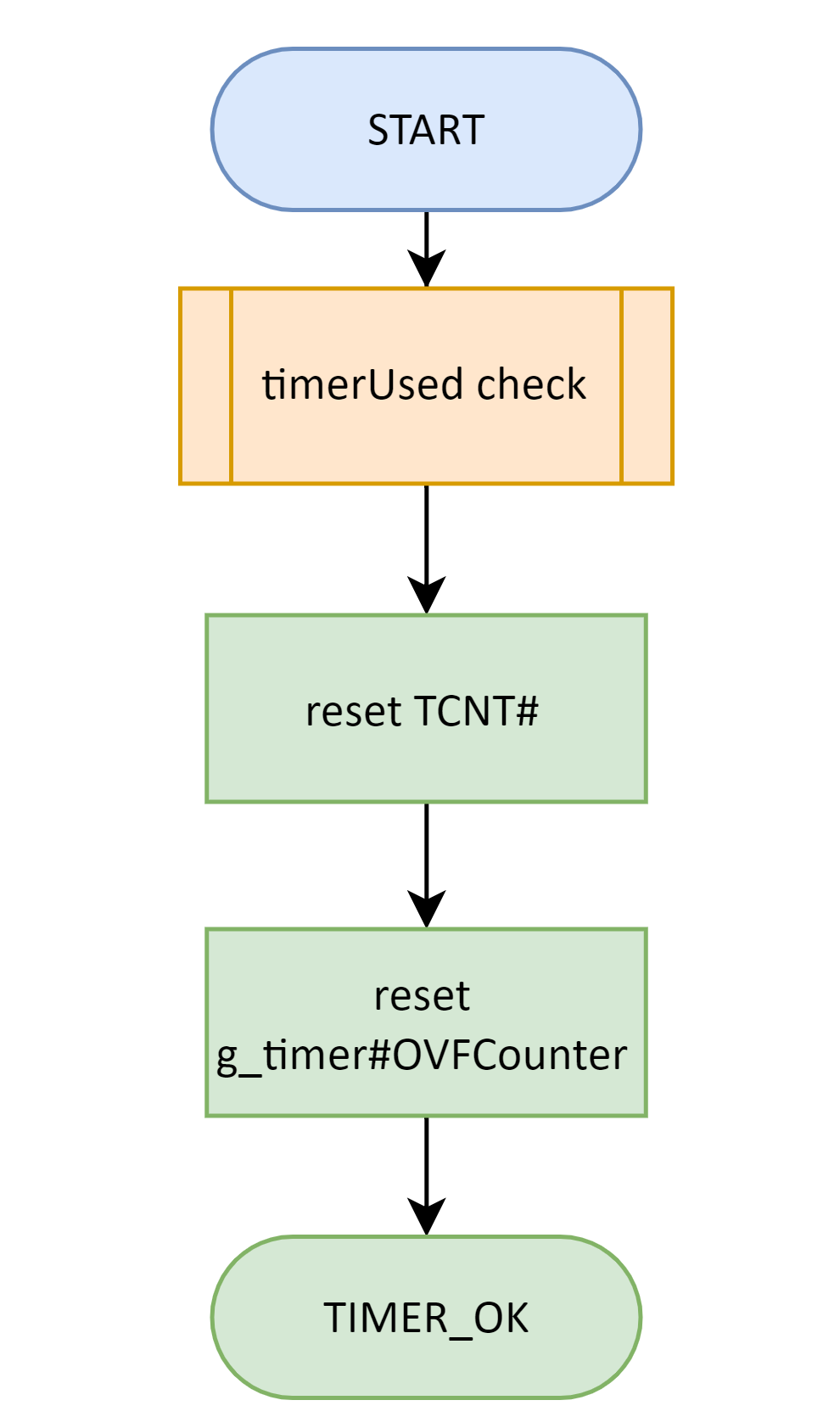
#### TIMER\_resume



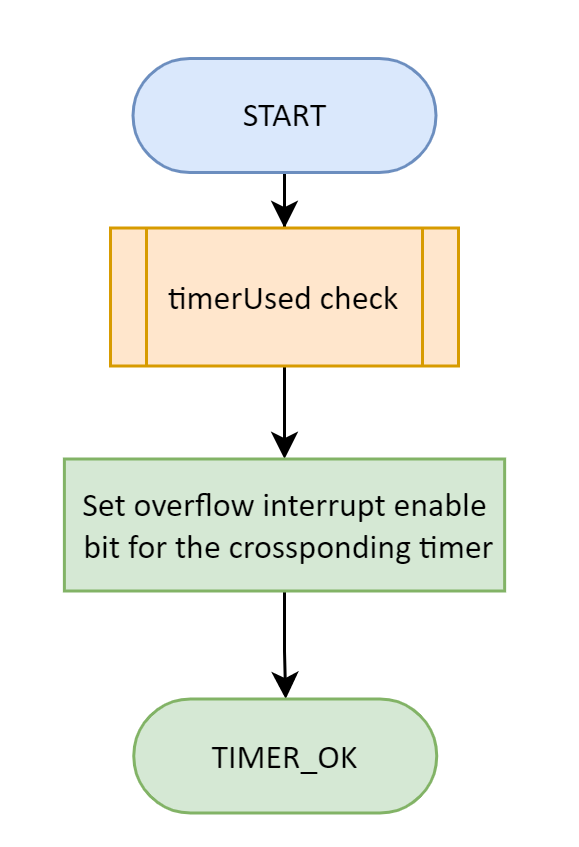
#### TIMER\_pause



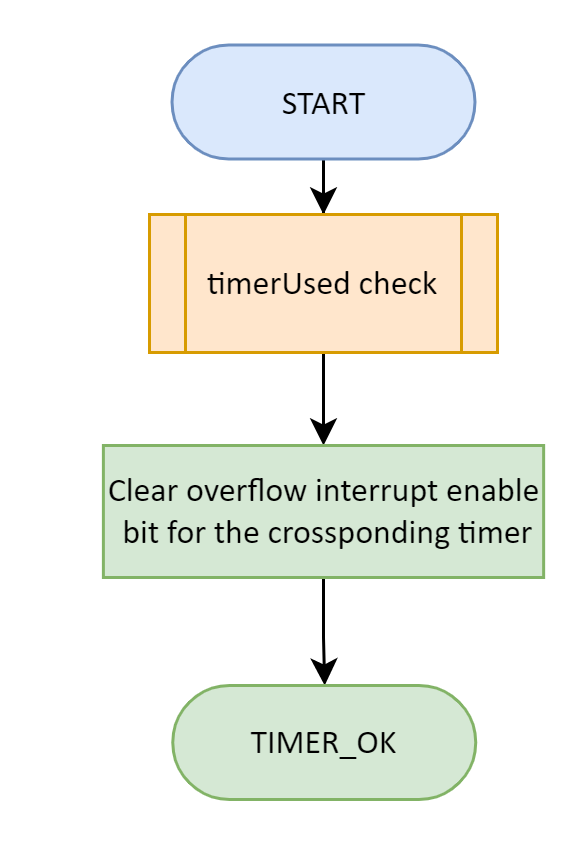
#### TIMER\_reset



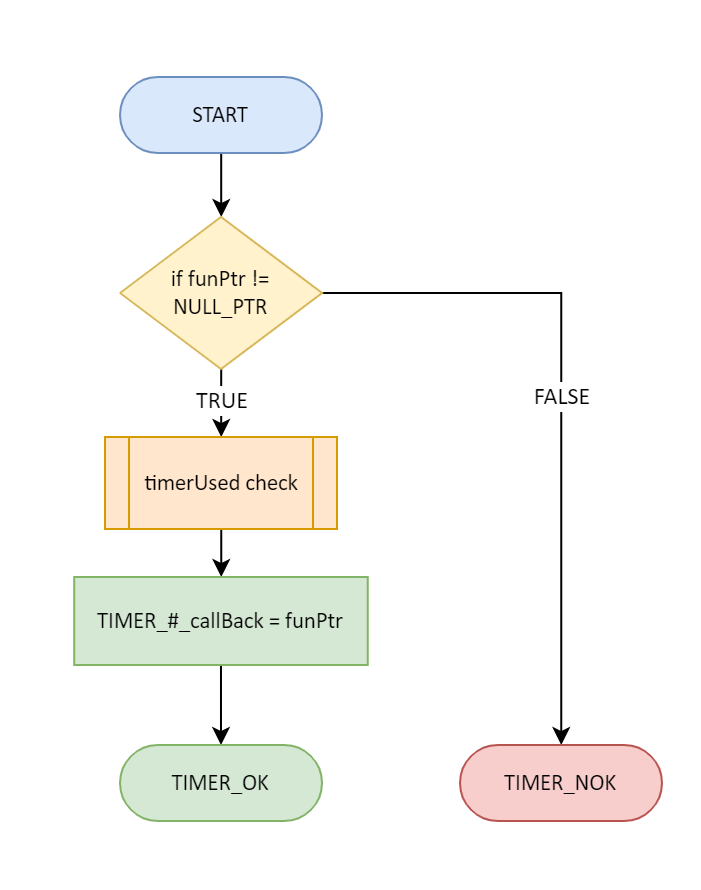
#### TIMER\_enableInterrupt



#### TIMER\_enableInterrupt



#### TIMER\_setCallBack



#### TIMER\_setPwmOnCallBack

#### 

#### TIMER\_setPwmOffCallBack

#### 

#### TIMER\_getElapsedTime

#### 

#### TIMER\_setDelayTime

#### 

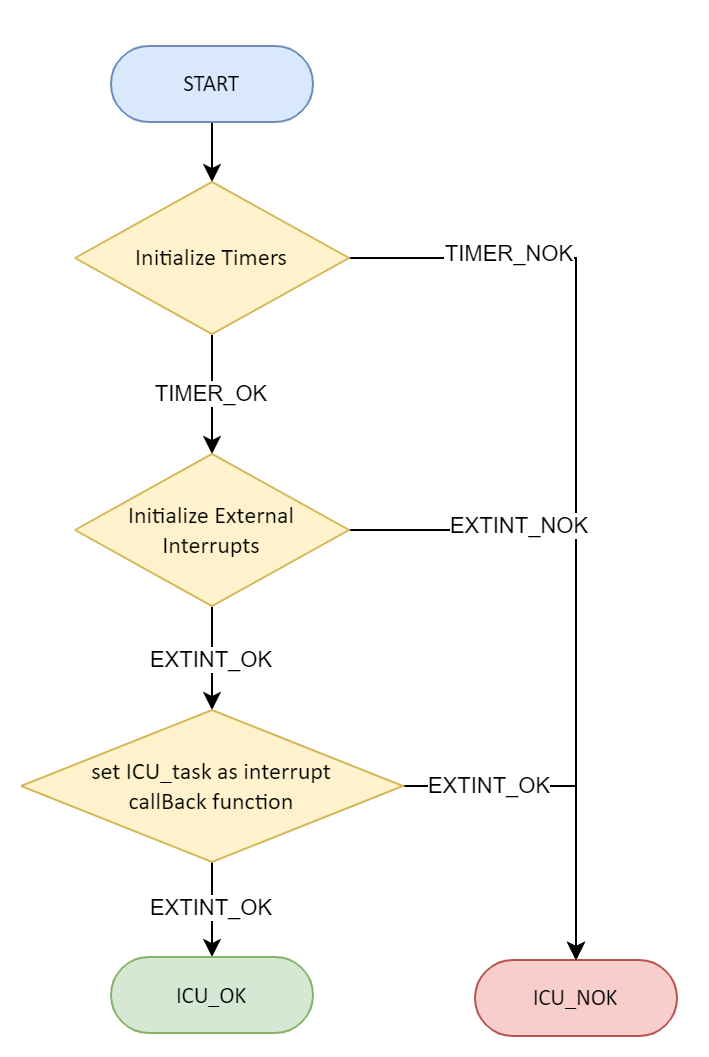
## 

## HAL Layer

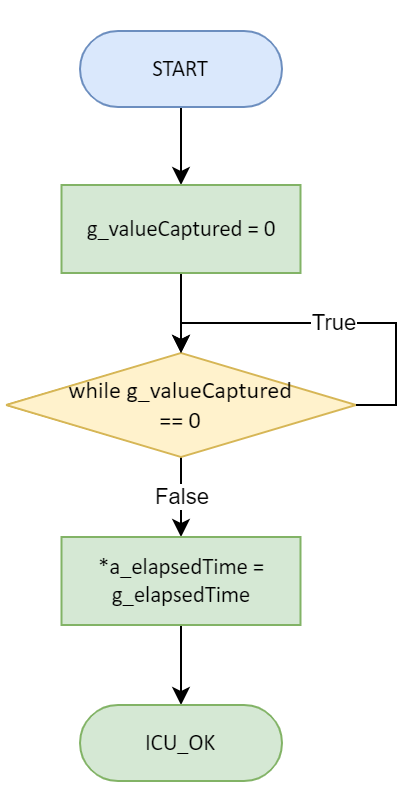
### ICU Module (Input Capture Unit)

Any variable has g\_ before it’s naming is a volatile global variable

#### ICU\_init



#### ICU\_getElapsedTime



#### ICU\_task

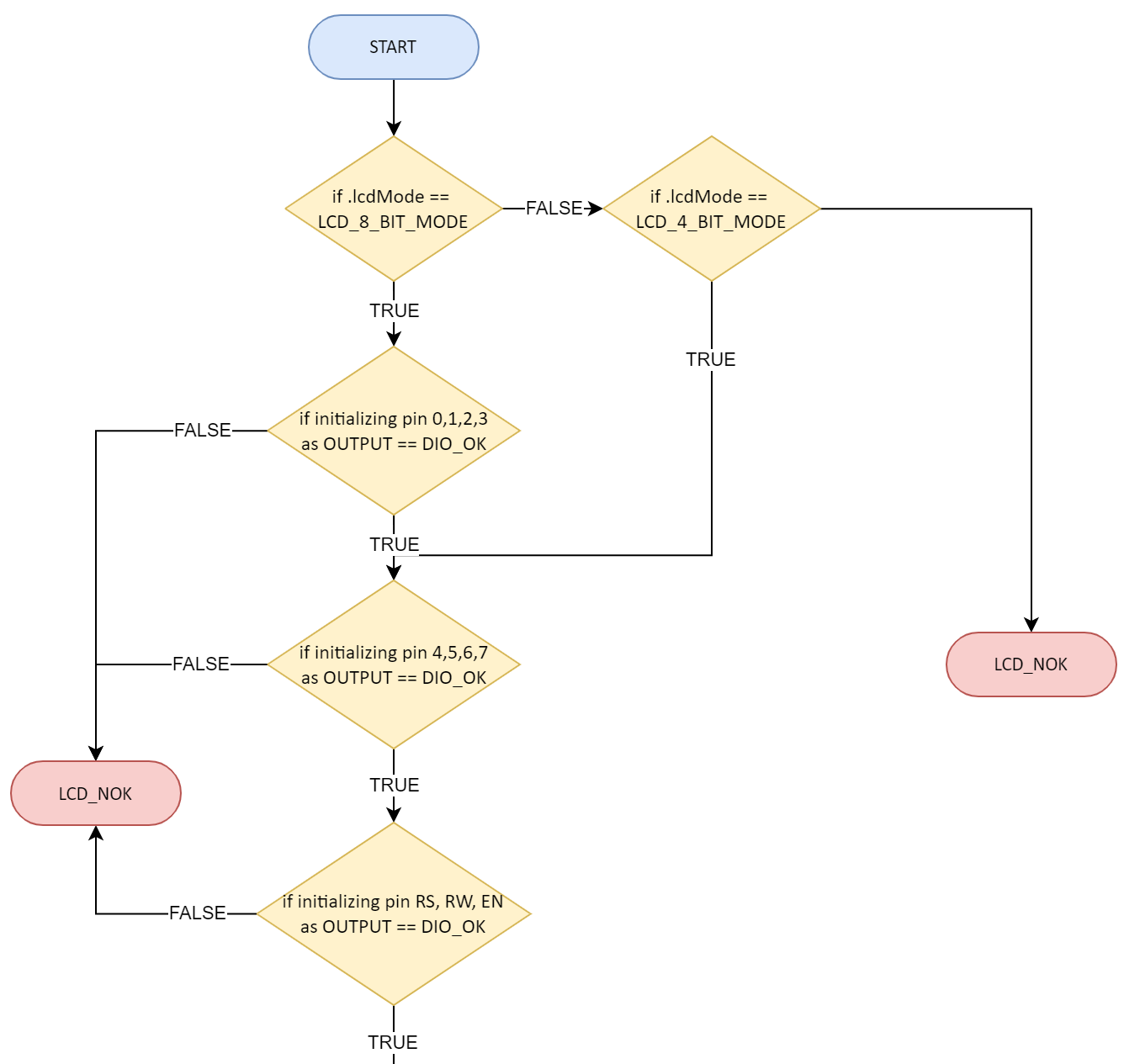
#### 

#### 

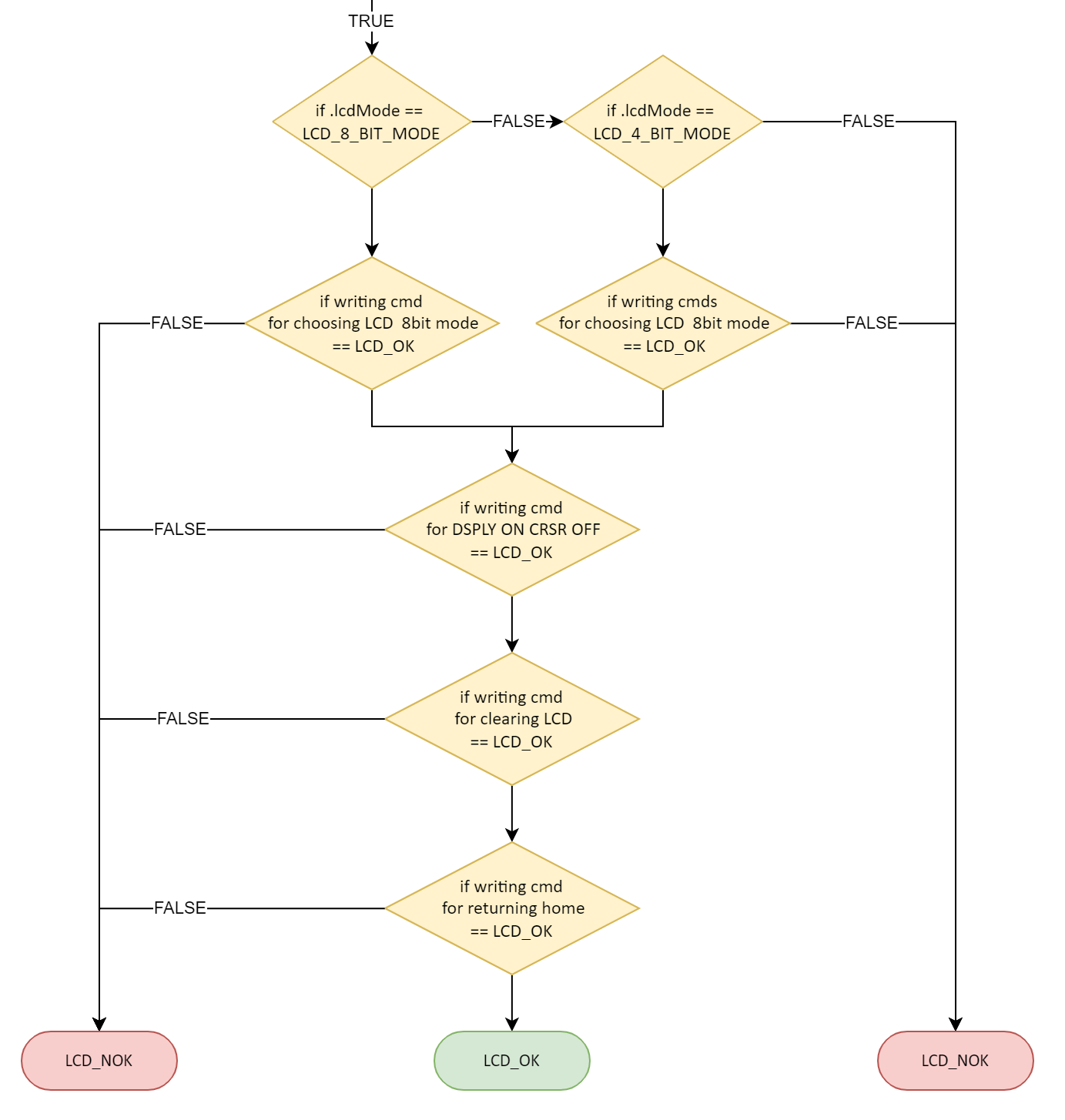
### LCD Module

#### LCD\_init

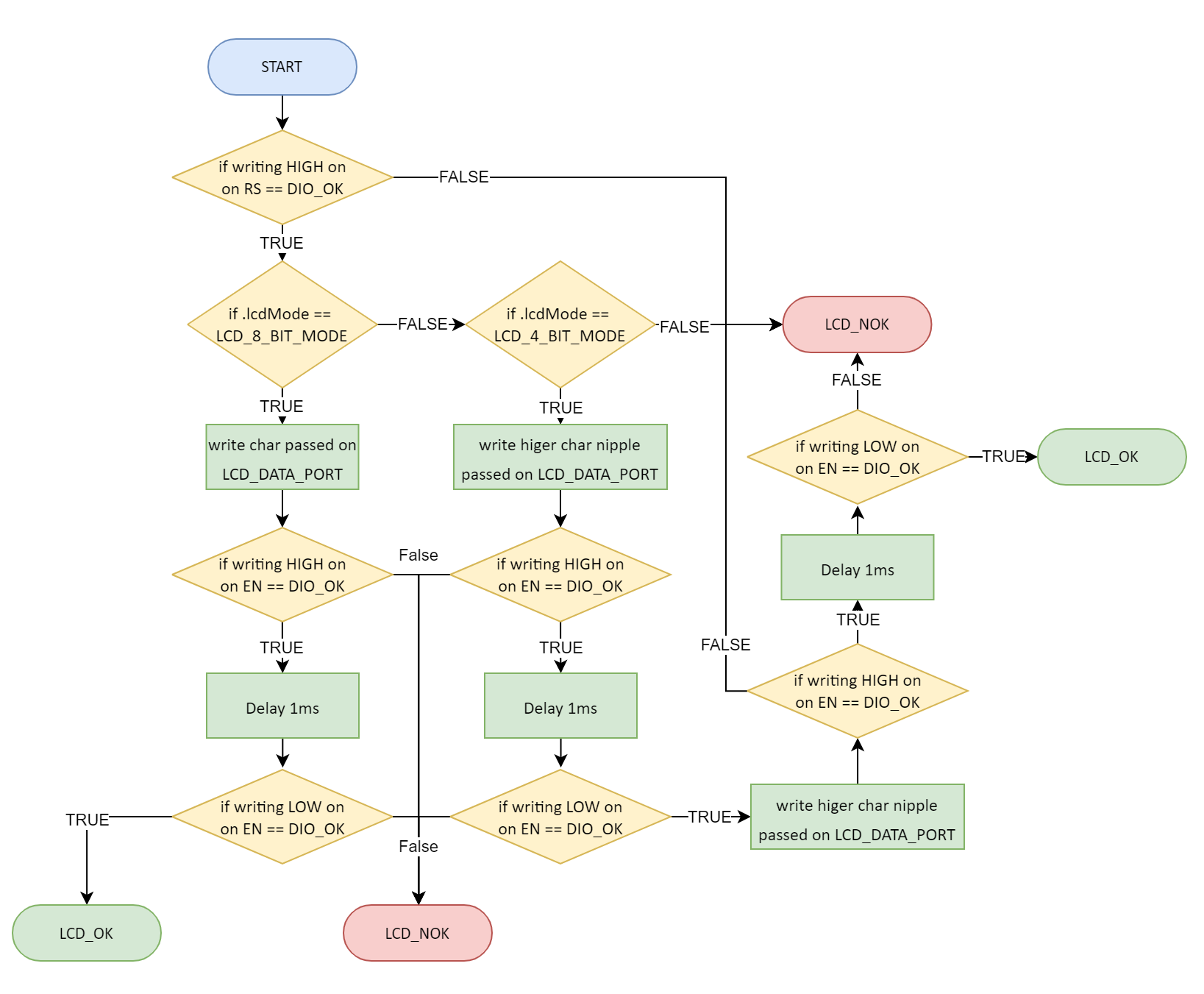
**Part 1**



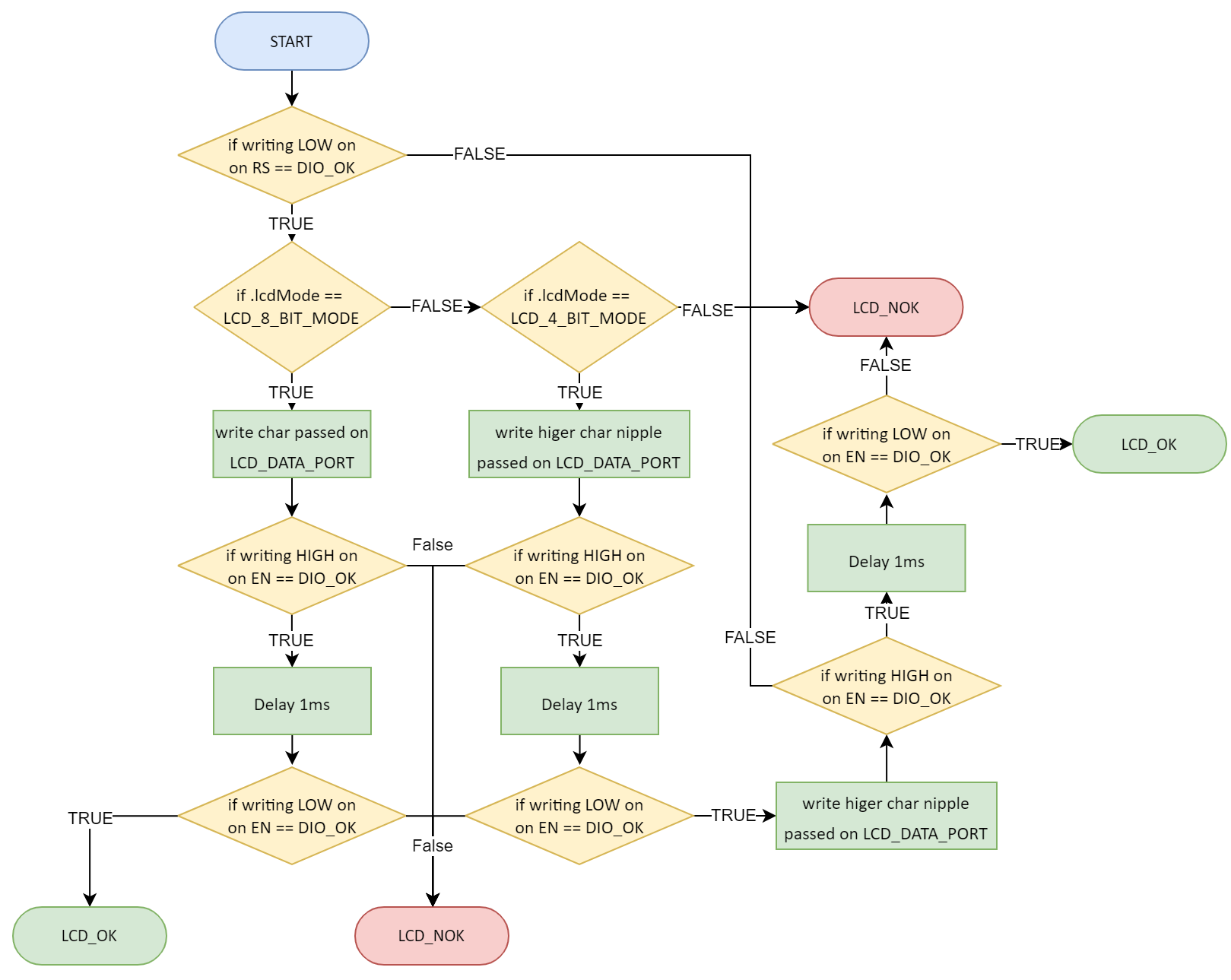
**Part 2**

****

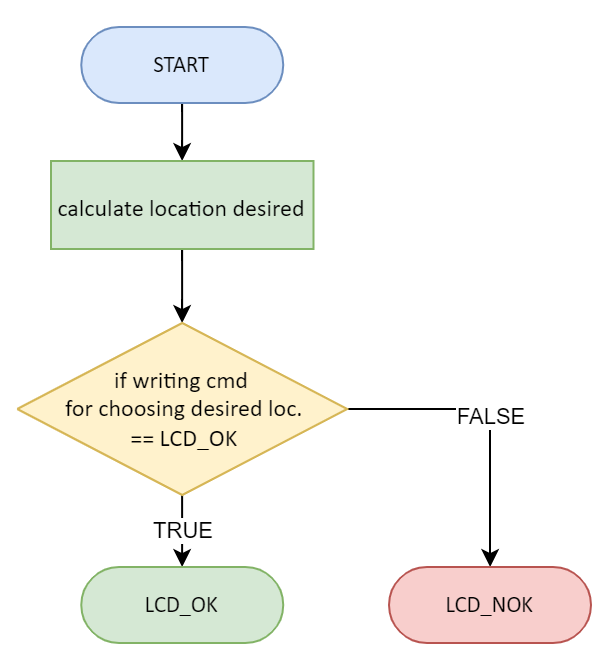
#### LCD\_writeChar



#### LCD\_writeCmd



#### LCD\_clr



#### LCD\_goTo

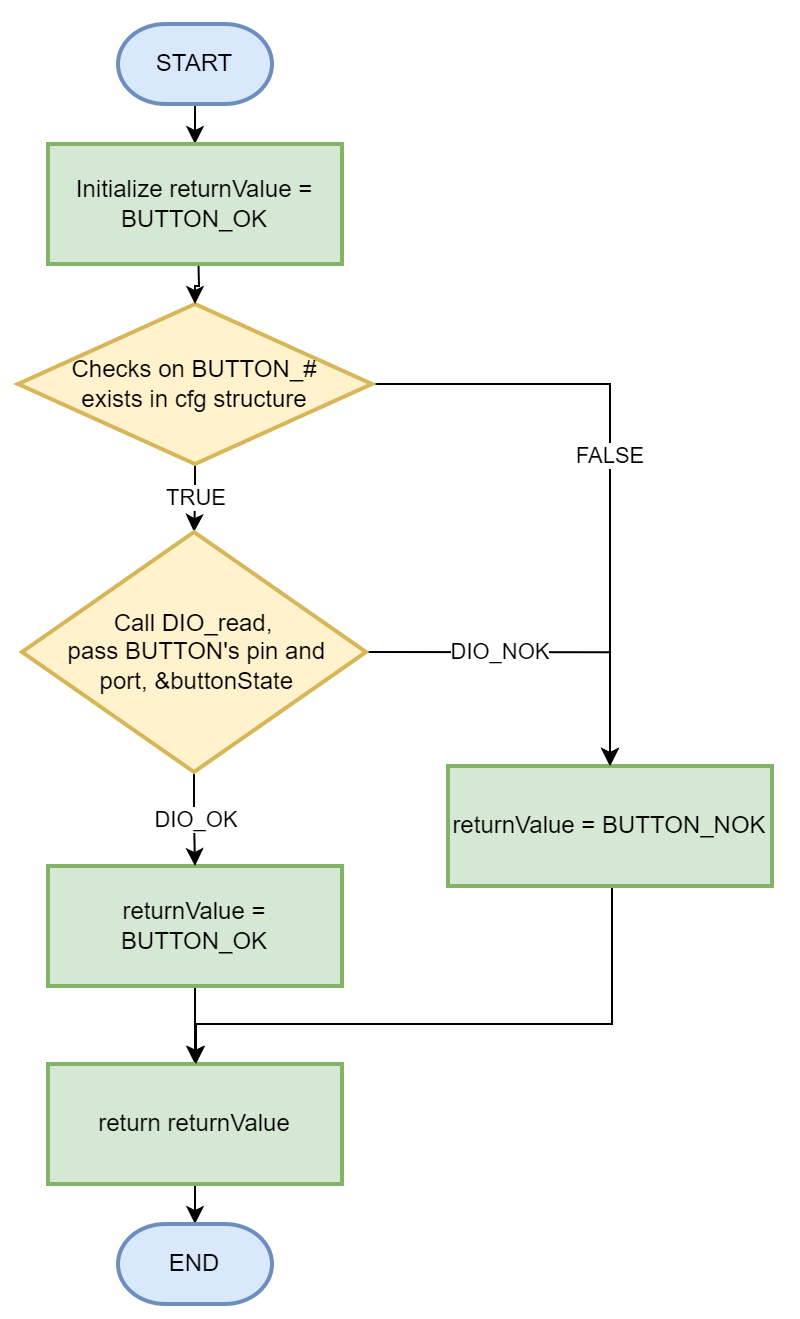
#### 

### BTN Module

#### BUTTON\_init

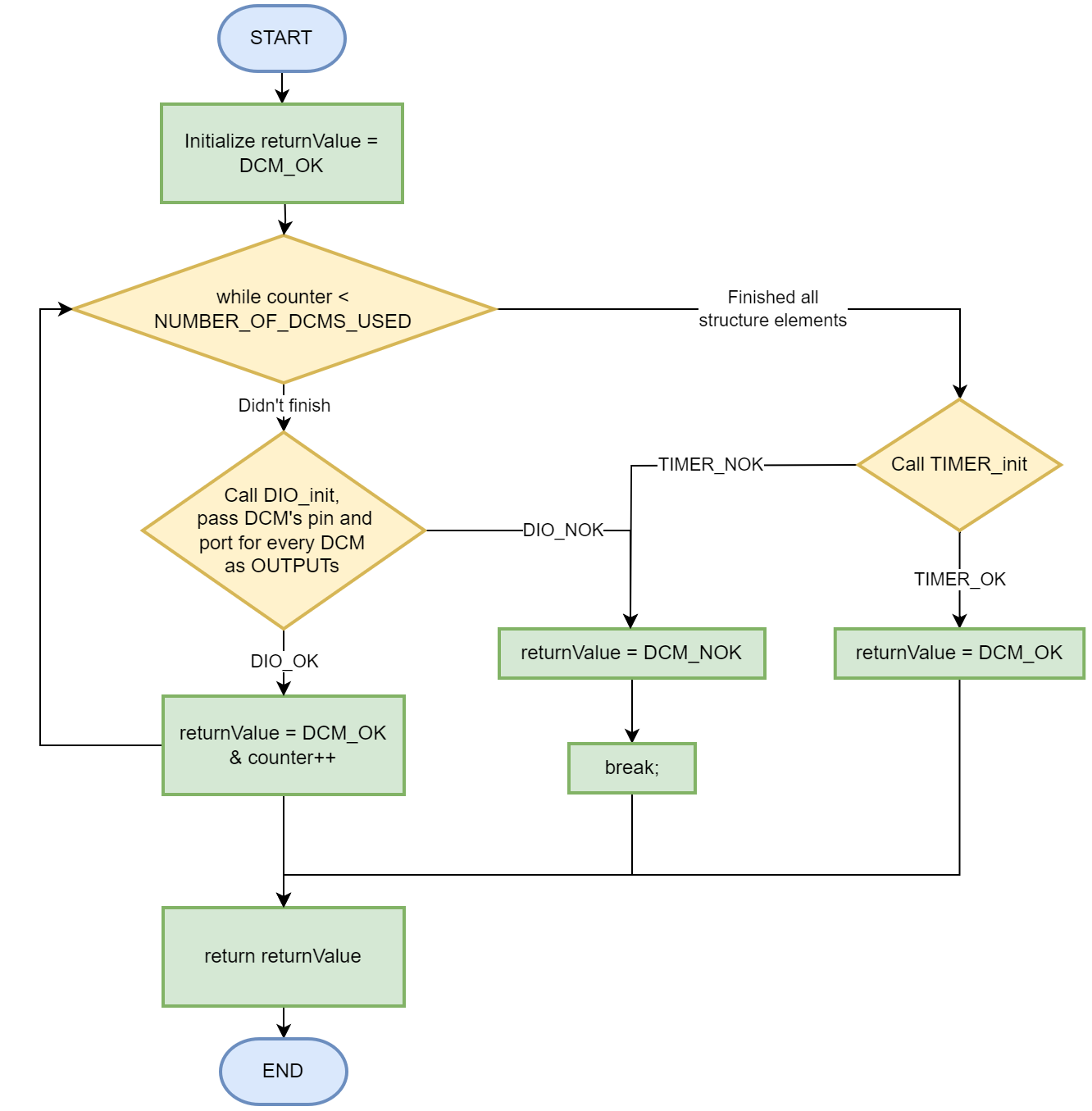
#### 

#### BUTTON\_read



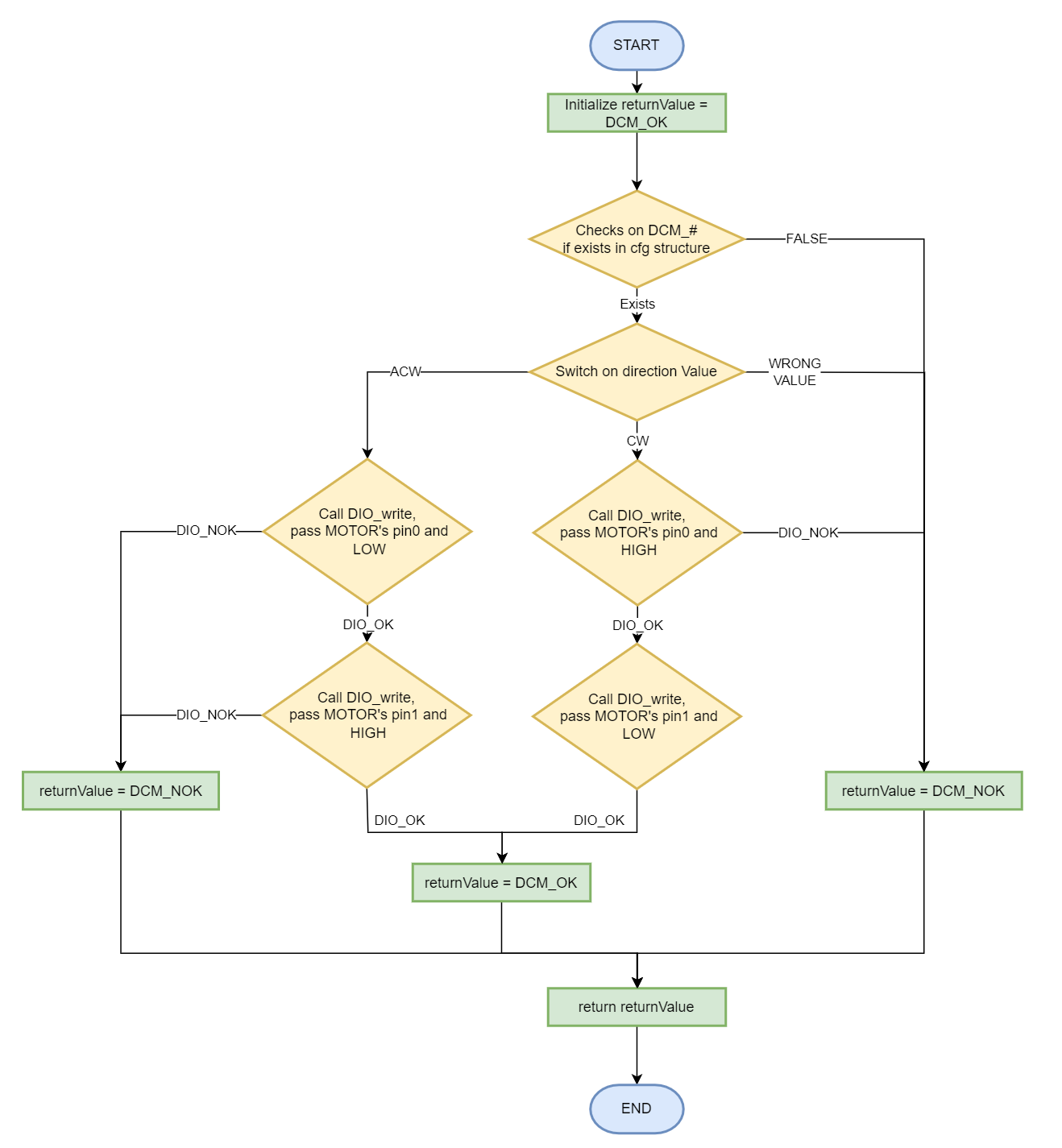
### DCM Module

#### DCM\_init

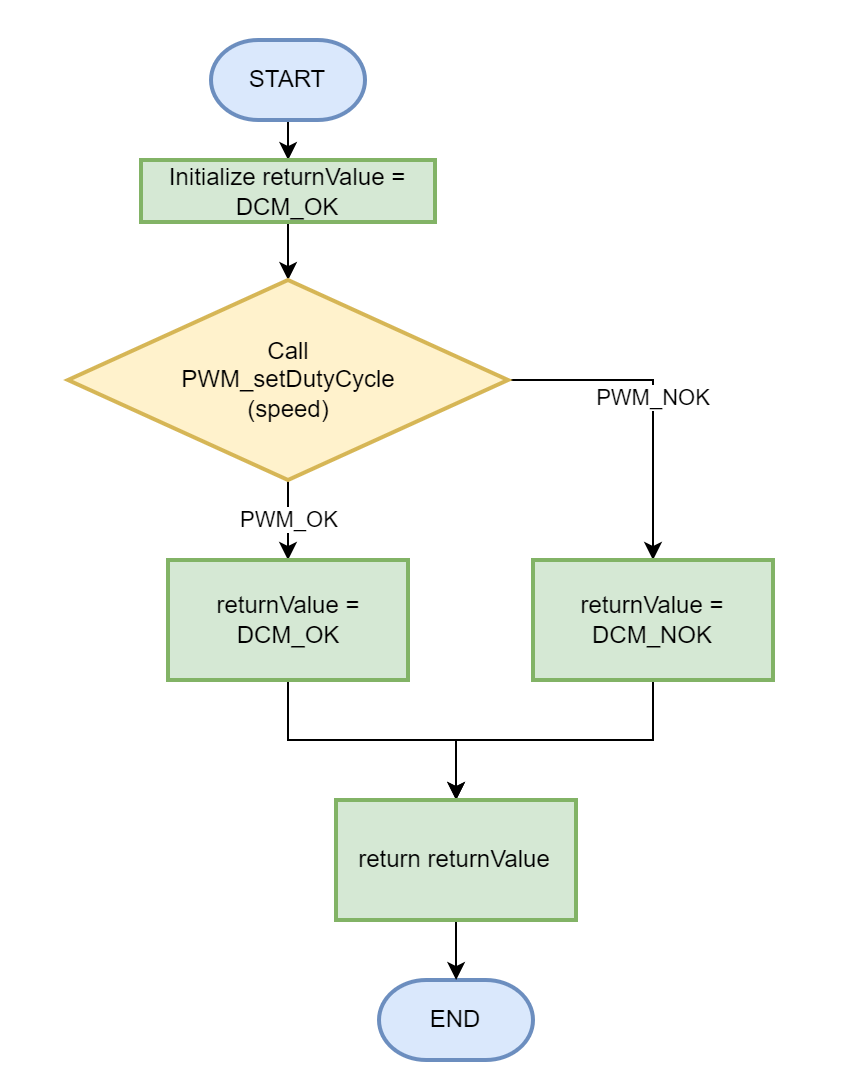


#### 

#### DCM\_setDirection

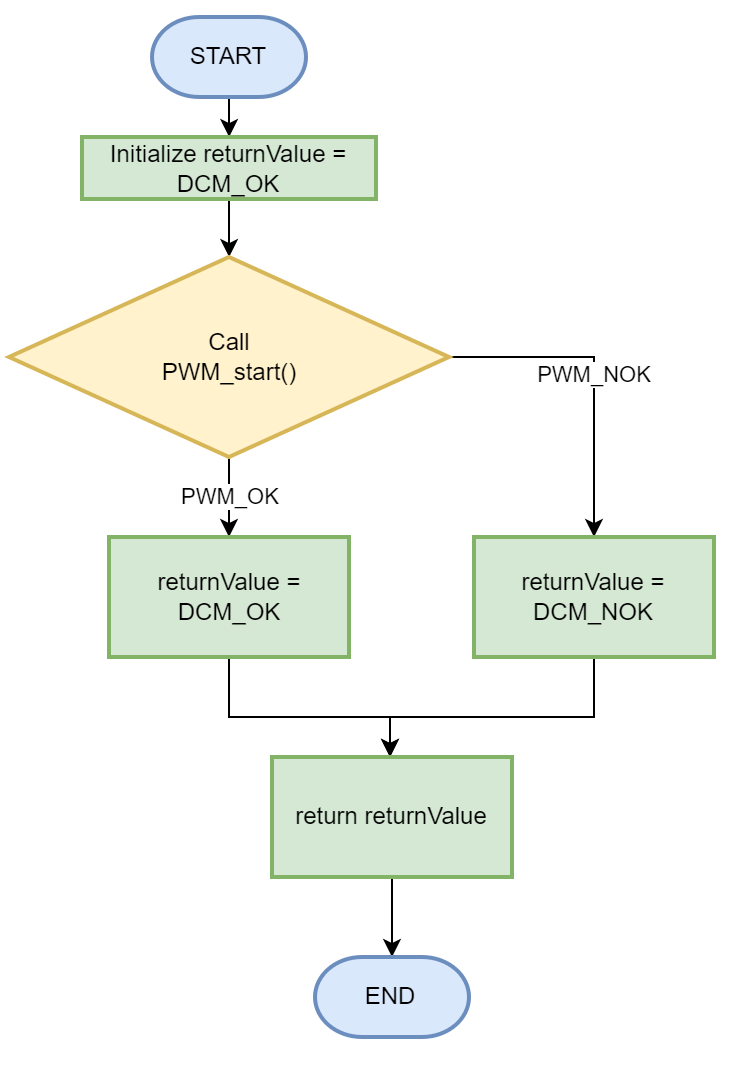


#### DCM\_speed

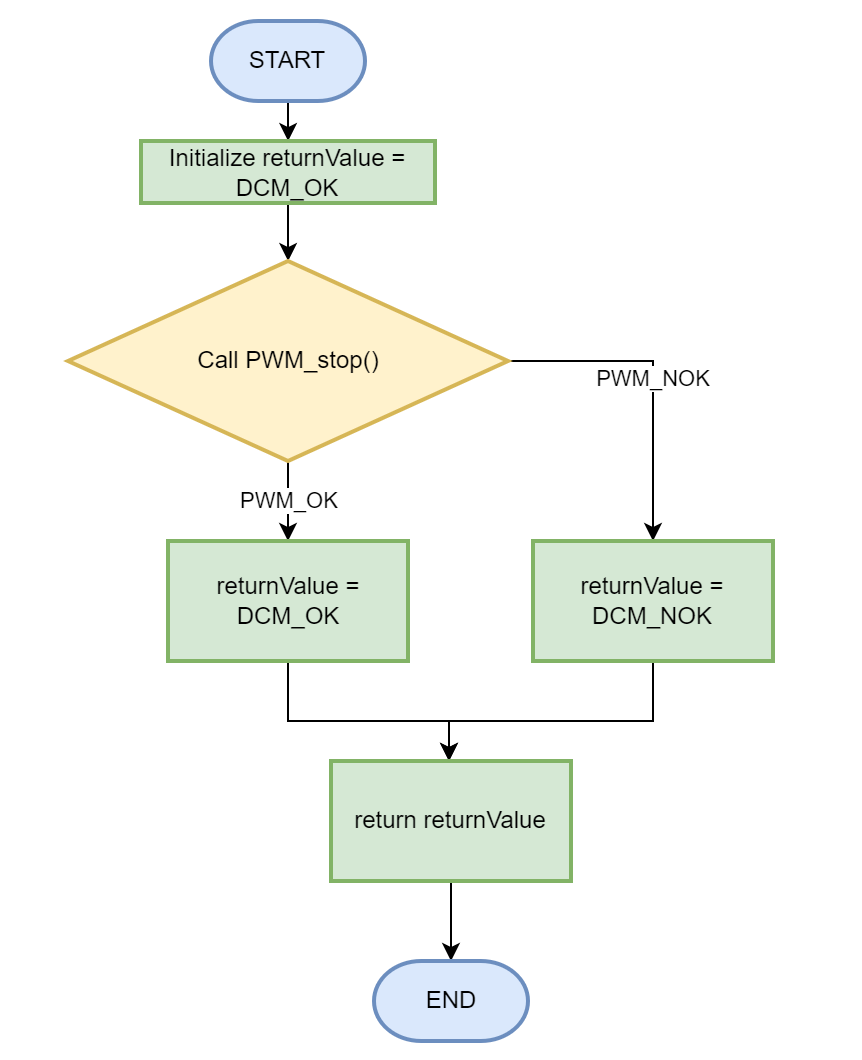


#### 

#### DCM\_start



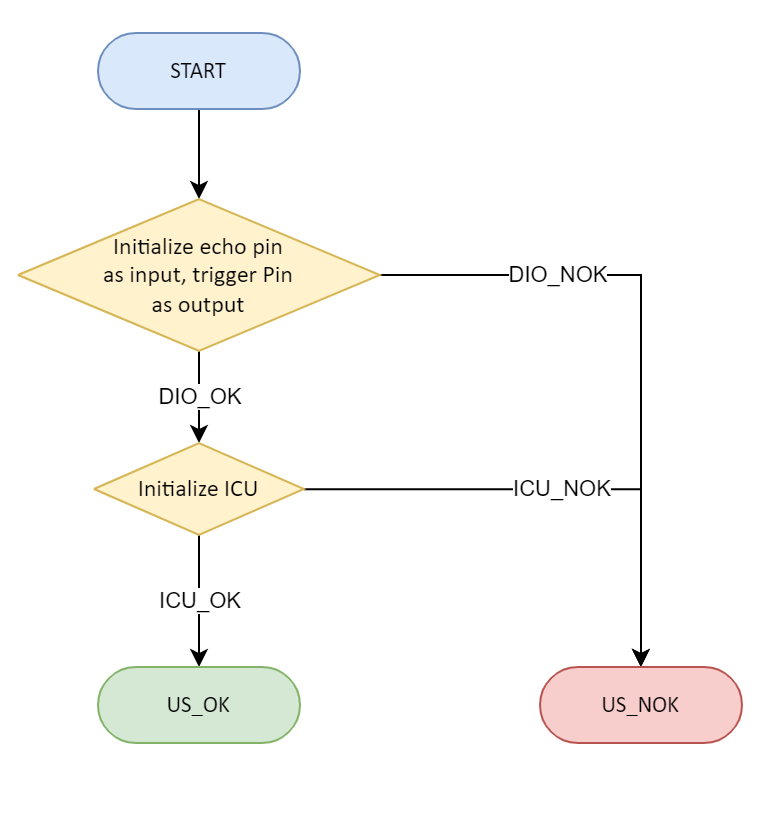
#### DCM\_stop



### 

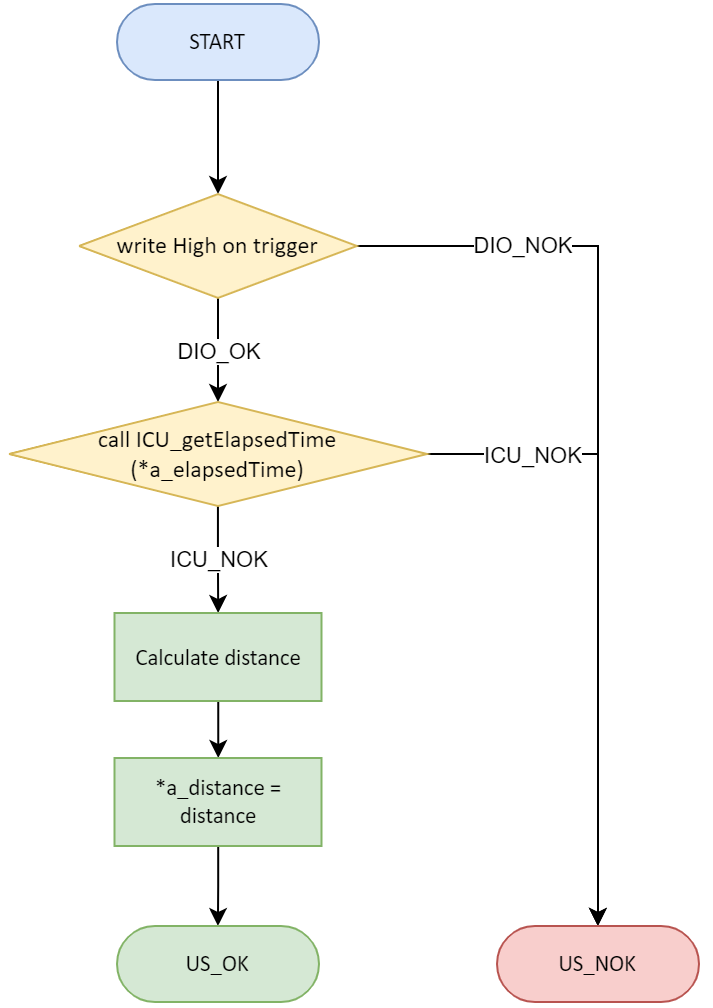
### Ultrasonic Module

#### US\_init



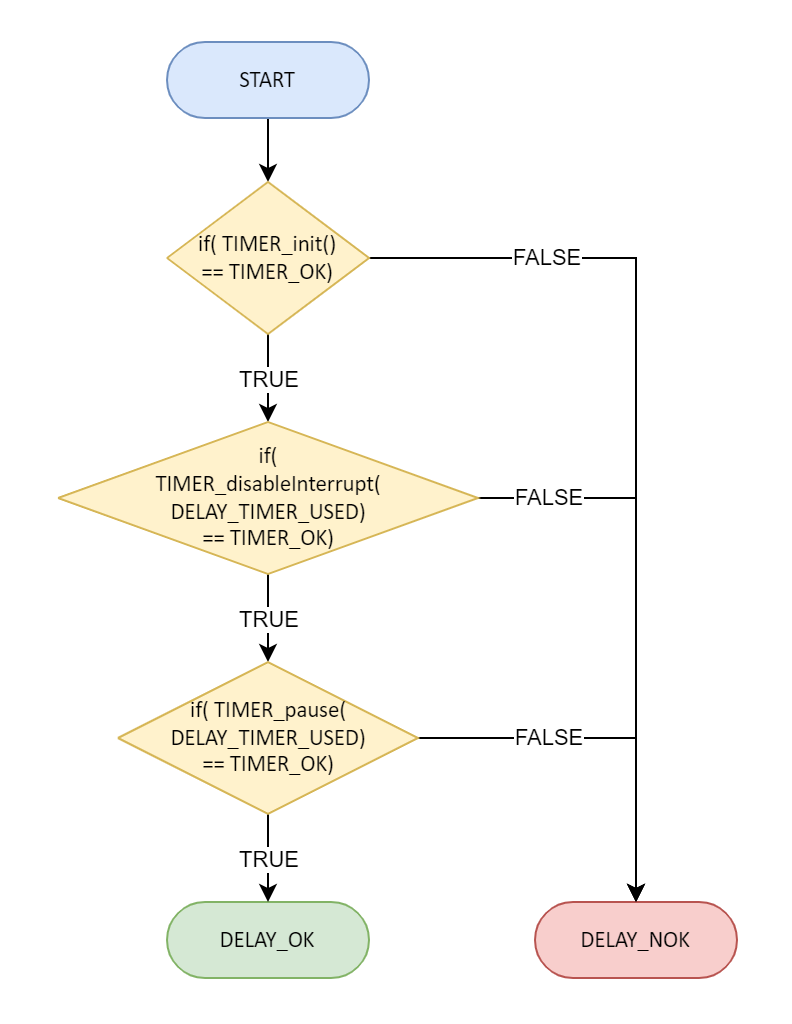
#### 

#### US\_getDistance



### Delay Module

#### DELAY\_init



#### DELAY\_setTime



#### DELAY\_setTimeBlocking

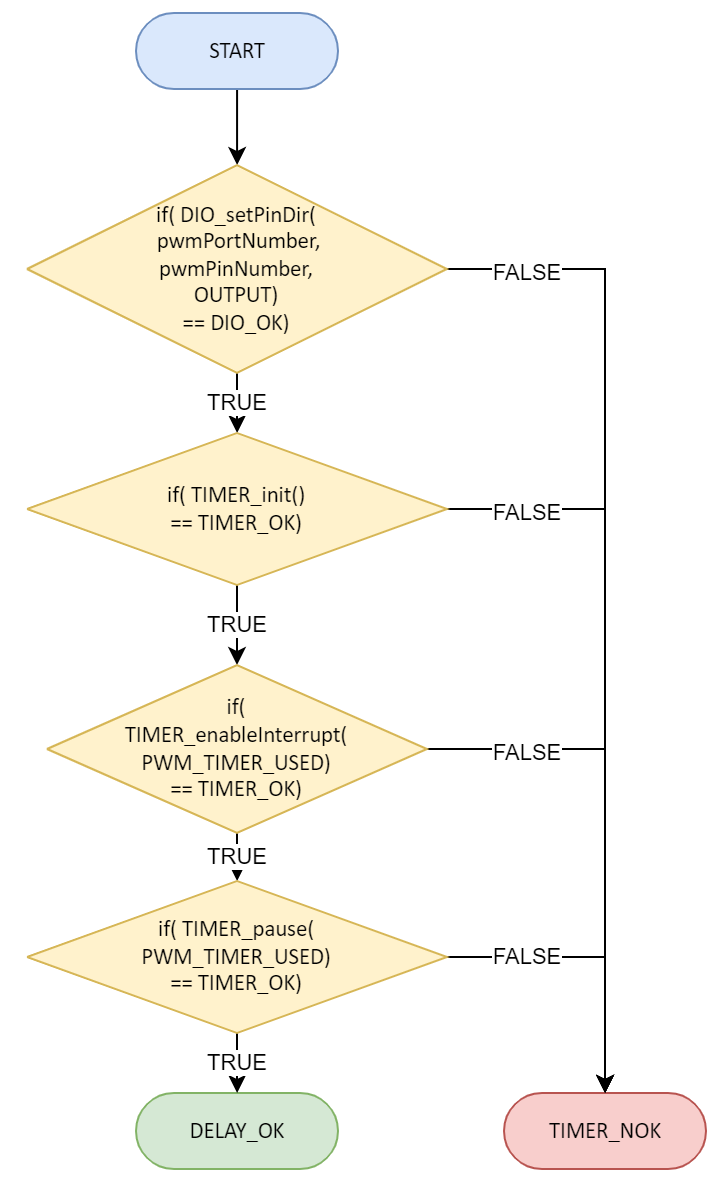


#### DELAY\_setTimeBlocking

## 

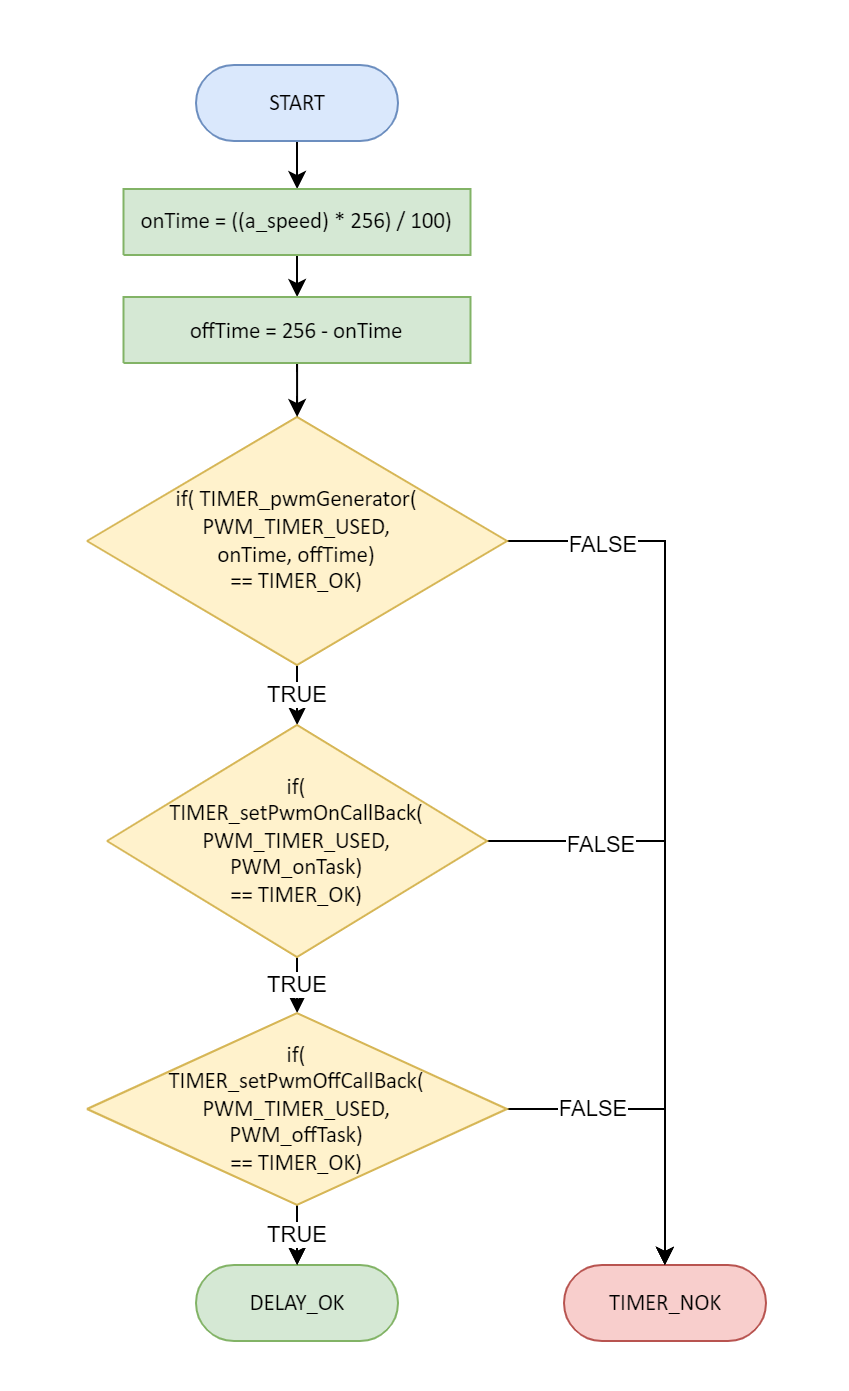
### PWM Module

#### PWM\_init

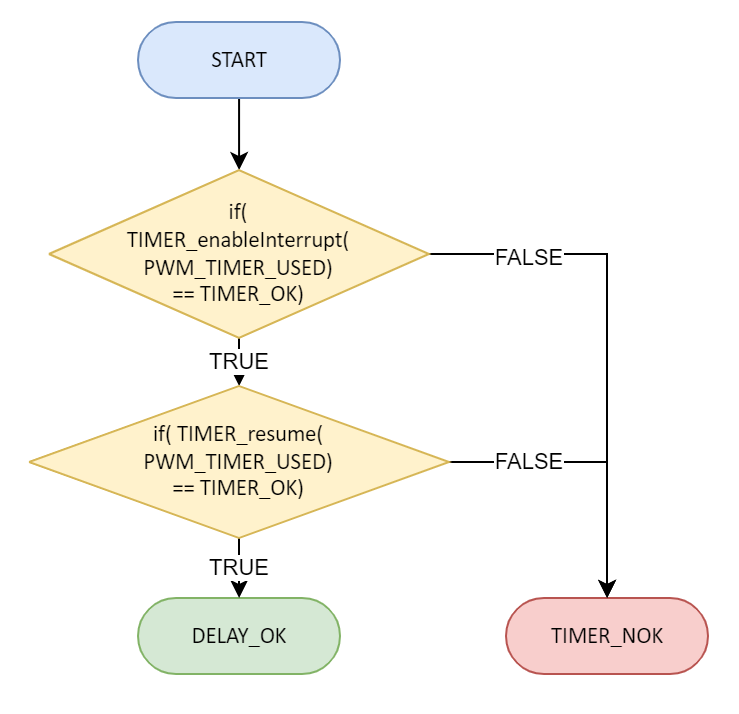


#### 

#### PWM\_setDutyCycle

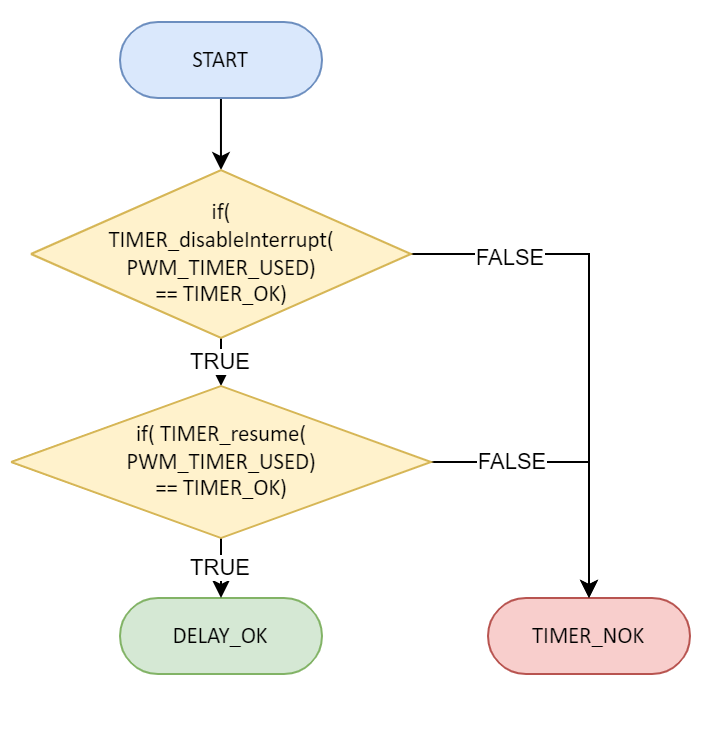


#### PWM\_start



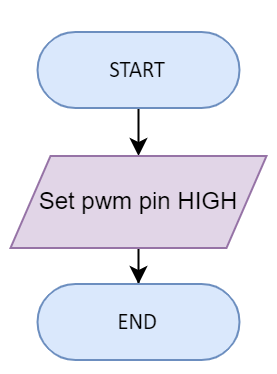
#### 

#### PWM\_stop



#### 

#### PWM\_onTask



#### PWM\_offTask

## 

## APP Layer

### App State Diagram

### 

# 

# Pre-compiling and linking configurations

## EXTINT Driver

### Linking Configurations

|  |
| --- |
| const st\_EXTINT\_config\_t EXTINT\_config [NUMBER\_OF\_TIMERS\_USED] =  { */\* EXTINT\_number, senseControlUsed\*/*  {INT\_0, EXTINT\_FALLING\_EDGE},  {INT\_1, EXTINT\_FALLING\_EDGE},  {INT\_2, EXTINT\_RISING\_EDGE} }; |

### Pre\_compiling Configurations

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_NUMBER\_OF\_EXTINTS\_USED\_/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* number of external interrupts used  \*/*  #define NUMBER\_OF\_EXTINTS\_USED 3 |

## Timer Driver

### Linking configurations

|  |
| --- |
| const st\_TIMER\_config\_t st\_TIMER\_config [NUMBER\_OF\_TIMERS\_USED] =  { */\* TIMER\_number, waveformUsed, prescalerUsed \*/*  {TIMER\_0, TIMER\_OV, TIMER\_PRESCLNG\_64},  {TIMER\_1, TIMER\_OV, TIMER\_PRESCLNG\_64},  {TIMER\_2, TIMER\_OV, TIMER\_PRESCLNG\_64} }; |

### 

### Pre-compiled Configurations

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SYSTEM\_OSCILLARTOR\_CLOCK\_FREQUENCY\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Enter microcontrollers frequency in Hz writing UL besides it  \*/* #define F\_CPU 8000000UL  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_NUMBER\_OF\_TIMERS\_USED\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* number of timers used  \*/* #define NUMBER\_OF\_TIMERS\_USED 3 |

## 

## ICU Driver

### Linking Configuration

|  |
| --- |
| */\* Predefined Pin options \*/* st\_ICU\_capturePins\_t st\_ICU\_predefinedPins = {  .PORT\_D\_PIN\_2 = {  .capturePin = PIN\_2,  .capturePort = PORT\_D,  .interruptNo = INT0  },  .PORT\_D\_PIN\_3 = {  .capturePin = PIN\_3,  .capturePort = PORT\_D,  .interruptNo = INT1  },  .PORT\_B\_PIN\_2 = {  .capturePin = PIN\_2,  .capturePort = PORT\_B,  .interruptNo = INT2  } };  */\* Configuration \*/* static st\_ICU\_config\_t st\_gs\_icuConfig = {  *// Capture Pin*  .icuCapturePin = PORT\_B\_PIN\_2,   *// Callback Function on time received*  .timeReceivedCallbackFun = NULL *// callback function* }; */\* Functions \*/* st\_ICU\_config\_t ICU\_getConfig() {  switch (st\_gs\_icuConfig.icuCapturePin) {  case PORT\_D\_PIN\_2:  st\_gs\_icuConfig.icuCapturePinData = st\_ICU\_predefinedPins.PORT\_D\_PIN\_2;  break;  case PORT\_D\_PIN\_3:  st\_gs\_icuConfig.icuCapturePinData = st\_ICU\_predefinedPins.PORT\_D\_PIN\_3;  break;  case PORT\_B\_PIN\_2:  st\_gs\_icuConfig.icuCapturePinData = st\_ICU\_predefinedPins.PORT\_B\_PIN\_2;  break;  }  return st\_gs\_icuConfig; }  void ICU\_setConfig(st\_ICU\_config\_t st\_a\_icuConfig) {  st\_gs\_icuConfig = st\_a\_icuConfig; } |

## 

## PWM Driver

### Linking Configurations

|  |
| --- |
| const st\_PWM\_config\_t st\_PWM\_config [NUMBER\_OF\_PWM\_PINS] =  { */\* PWM\_# PORT0\_#, PIN0\_#\*/*  {PWM\_0, DIO\_PORTD, DIO\_PIN\_4},  {PWM\_1, DIO\_PORTD, DIO\_PIN\_5} }; |

### Pre-compiled Configurations

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SYSTEM\_OSCILLARTOR\_CLOCK\_FREQUENCY\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Enter microcontrollers frequency in Hz writing UL besides it  \*/* #define F\_CPU 8000000UL  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_TIMER\_USED\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Timer used to generate delay  \*/* #define PWM\_TIMER\_USED TIMER\_0  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_NUMBER\_OF\_PWM\_PINS\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Number of pwm pins used to generate PWM signal on them  \*/* #define NUMBER\_OF\_PWM\_PINS 2 #define PWM\_0 0 #define PWM\_1 1 |

## Delay Driver

### Pre-compiled Configurations

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SYSTEM\_OSCILLARTOR\_CLOCK\_FREQUENCY\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Enter microcontrollers frequency in Hz writing UL besides it  \*/* #define F\_CPU 8000000UL  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_TIMER\_USED\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* Timer used to generate delay  \*/* #define DELAY\_TIMER\_USED TIMER\_2 |

## DCM Driver

### Linking Configurations

|  |
| --- |
| const st\_DCM\_config\_t st\_DCM\_config [NUMBER\_OF\_DCMS\_USED] =  { */\* DCM\_#, PORT0\_#, PIN0\_#\*, PORT1\_#, PIN1\_# \*/*  {DCM\_0, DIO\_PORTD, DIO\_PIN\_2, DIO\_PORTD, DIO\_PIN\_3},  {DCM\_1, DIO\_PORTD, DIO\_PIN\_6, DIO\_PORTD, DIO\_PIN\_7} }; |

### Pre-compiled Configurations

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_NUMBER\_OF\_DCMS\_USED\_\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* number of dcms used  \*/*  #define NUMBER\_OF\_DCMS\_USED 2 |

## US (ultrasonic) Driver

### Linking Configuration

|  |
| --- |
| static st\_US\_config\_t st\_gs\_usConfig = {  DIO\_PORTB, DIO\_P3, DIO\_P2 }; |

## 

## LCD Driver

### pre-compiling configuration

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SELECT\_LCD\_MODE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* OPTION 1-> [ \_4\_BIT\_MODE ]  \* OPTION 1-> [ \_8\_BIT\_MODE ]  \*/* #define LCD\_MODE \_4\_BIT\_MODE  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SELECT\_DATA\_REGISTER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* #define LCD\_DATA\_PORT DIO\_PORT\_A  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_SELECT\_MC\_PINS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* #define LCD\_RS\_PIN DIO\_PA1  #define LCD\_RW\_PIN DIO\_PA2 #define LCD\_EN\_PIN DIO\_PA3 #define LCD\_DATA\_4\_PIN DIO\_PA4 #define LCD\_DATA\_5\_PIN DIO\_PA5 #define LCD\_DATA\_6\_PIN DIO\_PA6 #define LCD\_DATA\_7\_PIN DIO\_PA7 |

## KEYPAD Driver

### pre-compiling configuration

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\* GLOBAL CONSTANT MACROS  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* #define KEYPAD\_NUMBER\_OF\_ROWS 3 #define KEYPAD\_NUMBER\_OF\_COLUMNS 3  #define KEYPAD\_R0\_PORT DIO\_PORT\_C #define KEYPAD\_R1\_PORT DIO\_PORT\_C #define KEYPAD\_R2\_PORT DIO\_PORT\_C   #define KEYPAD\_C0\_PORT DIO\_PORT\_C #define KEYPAD\_C1\_PORT DIO\_PORT\_C #define KEYPAD\_C2\_PORT DIO\_PORT\_C   #define KEYPAD\_R0\_PIN DIO\_PC2 #define KEYPAD\_R1\_PIN DIO\_PC3 #define KEYPAD\_R2\_PIN DIO\_PC4   #define KEYPAD\_C0\_PIN DIO\_PC5 #define KEYPAD\_C1\_PIN DIO\_PC6 #define KEYPAD\_C2\_PIN DIO\_PC7 |

### Linking configurations

|  |
| --- |
| const u8 u8\_arr\_g\_keypadArr [KEYPAD\_NUMBER\_OF\_ROWS][KEYPAD\_NUMBER\_OF\_COLUMNS] ={  {'1','2','3'}, {'4','5','6'}, {'7','8','9'} }; |

## BUTTON Driver

### pre-compiling configuration

|  |
| --- |
| */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\_NUMBER\_OF\_BUTTONS\_USED\_/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* */\*  \* number of buttons used  \*/*  #define NUMBER\_OF\_BUTTONS\_USED 1 |

### Linking configurations

|  |
| --- |
| const st\_BUTTON\_config\_t st\_BUTTON\_config [NUMBER\_OF\_MOTORS\_USED] =  { */\* BUTTON\_#, PORT\_#, PIN\_#\*/*  {BUTTON\_0, DIO\_PORT\_C, DIO\_PC4} }; |