# Obstacle Avoidance Robot

# V1.0 Design

By: Alaa Hisham

Team: 1

Date: 14-5-2023

# Introduction

The purpose of this design document is to outline the development of an obstacle avoidance robot. The robot is designed to navigate through an environment and avoid obstacles in its path using a combination of sensors and motor control.

The design of the robot is based on a modular approach where this document provides a detailed description of the system architecture and software design. The information contained in this document is intended to provide a comprehensive overview of the robot's design and development, and to serve as a reference for future development and improvements.

# High Level Design

## I. Layered Architecture

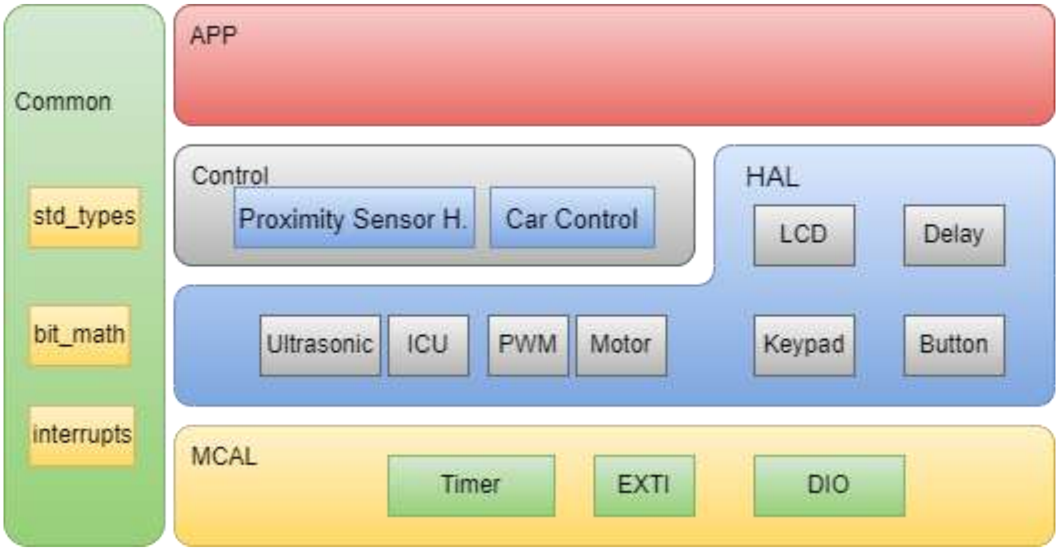


*Figure 1 System Layered Architecture*

# II. Module Description

## A. MCAL

- **DIO**

  The DIO module is a peripheral module used to provide digital input and output capabilities to interface with the external devices such as sensors, switches, LEDs, and other microcontrollers.

- **EXTI**

  A peripheral module that allows the processor to respond to external events in real-time. It provides a mechanism for the processor to interrupt its normal execution flow and execute a pre-defined interrupt service routine (ISR) in response to a specific event occurring on an external pin or input.

- **Timer**

  A peripheral module that provides precise timing capabilities for various applications. It consists of three hardware timers that are controlled and configured by software.

## B. HAL

- **Motor**

  A simple module to control a given motor.

- **Button**

  A simple module to interface with a given push button on an EXTI Pin.

- **Keypad**

  A module to interface with a keypad and read each pressed key.

- **LCD**

  A module for interfacing with a character LCD to display messages and data to user.

- **Ultrasonic**

  A simple module to interface with an ultrasonic sensor.

- **PWM**

  A software-implemented module for pulse width modulation to control the speed.

- **ICU**

  A Software-implemented input capture unit module to measure pulse times.

## C. CONTROL

- **Car Control**

  A module that provides a high-level interface to control the robot movement with respect to direction and speed.

- **Proximity Sensor Handler**

  A module that uses both ultrasonic and ICU modules to provide a high-level interface for measuring distances.

## D. COMMON

- **STD Types**

  Includes the standard types used in all layers.

- **Bit Math**

  Provides function like macros for bitwise operations.

- **Interrupt Table**

  Includes all interrupt vectors and provides macros for dealing with general interrupt.

# III. Driver Documentation

## A. MCAL

### • DIO APIs

```
/**
 * @def function to configure all pin directions and initial values
 *      (for DIO driver with pre-compile/linking configurations)
 */
void DIO_Init(void);

/**
 * @def function to configure a single DIO pin as input/output
 * @param Copy_u8Port the port of the required pin
 * @param Copy_u8Pin the pin number in the given port
 * @param Copy_u8Value desired value (high or low) to set the pin to
 * @return error status
 */
en_DIO_Error_t DIO_SetPinVal(u8 Copy_u8Port,u8 Copy_u8Pin, u8 Copy_u8Value);

/**
 * @def function to configure a single DIO pin as input/output
 * @param Copy_u8Port the port of the required pin
 * @param Copy_u8Pin the pin number in the given port
 * @param Copy_u8Value desired value (high or low) to set the entire port to
 * @return error status
 */
en_DIO_Error_t DIO_SetPortVal(u8 Copy_u8Port, u8 Copy_u8Value);

/**
 * @def function to get the value of a single DIO pin whether high or low
 * @param Copy_u8Port the port of the required pin
 * @param Copy_u8Pin the pin number in the given port
 * @param Copy_pu8Val pointer to a variable to store pin value (0-255)
 * @return error status
 */
en_DIO_Error_t DIO_GetPinVal(u8 Copy_u8Port,u8 Copy_u8Pin, u8* Copy_pu8Val);
```

### • EXTI APIs

```
/**
 * @def         Initialize given exti with given Sense mode
 * @param u8_a_IntNumber: the interrupt to be configured
 * @param u8_a_SenseMode: the exti trigger event
 * @return error state
 */
en_EXTIerror_t EXTI_Init(en_EXTI_t u8_a_IntNumber, en_SenseMode_t u8_a_SenseMode);

/**
 * @def         Function to enable/disable given exti
 * @param u8_a_IntNumber: the interrupt to be configured
 * @param en_a_intState: exti state
 * @return error state
 */
en_EXTIerror_t EXTI_SetState(en_EXTI_t u8_a_IntNumber, en_EXTI_State_t en_a_intState);
```

```
/**
 * @def   Function to set a CBF to call when exti is triggered
 * @param u8_a_IntNumber : the desired exti
 * @param pv_a_Function: The function to call
 * @return error state
 */
en_EXTIerror_t EXTI_u8SetCallbackFn(en_EXTI_t u8_a_IntNumber, void
(pv_a_Function)(void));
```

- **Timer APIs**

```
/**
 * @def   Function to Initialize the timer with given mode
 * @param u8_a_Mode: Mode to initialize the timer
 * @return error state
 */
en_TIMErrorState_t TIM0_voidInit(en_TIMMode_t u8_a_Mode);

/**
 * @def   Start the timer clock after prescaling it with given value
 * @param u8_a_prescaler: prescaler to adjust the timer clk
 * @return error state
 */
en_TIMErrorState_t TIM_Start(en_TIM_CLK_SELECT_t u8_a_prescaler);

/**
 * @def Stop the timer
 */
void TIM_Stop();

/**
 * @def   Set the timer to start from the given value
 * @param u8_a_startValue: the value to start from
 */
void TIM0_SetValue(u8 u8_a_startValue);

/**
 * @def   Function to read the value of the ovf flag
 * @param u8_a_FlagValue: ref to var to store flag value
 * @return error state
 */
en_TIMErrorState_t TIM_GetOVF(Uint8_t* u8_a_FlagValue);

/**
 * @def Function to clear timer 0 overflow flag
 */
void TIM_ClearOVF(void);

/**
 * @def   Get whether timer is Running or Stopped
 * @param pen_a_State : ptr to var to store timer state
 * @return error state
 */
en_TIMErrorState_t TIM_GetState(en_TIMState_t* pen_a_State);

/**
 * @def Enable timer overflow interrupt
 */
void TIM0_EnableOVFInterrupt(void);
```

```
/**
 * @def Disable timer overflow interrupt
 */
void TIM_DisableOVFInterrupt(void);

/**
 * @def   set a function to call when the timer0 OVF interrupt is triggered
 * @param pv_a_CallbackFn: reference to the function to call
 * @return error state
 */
en_TIMErrorState_t TIM0_SetOVFCallback(void (*pv_a_CallbackFn)(void));
```

## B. HAL

- **Motor APIs**

```
/**
 * @def   Function to Initialize the motor
 * @param pst_a_Motor: reference to motor
 * @return error state
 */
en_MotorError_t DCM_voidInit(st_Motor_t* pst_a_Motor);

/**
 * @def Start the given motor
 * @param pst_a_Motor: reference to motor
 * @return error state
 */
en_MotorError_t DCM_Start(st_Motor_t* pst_a_Motor);

/**
 * @def Stop the given motor
 * @param pst_a_Motor: reference to motor
 * @return error state
 */
en_MotorError_t DCM_Stop(st_Motor_t* pst_a_Motor);
```

- **Button APIs**

```
/**
 * @def   Function to Initialize the button and set notification function
 *         for when the button is pressed
 * @param pst_a_button: reference to button
 * @return error state
 */
en_ButtonError_t BTN_Init(st_Button_t* pst_a_button, void (*pv_a_Notification)(void));
```

- **Keypad APIs**

```
/**
 * @def Function to Initialize the keypad and pins
 */
void KPD_Init(void);

/**
 * @def    Function to check if a key is pressed and return it
 * @return the pressed key (or NO_KEY if no key is pressed)
 */
 Uint8_t KPD_getKey(void);
```

- **LCD APIs**

```
/**
 * @def Initialize the LCD
 */
void LCD_Init(void);

/**
 * @def   Function to display the given character on the LCD
 * @param u8_a_char: Character to display
 */
void LCD_WriteChar(u8 u8_a_char);

/**
 * @def   Function to move the cursor to a certain position
 * @param u8_a_row   :
 * @param u8_a_column
 */
void LCD_GoToXY(en_LCDrow_t u8_a_row, en_LCDcol_t u8_a_column);

/**
 * @def  Function to display a string to the LCD
 * @param pchar_a_str: reference to the string to display
 */
void LCD_WriteStr(const char* Copy_pcData);

/**
 * @def   Function to display a number on the LCD
 * @param u32_a_Number: Number to display
 */
void LCD_WriteNumber(Uint32_t u32_a_Number);
```

- **Ultrasonic APIs**

```
/**
 * @def Function to Initialize the Ultrasonic sensor pins
 */
void US_Init(void);

/**
 * @def Function to set the sensor trigger pin
 */
 void US_Trigger(void);

/**
 * @def Function to clear the sensor trigger pin
 */
 void US_DeTrigger(void);
```

- **Delay APIs**

```
/**
 * @def   Generate Synchronous delay (busy waiting)
 *
 * @param Copy_delayTime
 * @param Copy_timeUnit
 *
 * @return EN_TIMErrorState_t
 */
en_HTIMErrorState_t HTIM0_SyncDelay(Uint32_t u32_a_delay, en_timeUnits_t u8_a_timeUnit);

/**
 * @def   Generate an asynchronous delay using the OVF interrupt
 * @param u32_a_delay
 * @param u8_a_timeUnit
 * @param Copy_pvCallbackFn
 * @return en_TIMErrorState_t
 */
en_HTIMErrorState_t HTIM0_AsyncDelay(Uint32_t u32_a_delay, en_timeUnits_t u8_a_timeUnit,
void (*Copy_pvCallbackFn)(void));

/**
 * @def   Function to end an asynchronous delay
 * (should be called in an ISR/Callback function to end sync. delay)
 * @return void
 */
void HTIM0_AsyncEndDelay();
```

- **PWM APIs**

```
/**
 * @def Initialize the sensor pins
 */
void PWM_Init(void);

/**
 * @def   Function to generate signal with given parameters
 * @param u16_a_frequency: desired signal frequency
 * @param u8_a_dutyCycle : desired duty cycle
 */
void PWM_GenerateSignal(u16 u16_a_frequency, u8 u8_a_dutyCycle);

/**
 * @def Set the trigger pin to Low
 */
void PWM_StopSignal(void);
```

- **ICU APIs**

```
/**
 * @def Function to initialize the SW ICU
 */
void ICU_Init(void);

/**
 * @def   Function to get the elapsed time until certain external event
 * @param pu16_a_msTime: reference to variable to store elapsed time in ms
 * @return error state
 */
en_ICUerror_t ICU_GetTime(Uint16_t *pu16_a_msTime);
```

## C. CONTROL

- **Car Control APIs**

```
/**
 * @def Function to initialize Robot for movement
 */
en_RobotError_t ROBOT_Init();

/**
 * @def   Function to move the robot in a given direction with certain speed
 * @param en_a_dir:   Direction to move the robot
 * @param u8_a_speed: (0-100) percentage of the robot's max speed
 * @return error state
 */
en_RobotError_t ROBOT_Move(en_ROBOT_Dir_t en_a_dir, Uint8_t u8_a_speed);

/**
 * @def   Function to rotate the robot left or right
 * @param en_a_direction: rotation direction
 * @param u8_a_speed: rotation speed (0-100)
 * @return error state
 */
en_RobotError_t ROBOT_Stop(en_Direction_t en_a_direction, Uint8_t u8_a_speed);
```

- **Proximity Sensor Handler APIs**

```
/**
 * @def Initialize the sensor and ICU
 */
void HUS_Init(void);

/**
 * @def Set the trigger pin to Low
 * @return distance in cm
 */
Uint8_t HUS_GetDistance_cm(void);
```

## D. APP

```
/**
 * @def Function to initialize all modules used in app
 */
void APP_Init(void);

/**
 * @def The main logic of the application
 */
void APP_Start(void);
```

# Low Level Design

## I. API Flow Charts

### A. MCAL

- **DIO**

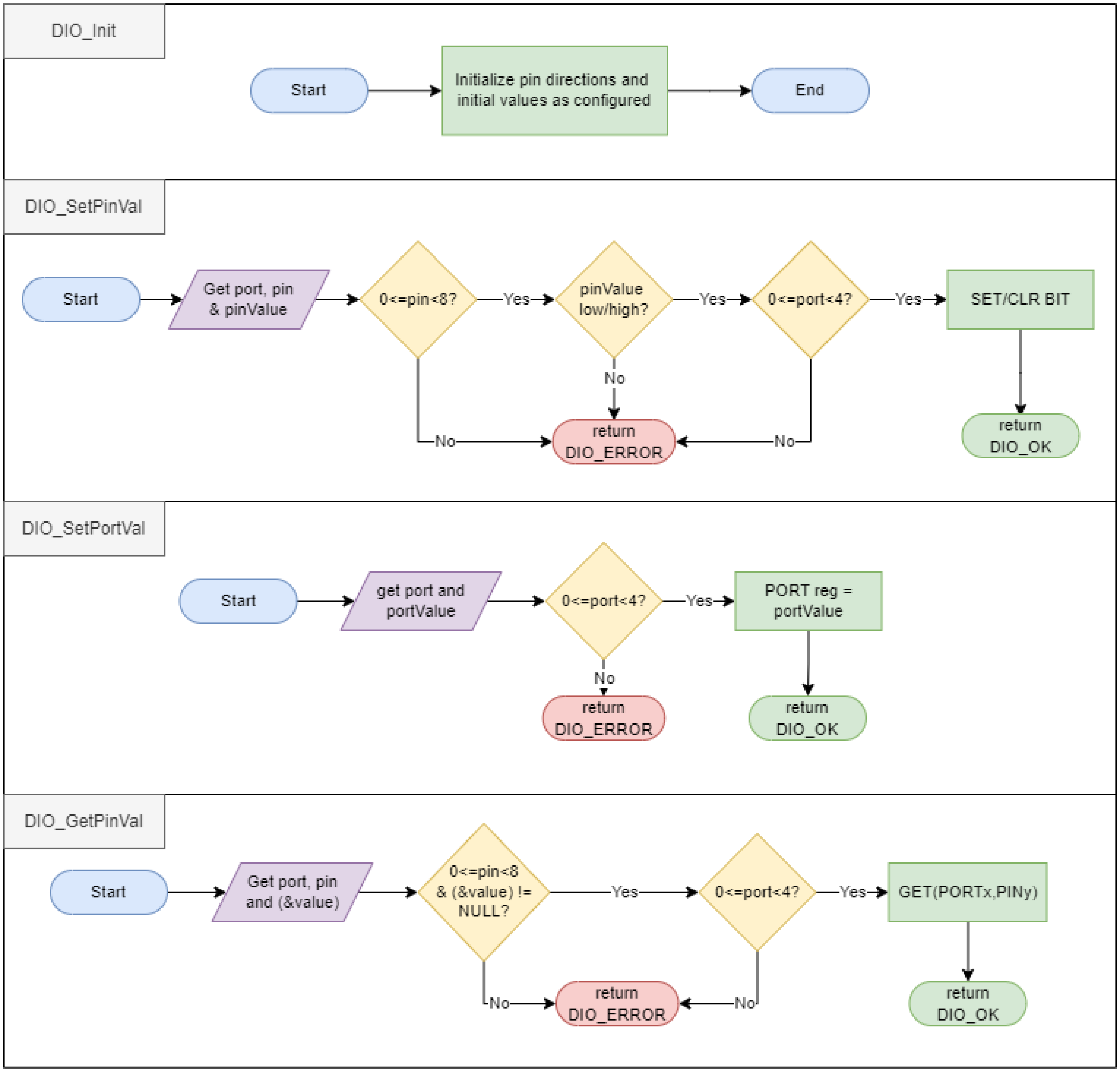


*Figure 2 DIO API flow charts*

- **EXTI**

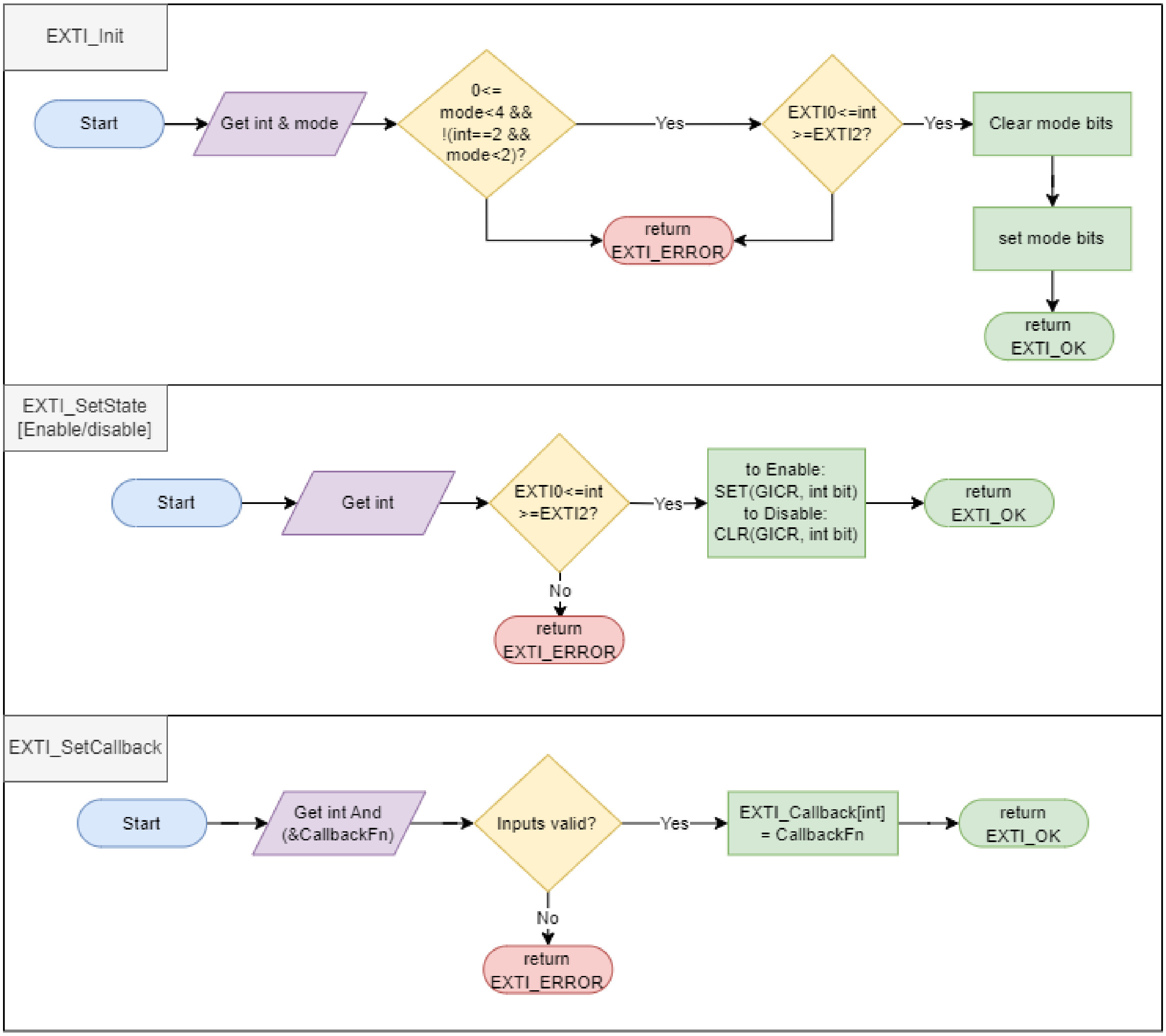
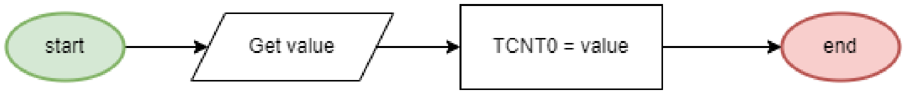


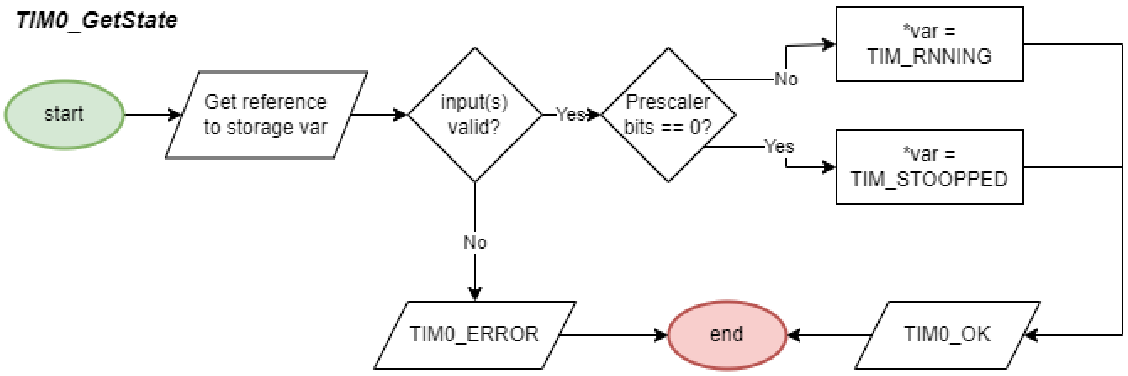
*Figure 3 EXTI API flow charts*

- ***Timer***.

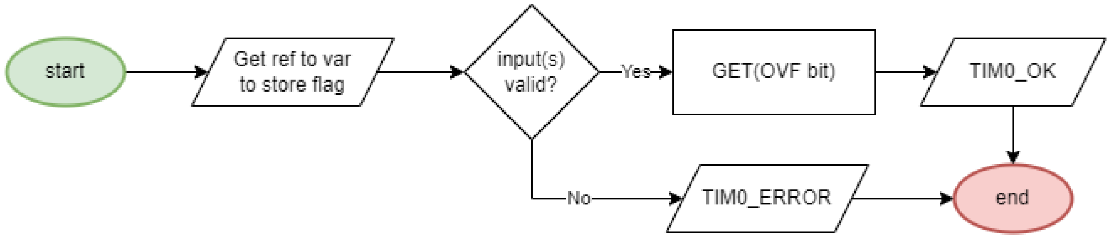
### *TIM0_SetValue*



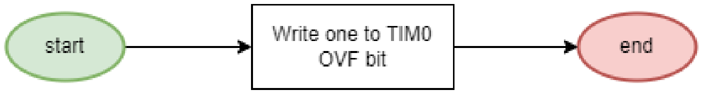
### *TIM0_GetState*



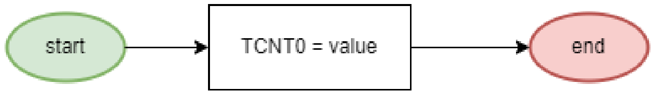
### *TIM0_GetOVF*



### *TIM0_ClearOVF*



### *TIM0_EnableOVFInt*



## B. HAL

- **Motor**

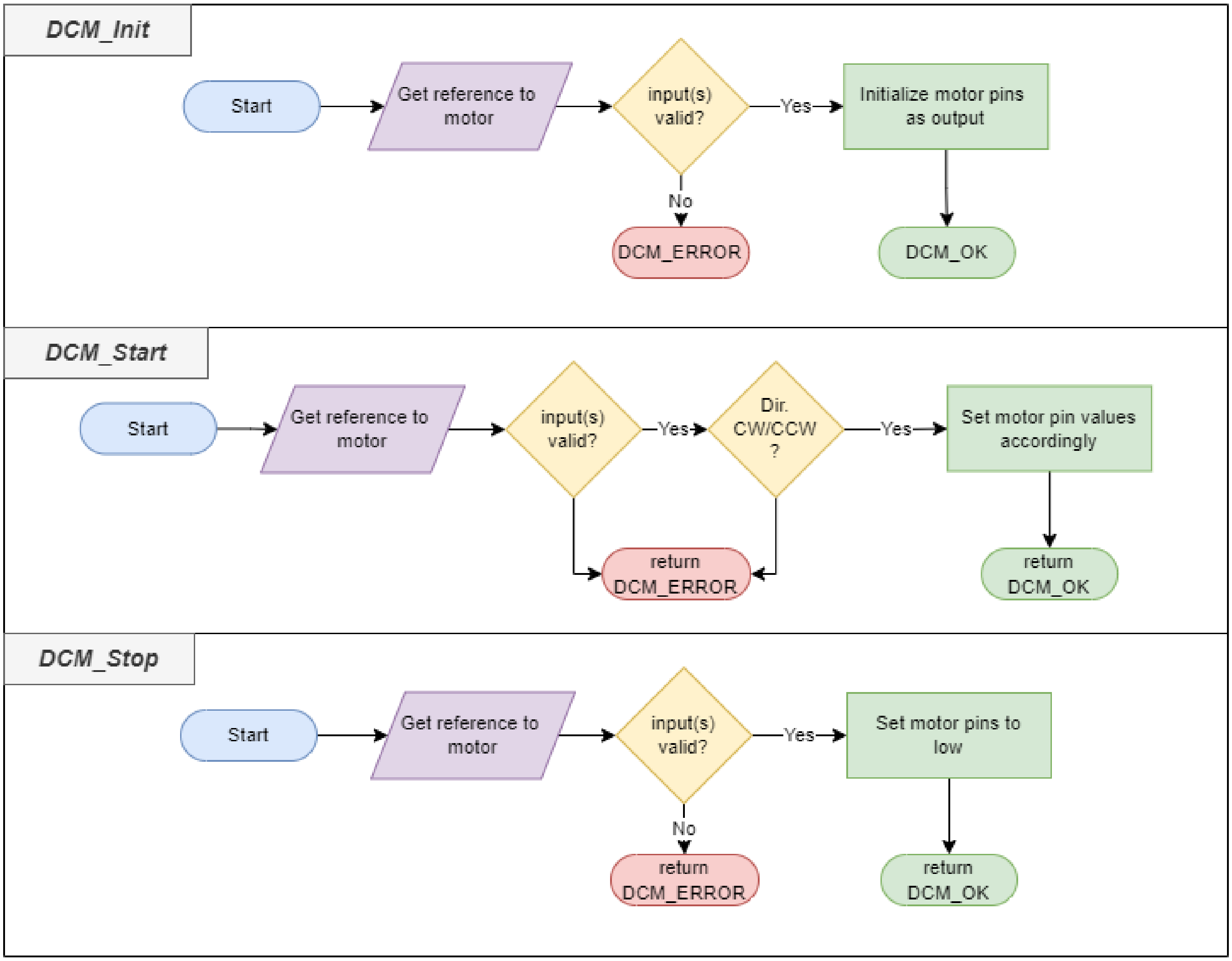


*Figure 4 Motor API Flow charts*

- **Button**

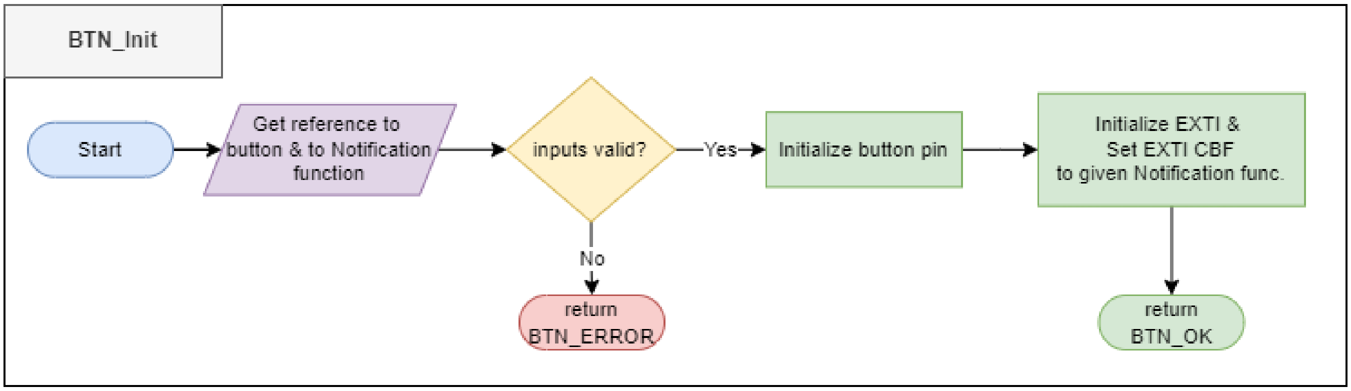

BTN_Init
Start
Get reference to button & to Notification function
inputs valid?
Yes
Initialize button pin
Initialize EXTI & Set EXTI CBF to given Notification func.
No
return BTN_ERROR
return BTN_OK

*Figure 5 Button (on EXTI pin) flow chart*

- **Keypad**

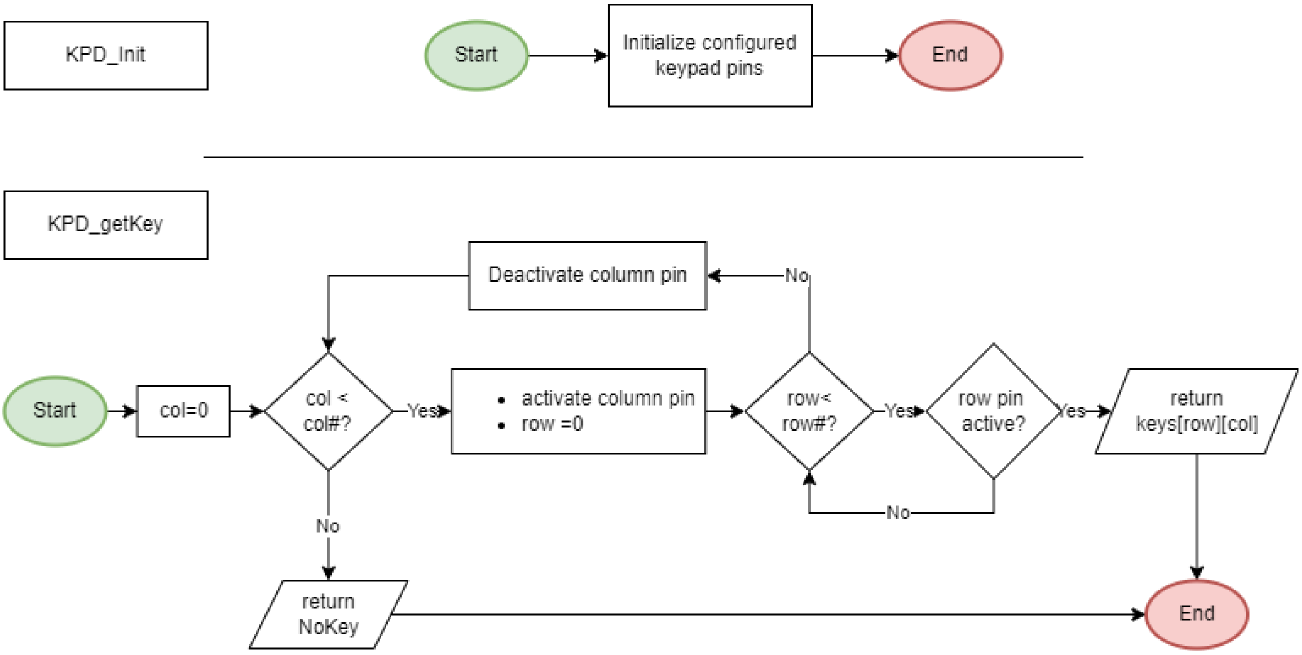


*Figure 6 Keypad API Flow charts*

- **Ultrasonic**

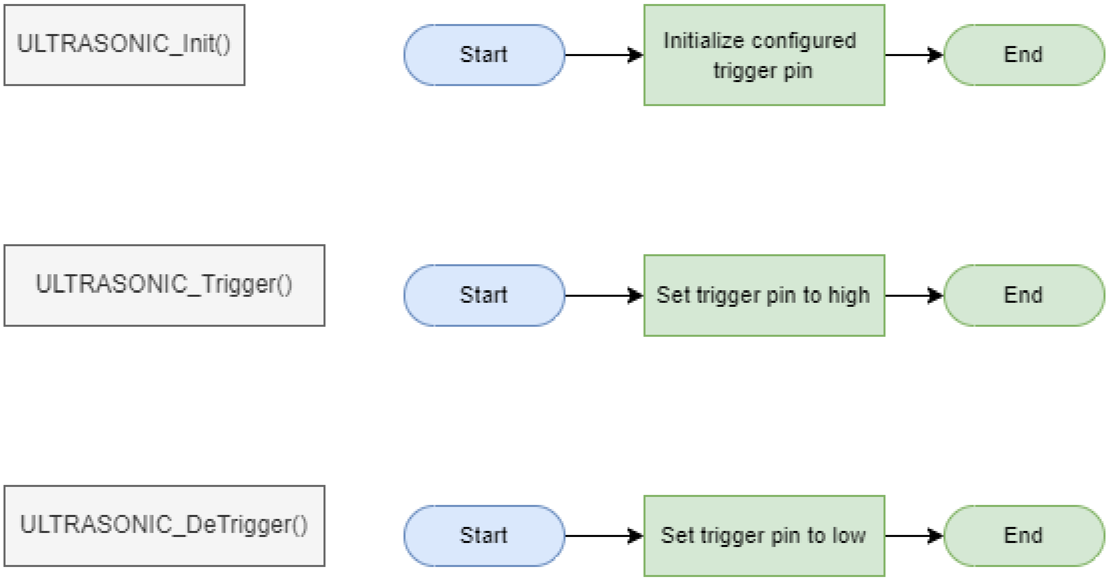


*Figure 7 Ultrasonic API flow charts*

- **PWM**

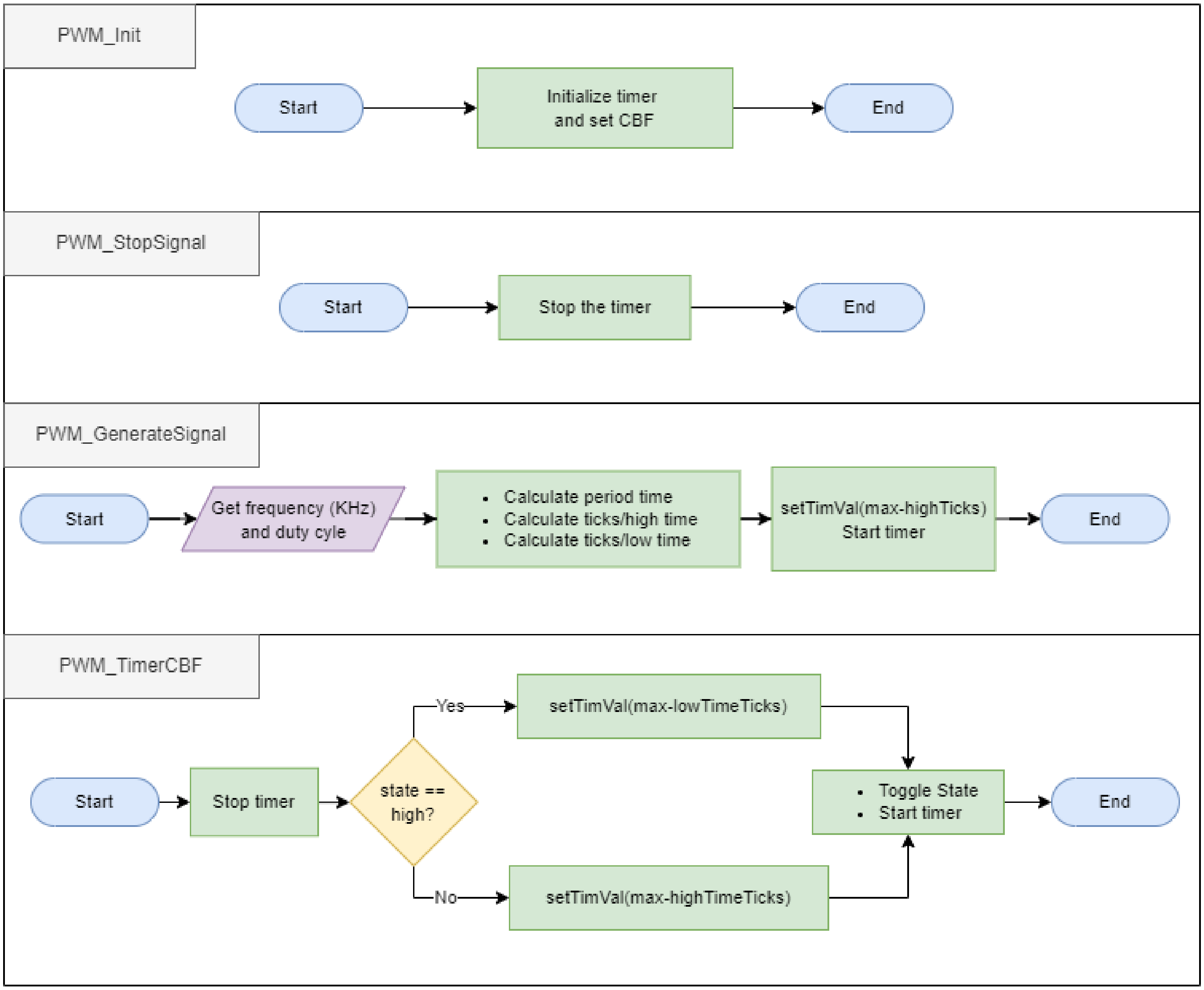


*Figure 8 PWM API flow charts*

- **ICU**

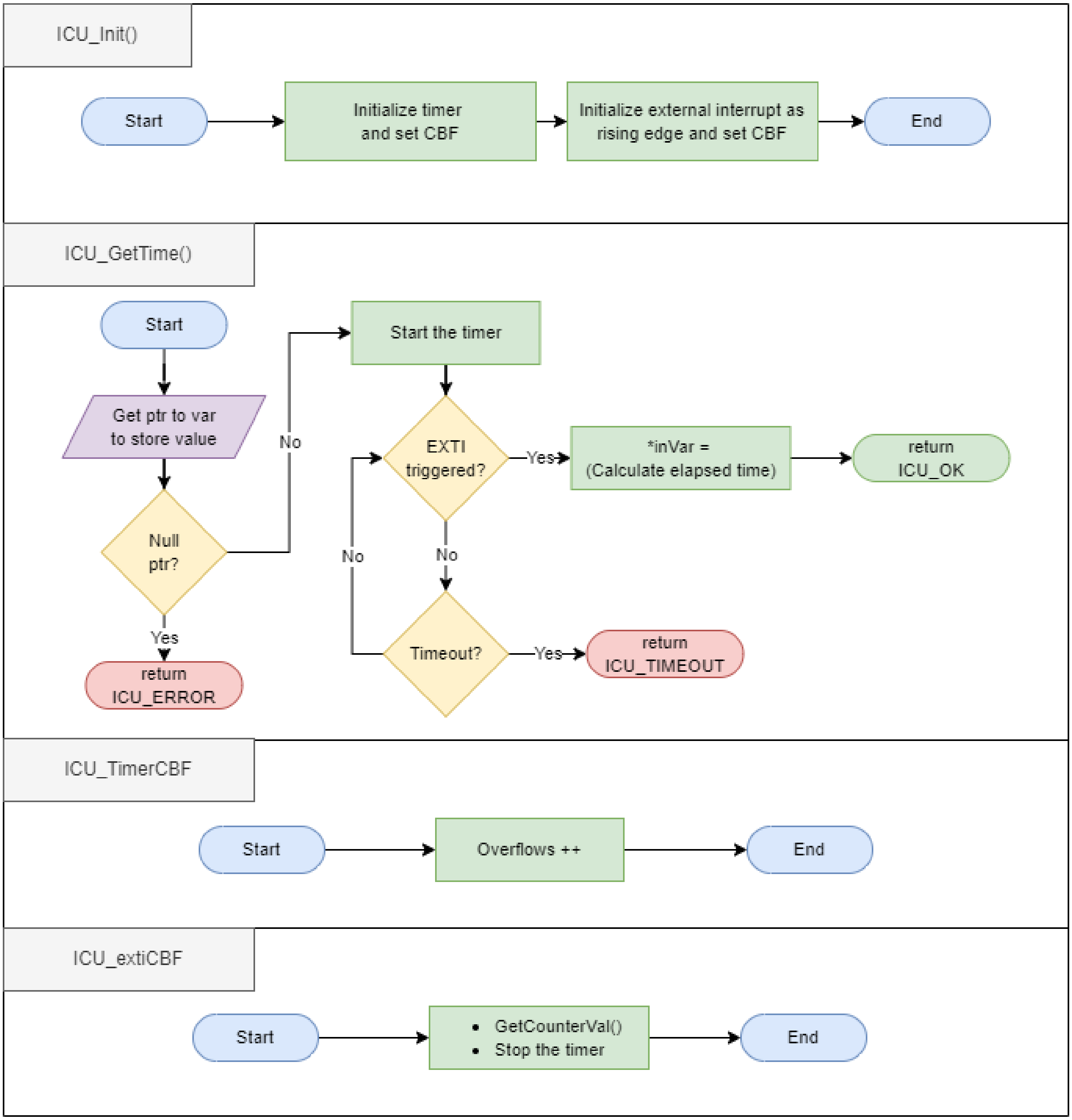


*Figure 9 ICU API flow charts*

## C. CONTROL

- **Robot Control**

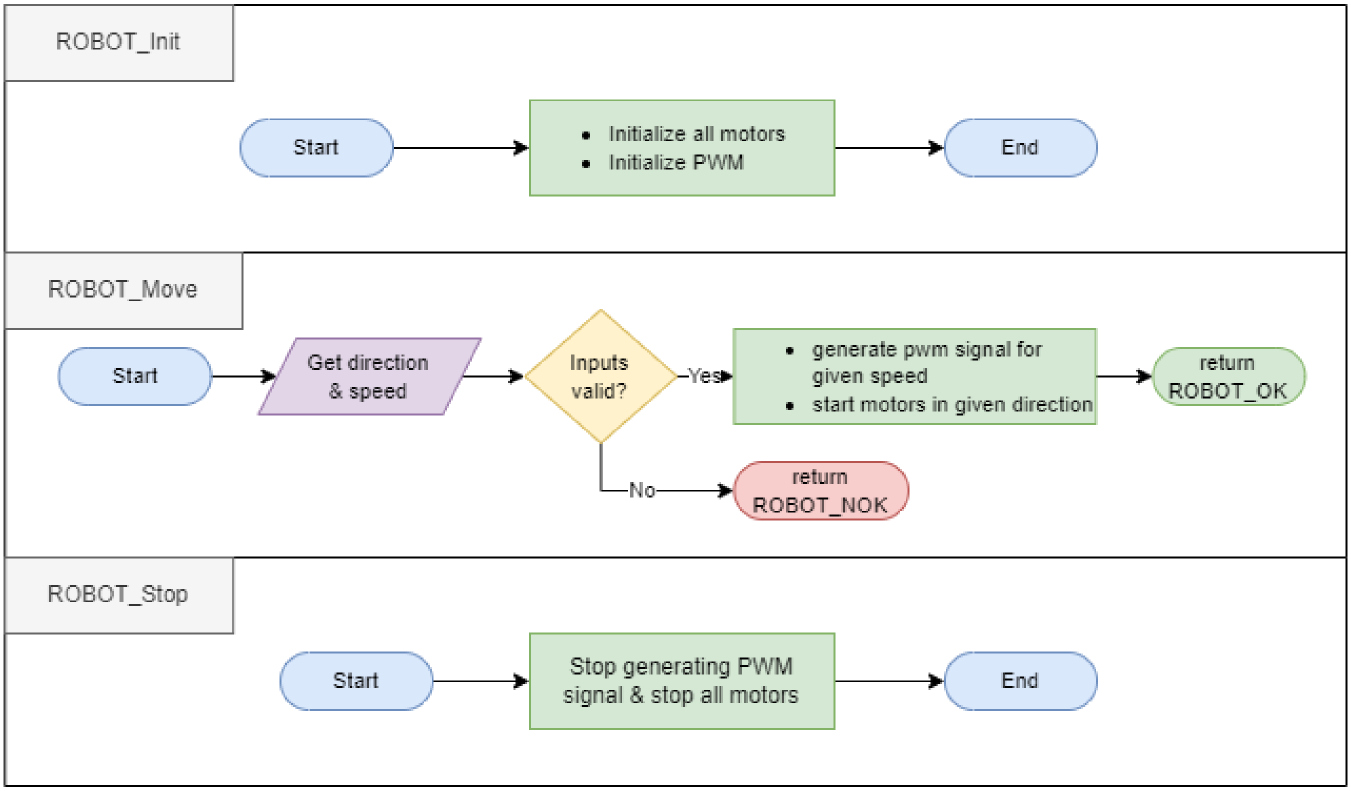


*Figure 10 Robot Control API flow charts*

- **Proximity Sensor Handler**

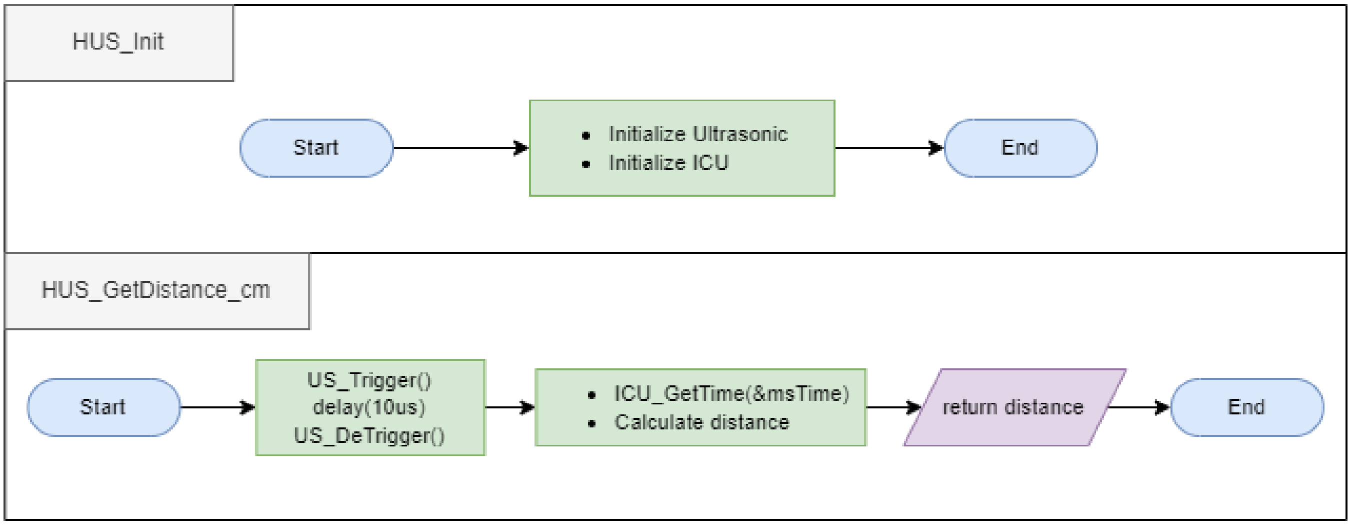


*Figure 11 Proximity Sensor Handler API flow charts*

## D. APP

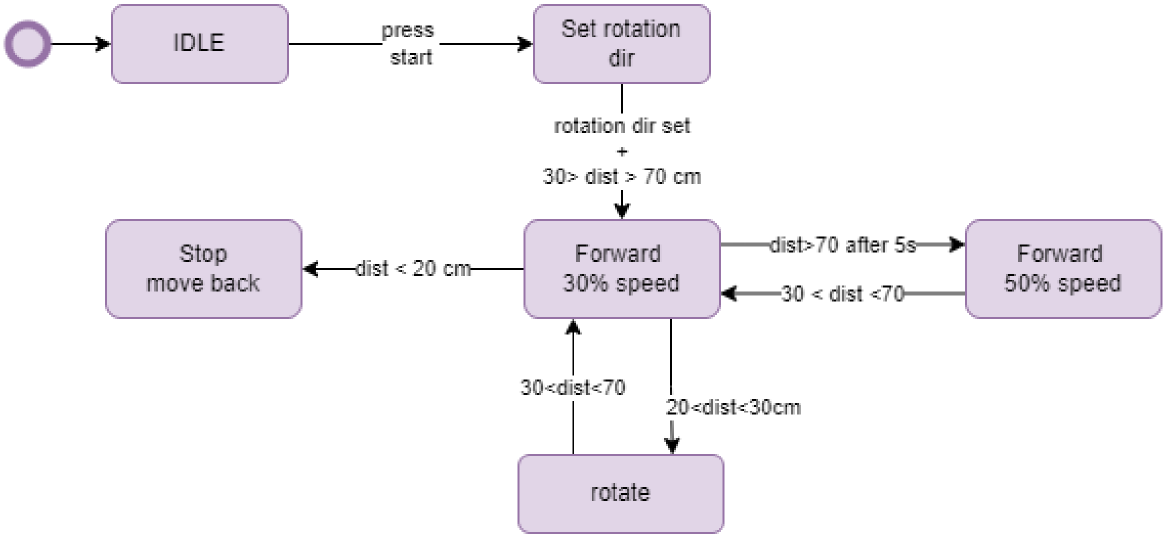


*Figure 12 App State Machine (first draft)*

*Note:*

*At the press of the stop button, we go from any state back to IDLE.*

# II. Module Pre-Compile & Linking Configurations

## A. MCAL

- **DIO**

```
 typedef enum
 {
     INPUT_PIN,
     OUTPUT_PIN
 }en_PinDir_t;

 typedef struct
 {
     en_PinDir_t PORTA_PIN0_DIR: 1;
     en_PinDir_t PORTA_PIN1_DIR: 1;
     en_PinDir_t PORTA_PIN2_DIR: 1;
     en_PinDir_t PORTA_PIN3_DIR: 1;
     en_PinDir_t PORTA_PIN4_DIR: 1;
     en_PinDir_t PORTA_PIN5_DIR: 1;
     en_PinDir_t PORTA_PIN6_DIR: 1;
     en_PinDir_t PORTA_PIN7_DIR: 1;
 }st_PortADirConfig_t;

 typedef struct
 {
     en_PinDir_t PORTB_PIN0_DIR: 1;
     en_PinDir_t PORTB_PIN1_DIR: 1;
     en_PinDir_t PORTB_PIN2_DIR: 1;
     en_PinDir_t PORTB_PIN3_DIR: 1;
     en_PinDir_t PORTB_PIN4_DIR: 1;
     en_PinDir_t PORTB_PIN5_DIR: 1;
     en_PinDir_t PORTB_PIN6_DIR: 1;
     en_PinDir_t PORTB_PIN7_DIR: 1;
 }st_PortBDirConfig_t;

typedef struct
{
     en_PinDir_t PORTC_PIN0_DIR: 1;
     en_PinDir_t PORTC_PIN1_DIR: 1;
     en_PinDir_t PORTC_PIN2_DIR: 1;
     en_PinDir_t PORTC_PIN3_DIR: 1;
     en_PinDir_t PORTC_PIN4_DIR: 1;
     en_PinDir_t PORTC_PIN5_DIR: 1;
     en_PinDir_t PORTC_PIN6_DIR: 1;
     en_PinDir_t PORTC_PIN7_DIR: 1;
}st_PortCDirConfig_t;

typedef struct
{
     en_PinDir_t PORTD_PIN0_DIR: 1;
     en_PinDir_t PORTD_PIN1_DIR: 1;
     en_PinDir_t PORTD_PIN2_DIR: 1;
     en_PinDir_t PORTD_PIN3_DIR: 1;
     en_PinDir_t PORTD_PIN4_DIR: 1;
     en_PinDir_t PORTD_PIN5_DIR: 1;
     en_PinDir_t PORTD_PIN6_DIR: 1;
     en_PinDir_t PORTD_PIN7_DIR: 1;
}st_PortDDirConfig_t;
```

```
typedef enum
{
    PIN_LOW,
    PIN_HIGH
}en_PinVal_t;

typedef struct
{
    en_PinVal_t PORTA_PIN0_INIT: 1;
    en_PinVal_t PORTA_PIN1_INIT: 1;
    en_PinVal_t PORTA_PIN2_INIT: 1;
    en_PinVal_t PORTA_PIN3_INIT: 1;
    en_PinVal_t PORTA_PIN4_INIT: 1;
    en_PinVal_t PORTA_PIN5_INIT: 1;
    en_PinVal_t PORTA_PIN6_INIT: 1;
    en_PinVal_t PORTA_PIN7_INIT: 1;
}st_PortAInitVal_t;

typedef struct
{
    en_PinVal_t PORTB_PIN0_INIT: 1;
    en_PinVal_t PORTB_PIN1_INIT: 1;
    en_PinVal_t PORTB_PIN2_INIT: 1;
    en_PinVal_t PORTB_PIN3_INIT: 1;
    en_PinVal_t PORTB_PIN4_INIT: 1;
    en_PinVal_t PORTB_PIN5_INIT: 1;
    en_PinVal_t PORTB_PIN6_INIT: 1;
    en_PinVal_t PORTB_PIN7_INIT: 1;
}st_PortBInitVal_t;

typedef struct
{
    en_PinVal_t PORTC_PIN0_INIT: 1;
    en_PinVal_t PORTC_PIN1_INIT: 1;
    en_PinVal_t PORTC_PIN2_INIT: 1;
    en_PinVal_t PORTC_PIN3_INIT: 1;
    en_PinVal_t PORTC_PIN4_INIT: 1;
    en_PinVal_t PORTC_PIN5_INIT: 1;
    en_PinVal_t PORTC_PIN6_INIT: 1;
    en_PinVal_t PORTC_PIN7_INIT: 1;
}st_PortCInitVal_t;

typedef struct
{
    en_PinVal_t PORTD_PIN0_INIT: 1;
    en_PinVal_t PORTD_PIN1_INIT: 1;
    en_PinVal_t PORTD_PIN2_INIT: 1;
    en_PinVal_t PORTD_PIN3_INIT: 1;
    en_PinVal_t PORTD_PIN4_INIT: 1;
    en_PinVal_t PORTD_PIN5_INIT: 1;
    en_PinVal_t PORTD_PIN6_INIT: 1;
    en_PinVal_t PORTD_PIN7_INIT: 1;
}st_PortDInitVal_t;
```

- **EXTI**

```
typedef enum
{
        EXTI_ENABLE ,
        EXTI_DISABLE
}en_EXTI_State_t;

typedef enum
{
        EXTI0,
        EXTI1,
        EXTI2
}en_EXTI_Num_t;
```

```
typedef enum
{
        LOW_LEVEL,
        ON_CHANGE,
        FALLING_EDGE,
        RISING_EDGE
}en_EXTI_SenseMode_t;

typedef struct
{
        en_EXTI_Num_t         EXTI_NUM;
        en_EXTI_SenseMode_t SENSE_MODE;
        en_EXTI_State_t       EXTI_EN;
}st_PortDDirConfig_t;
```

- **Timer**

```
typedef enum
{
        NORMAL_MODE,
        PWM_MODE,
        CTC_MODE,
        FAST_PWM_MODE
}en_TIMMode_t;
```

```
typedef enum
{
        TIM_NO_CLOCK                ,
        TIM_DIV_BY_1                ,
        TIM_DIV_BY_8                ,
        TIM_DIV_BY_64               ,
        TIM_DIV_BY_256              ,
        TIM_DIV_BY_1024             ,
        TIM_EXTERNAL_FALLING_EDGE,
        TIM_EXTERNAL_RISING_EDGE
}en_TIM_CLK_SELECT_t;
```

## B. HAL

- **Motor**

```
typedef struct
{
        Uint8_t MOTOR_PORT;
        Uint8_t MOTOR_PIN1;
        Uint8_t MOTOR_PIN2;
}st_MotorConfig_t;
```

- **Button**

(For Pre-Compile & Linking Configurations)

```
typedef enum
{
        PULL_UP,
        PULL_DOWN
}en_Pull_t;
```

(For Linking Configuration)

```
typedef struct
{
        Uint8_t   BUTTON_PORT;
        Uint8_t   BUTTON_PIN;
        Uint8_t   BUTTON_EXTI_NUM;
        en_Pull_t BUTTON_PULL_TYPE;
}st_ButtonConfig_t;
```

- **Keypad**

```
typedef struct
{
	Uint8_t KEYPAD_ROW_PORT;

	Uint8_t KEYPAD_COL_PORT;
	Uint8_t KEYPAD_ROW_NUM;
	Uint8_t KEYPAD_COL_NUM;

	Uint8_t KEYPAD_ROW_START_PIN;
	Uint8_t KEYPAD_ROW_END_PIN;
	Uint8_t KEYPAD_COL_START_PIN;
	Uint8_t KEYPAD_COL_END_PIN;

}st_KeypadConfig_t;
```

- **LCD**

(For Pre-Compile & Linking Configurations)

```
typedef enum
{
	LCD_4BIT_MODE,
	LCD_8BIT_MODE
}en_LCDmode_t;

typedef enum
{
	ONE_LINE_MODE,
	TWO_LINE_MODE
}en_LCDlineMode_t;

typedef enum
{
	LCD_5_BY_7,
	LCD_5_BY_10
}en_LCDfont_t;
```

(For Linking Configuration)

```
typedef struct
{
	Uint8_t          LCD_DATA_PORT;
	Uint8_t          LCD_CONTROL_PORT;
	Uint8_t          LCD_RS_PIN;
	Uint8_t          LCD_RW_PIN;
	Uint8_t          LCD_EN_PIN;
	en_LCDmode_t     LCD_MODE;
	en_LCDlineMode_t LCD_LINE_MODE;
	en_LCDfont_t     LCD_FONT;
}st_LCDconfig_t;
```

- **Ultrasonic**

```
typedef struct
{
	Uint8_t US_PORT;
	Uint8_t US_TRIG_PIN;
	Uint8_t US_ECHO_PIN;
}st_USconfig_t;
```

*Note:*

*No pre-compile or linking cofigurations in the control or in application layers as all configurations here are done in lower layers.*