

# Obstacle Avoiding Car System Design

By : Team 4

## Contents

<b>Flowchart Table</b> .....	3
<b>Detailed Requirements</b> .....	5
System Requirements: .....	5
<b>Layered architecture</b> .....	6
<b>System moduls</b> .....	6
<b>MCAL</b> .....	7
DIO API :.....	7
<i>External Interrupt API</i> :.....	10
TIMER API :.....	12
<b>HAL APIs</b> .....	16
ULTRASONIC API: .....	16
LCD API:.....	19
Motor API:.....	22
Car Control API :.....	26
TIMING API :.....	32
Interrupt Manager API :.....	38
PWM API: .....	40
Button API: .....	45
<b>App</b> .....	48
App API:.....	48

## Flowchart Table

Figure 1 Desgin_arch .....	6
Figure 2 Obstical_Avoiding_Car_Design_Modules .....	6
Figure 3 DIO_init() .....	7
Figure 4 DIO_write_pin() .....	8
Figure 5 DIO_read_pin().....	9
Figure 6 ext_interrupt_int() .....	10
Figure 7 ext_interrupt_enable() .....	10
Figure 8 ext_interrupt_disable() .....	11
Figure 9 ext_interrupt_set_callback_init() .....	11
Figure 10 timer_x_initialization() .....	12
Figure 11 timer_x_start() .....	13
Figure 12 timer_x_set_tcnt() .....	14
Figure 13 timer_x_initialize_callback_OVF() .....	15
Figure 14 timer_x_initialize_callback_COMP() .....	15
Figure 15 HULTRASONIC_vidSigCalc() .....	16
Figure 16 HULTRASONIC_vidTrigger() .....	17
Figure 17 HULTRASONIC_vidInit() .....	17
Figure 18 HULTRASONIC_vidInterruptEnable() .....	18
Figure 20 HULTRASONIC_vidInterruptDisable() .....	18
Figure 19 HULTRASONIC_u8Read() .....	18
Figure 20 LCD_clear() .....	19
Figure 21 LCD_init () .....	19
Figure 22 LCD_writeString() .....	20
Figure 23 LCD_setCursor() .....	20
Figure 24 LCD_writeSpChar() .....	21
Figure 21 MOTOR_INIT () .....	22
Figure 22 MOTOR_FORWARD() .....	23
Figure 23 MOTOR_BACKWARD() .....	24
Figure 24 MOTOR_STOP() .....	25
Figure 25 CAR_INIT() .....	26
Figure 26 CAR_FORWARD() .....	27
Figure 27 CAR_BACKWARD() .....	28
Figure 28 CAR_REVERSE_RIGHT() .....	29
Figure 29 CAR_REVERSE_LEFT() .....	30
Figure 30 CAR_STOP .....	31
Figure 31 timing_init () .....	32
Figure 32 timing_star() .....	33
Figure 33 timing_stop .....	33
Figure 34 timing_init_1() .....	34
Figure 35 timing_start_1() .....	35
Figure 37 timing_stop_1() .....	35
Figure 36 timing_time_out() .....	35
Figure 38 timing_break_time_out() .....	36
Figure 39 delay_s() .....	36

Figure 40 timing_init_2()	36
Figure 41 timing_start_2()	37
Figure 42 timing_stop_2()	37
Figure 43 timing_get_ticks_2()	37
Figure 44 extim_init()	38
Figure 45 extim_enable()	39
Figure 46 extim_disable()	39
Figure 47 pwm_init	40
Figure 49 pwm_start_tick()	41
Figure 48 pwm_start()	41
Figure 50 pwm_checking()	42
Figure 51 pwm_change_frequency_or_duty_cycle()	42
Figure 52 pwm_stop()	43
Figure 53 pwm_end_tick()	43
Figure 54 BTN_init()	45
Figure 55 BTN_get_state()	47
Figure 56 APP_init()	48
Figure 57 APP_start() added besides the pdf as SVG	48
Figure 58 APP_make_decision()	49
Figure 59 APP_updateDirection()	49
Figure 60 intToString()	50
Figure 61 BUTTON_vidChangeState()	51

# Detailed Requirements

## System Requirements:

1. The car starts initially from 0 speed
2. The default rotation direction is to the right
3. Press PB2 to start or stop the robot
4. After Pressing Start:
  - 1) The LCD will display a centered message in line 1 "Set Def. Rot."
  - 2) The LCD will display the selected option in line 2 "Right"
  - 3) The robot will wait for 5 seconds to choose between Right and Left
    - (1) When PB1 is pressed once, the default rotation will be Left and the LCD line 2 will be updated
    - (2) When PB1 is pressed again, the default rotation will be Right and the LCD line 2 will be updated
    - (3) For each press, the default rotation will be changed and the LCD line 2 is updated
    - (4) After the 5 seconds, the default value of rotation is set
  - 4) The robot will move after 2 seconds from setting the default direction of rotation.
- 5) For No obstacles or object is far than 70 centimeters:
  - (1) The robot will move forward with 30% speed for 5 seconds
  - (2) After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance
  - (3) The LCD will display the speed and moving direction in line 1: "Speed:00% Dir: F/B/R/S", F: forward, B: Backwards, R: Rotating, and S: Stopped
  - (4) The LCD will display Object distance in line 2 "Dist.: 000 Cm"
- 6) For Obstacles located between 30 and 70 centimeters
  - (1) The robot will decrease its speed to 30%
  - (2) LCD data is updated
- 7) For Obstacles located between 20 and 30 centimeters
  - (1) The robot will stop and rotates 90 degrees to right/left according to the chosen configuration
  - (2) The LCD data is updated
- 8) For Obstacles located less than 20 centimeters
  - (1) The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30
  - (2) The LCD data is updated
  - (3) Then preform point 8
- 9) Obstacles surrounding the robot (Bonus)
  - (1) If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop
  - (2) LCD data will be updated.
  - (3) The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

## Layered architecture

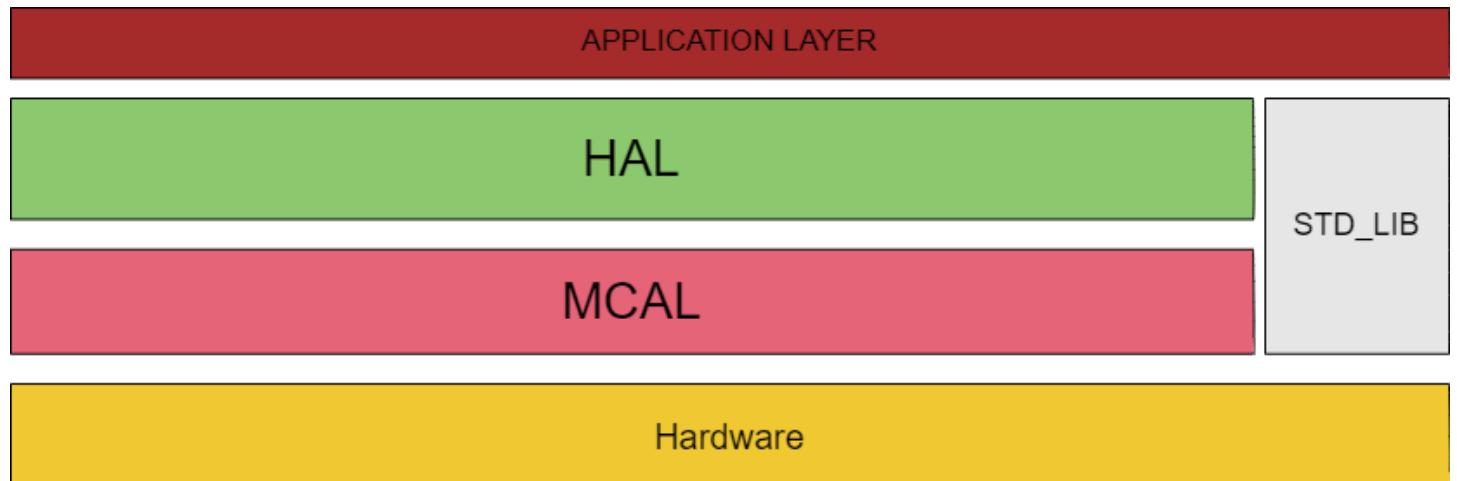


Figure 1 Desgin\_arch

## System modules

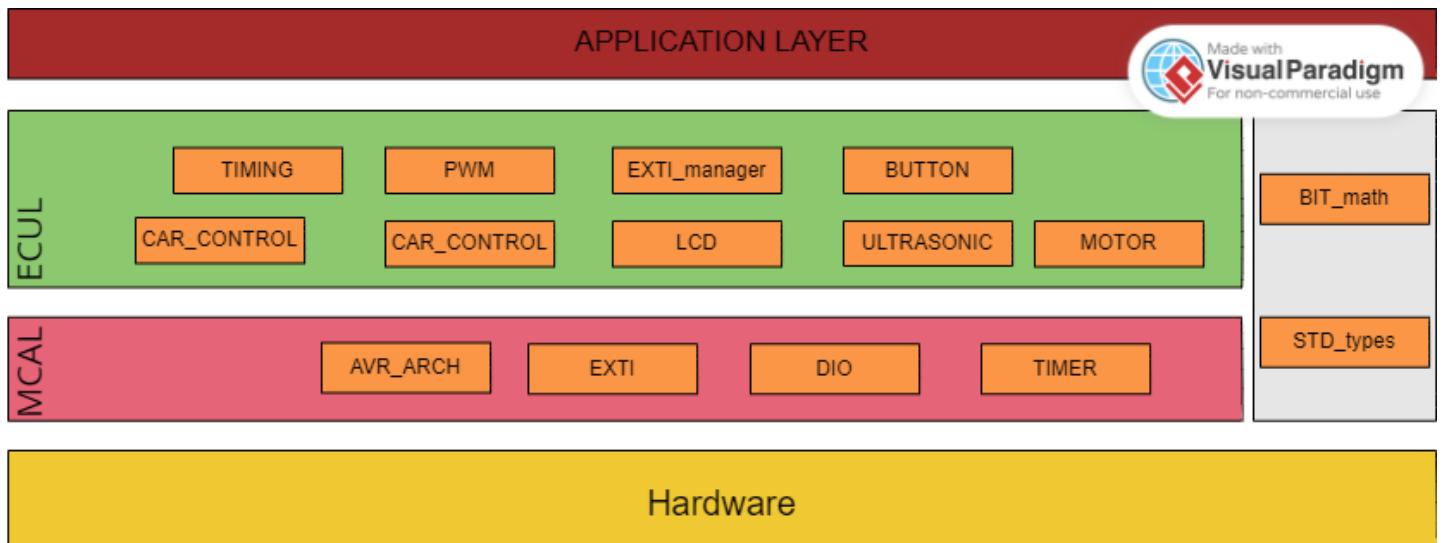


Figure 2 Obstical\_Avoiding\_Car\_Design\_Modules

# MCAL

## DIO API :

Flowcharts:

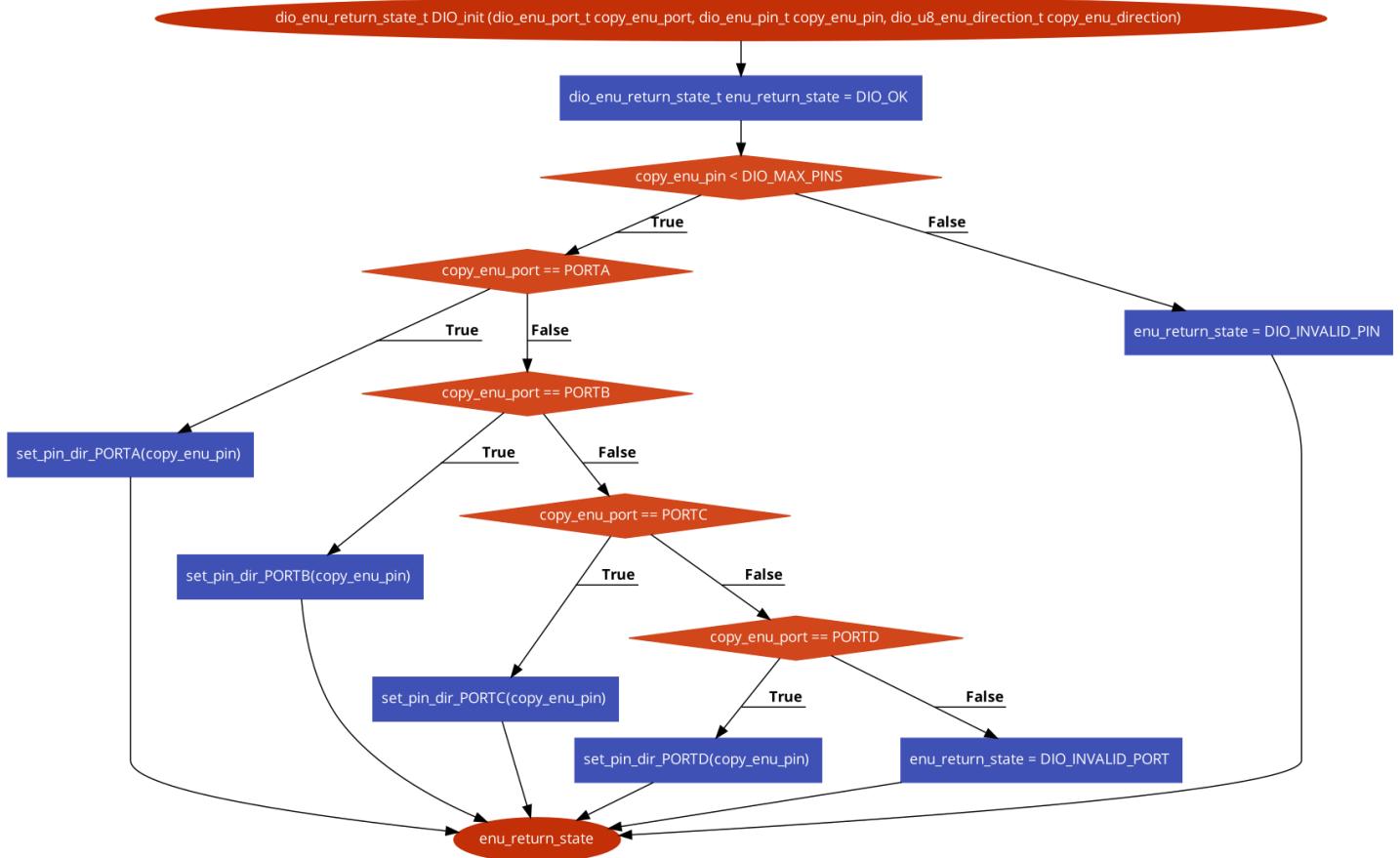


Figure 3 DIO\_init()

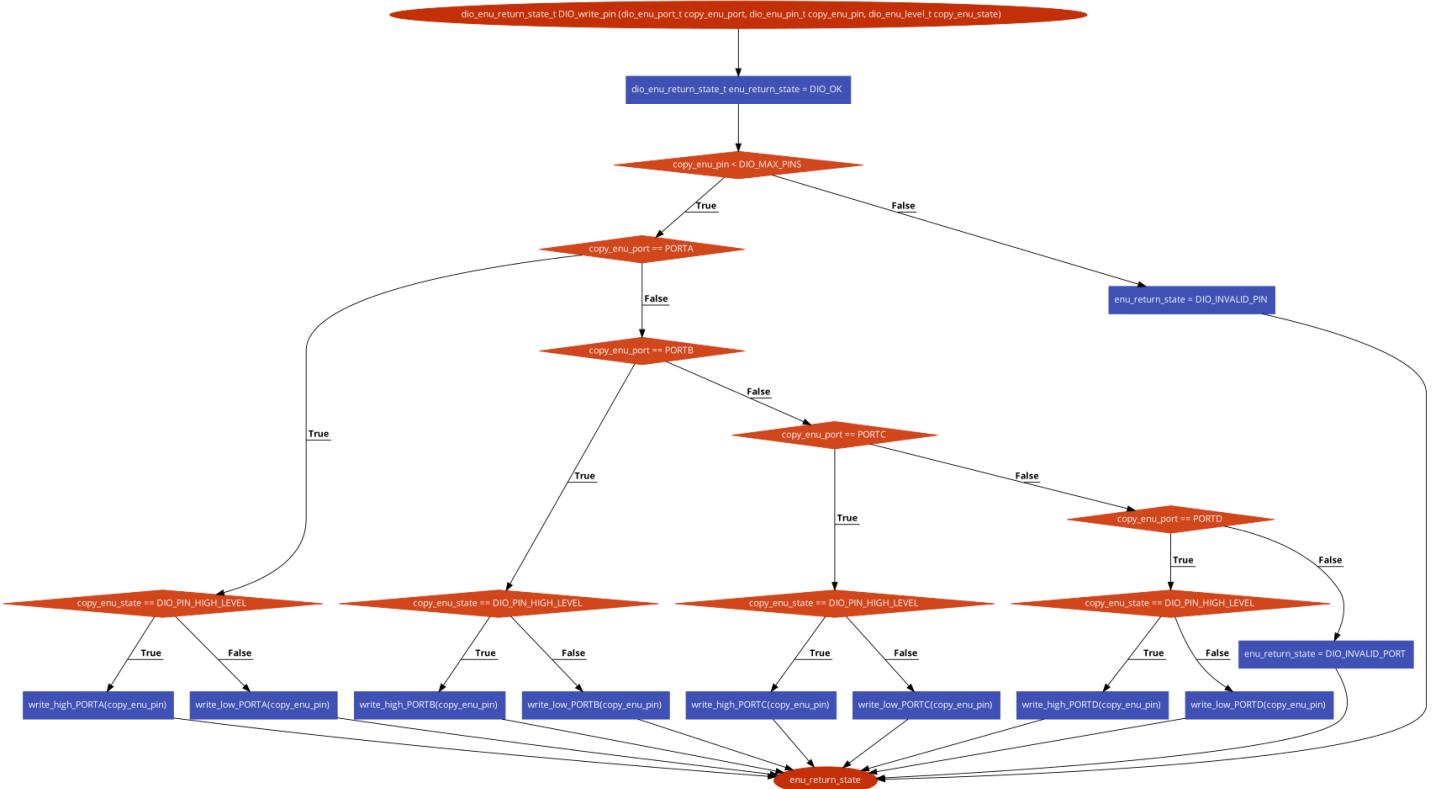


Figure 4 DIO\_write\_pin()

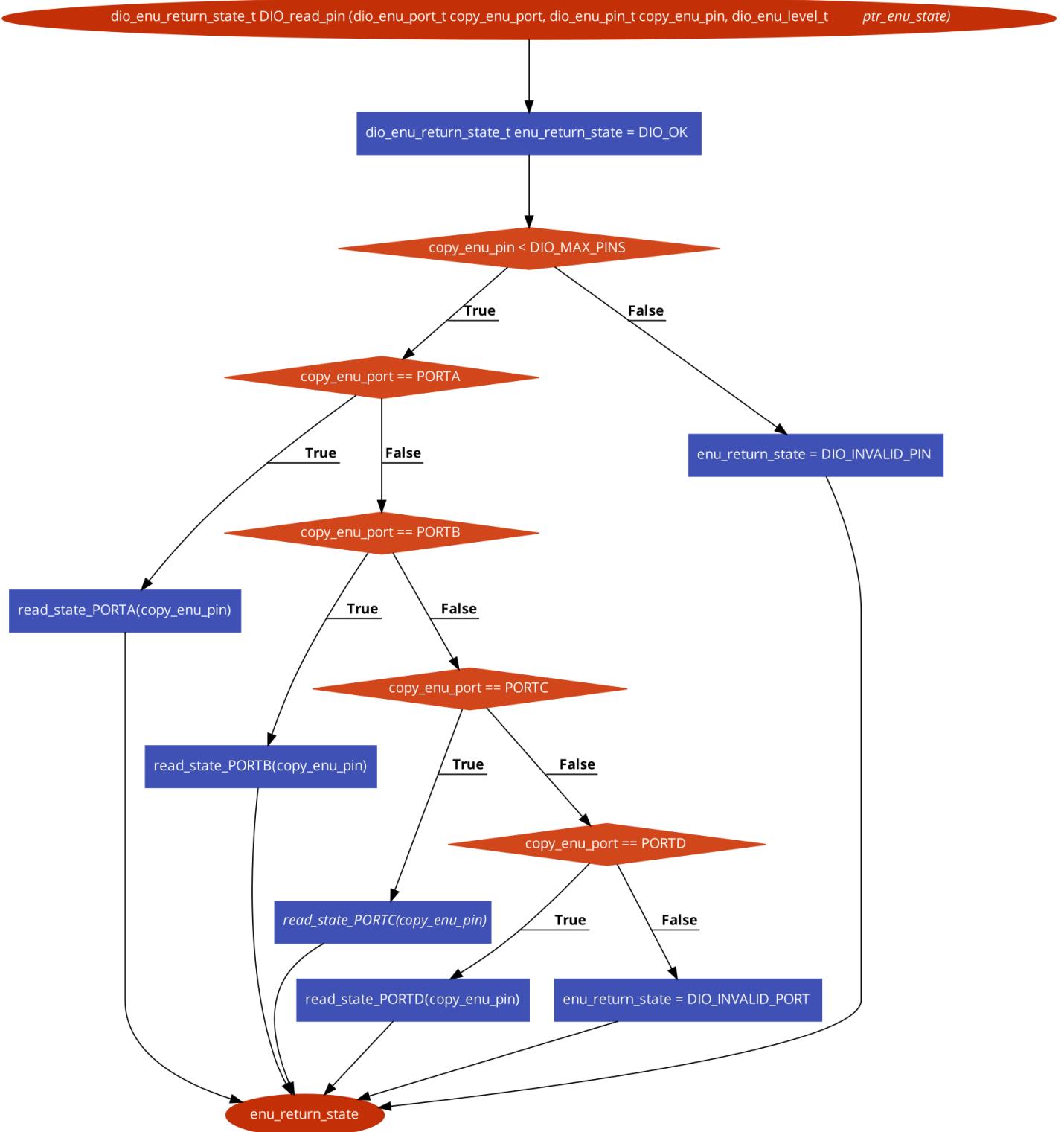


Figure 5 DIO\_read\_pin()

## *External Interrupt API :*

Flowcharts:

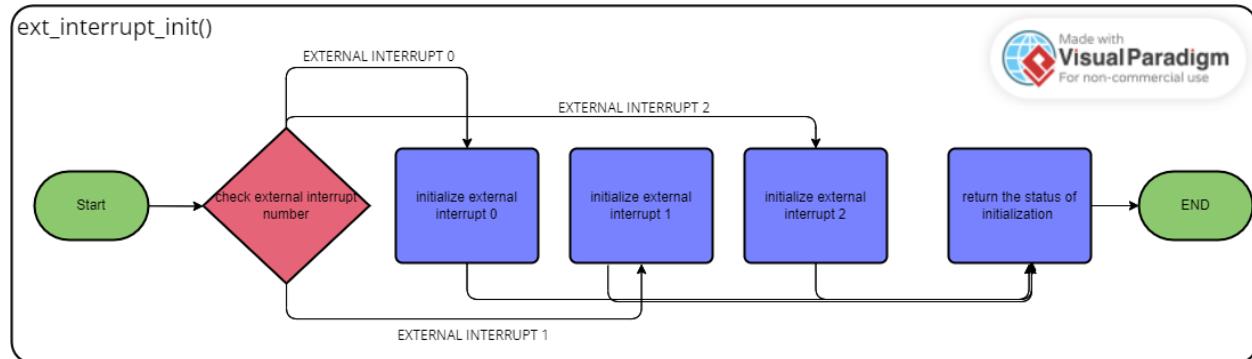


Figure 6 `ext_interrupt_int()`

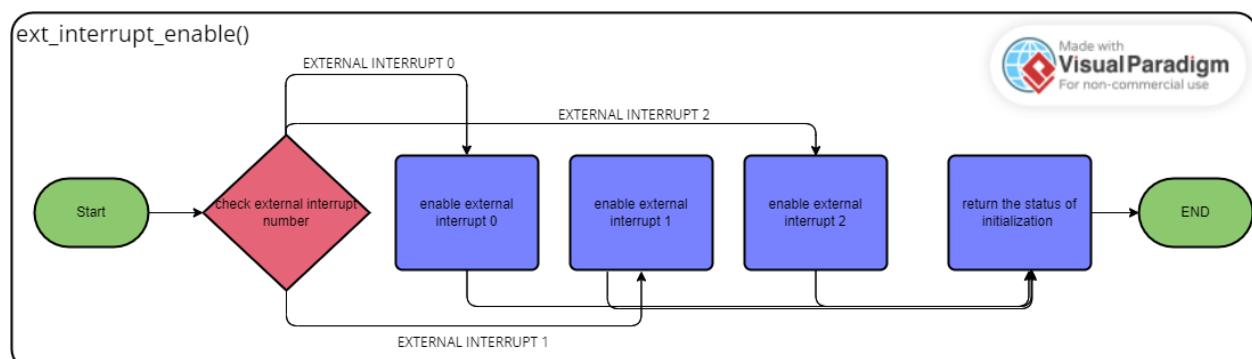


Figure 7 `ext_interrupt_enable()`

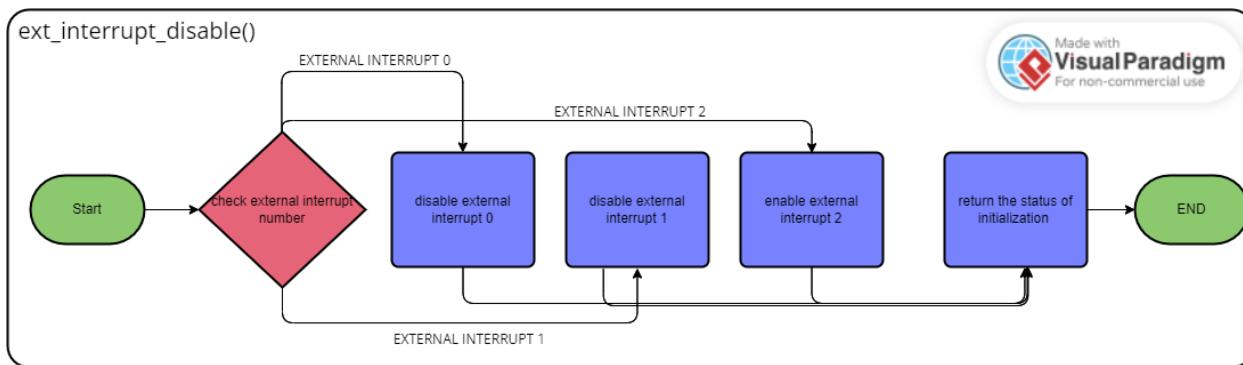


Figure 8 `ext_interrupt_disable()`

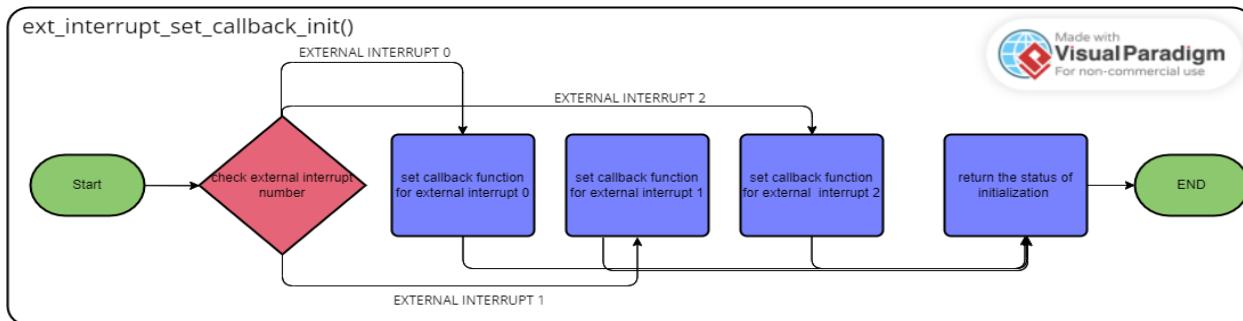


Figure 9 `ext_interrupt_set_callback_init()`

## TIMER API :

Flowcharts:

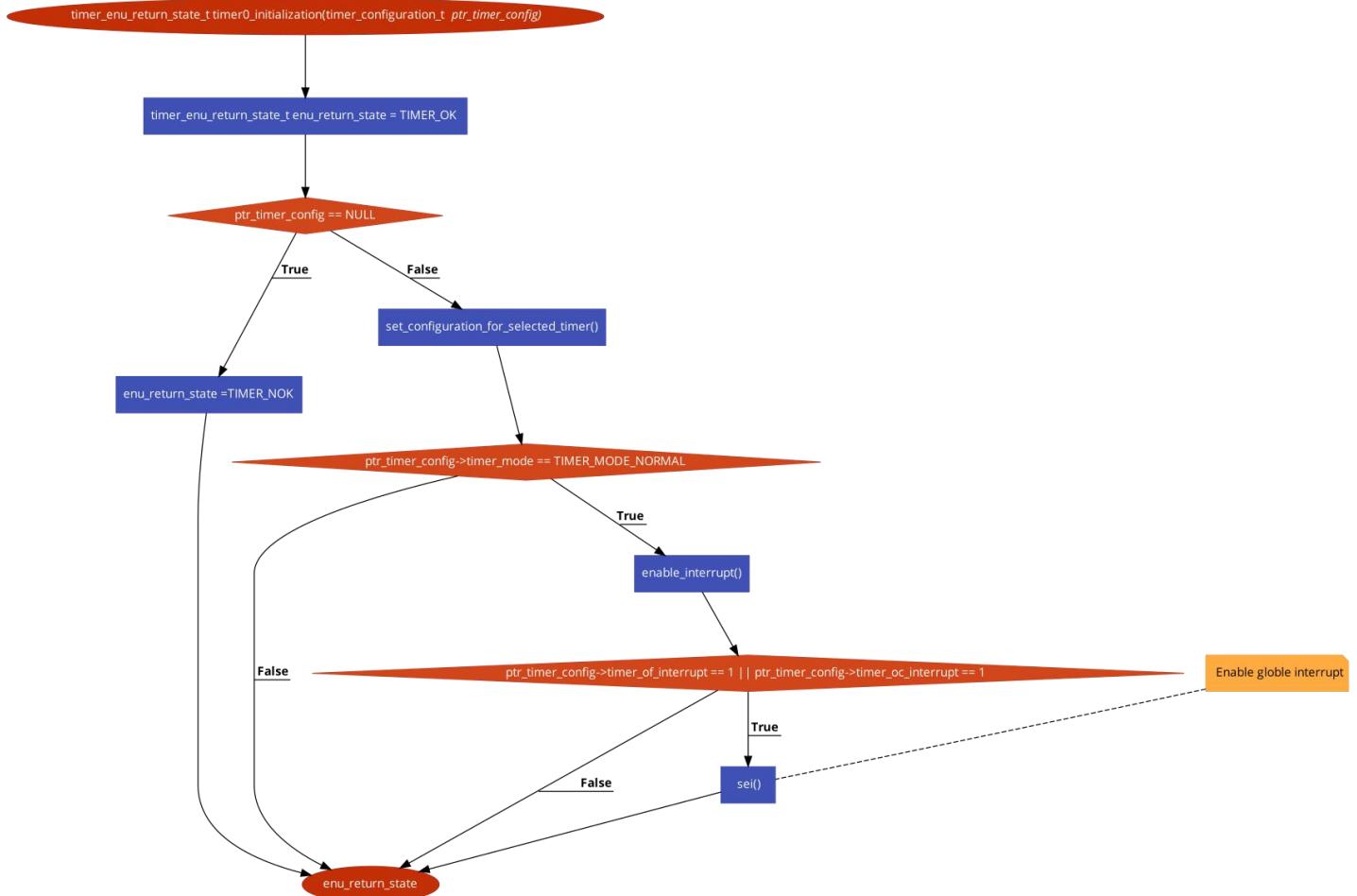


Figure 10 `timer_x_initialization()`

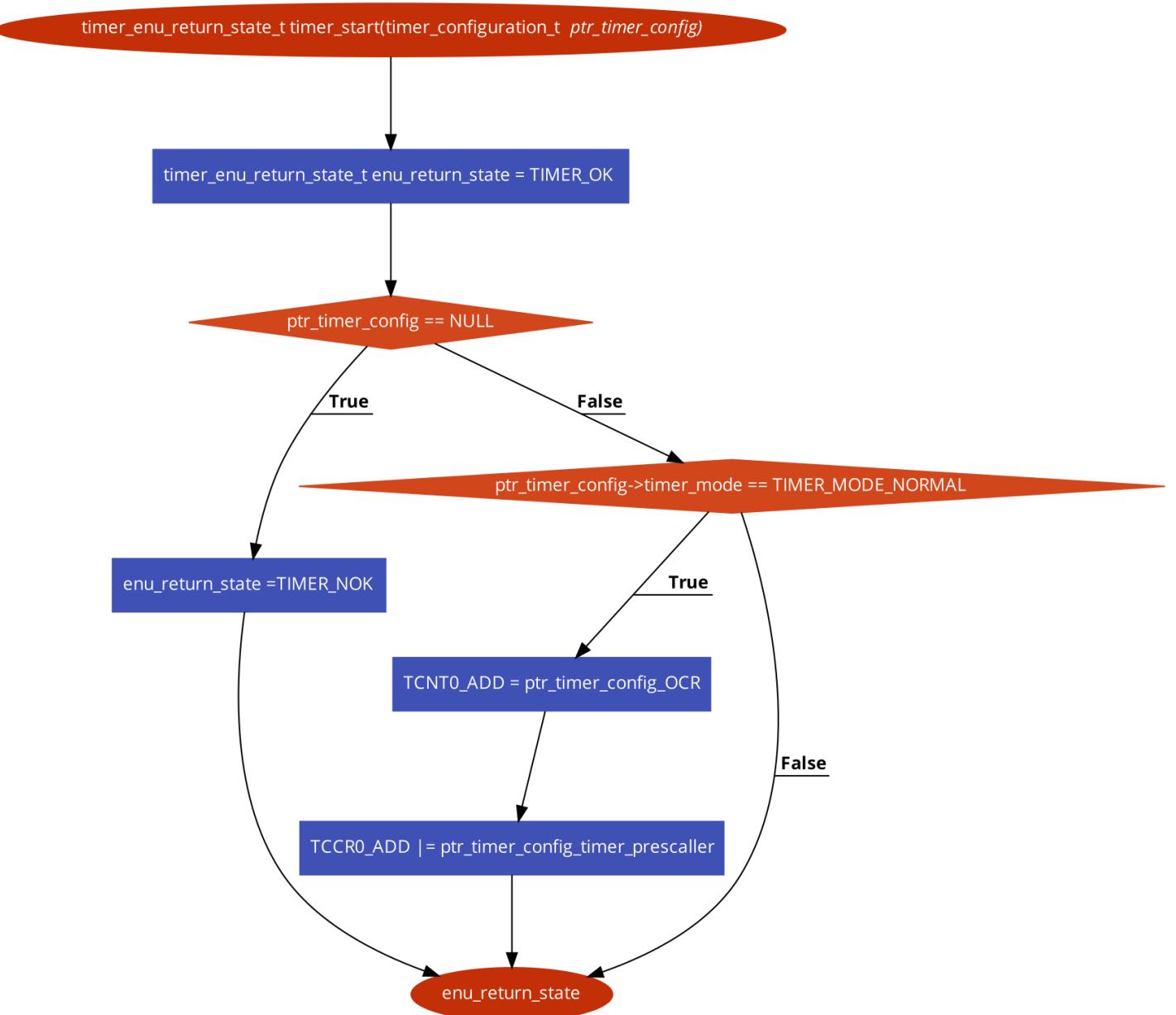


Figure 11 `timer_x_start()`

```
timer_enu_return_state_t timer_set_tcnt(timer_configuration_t *ptr_timer_config)
```

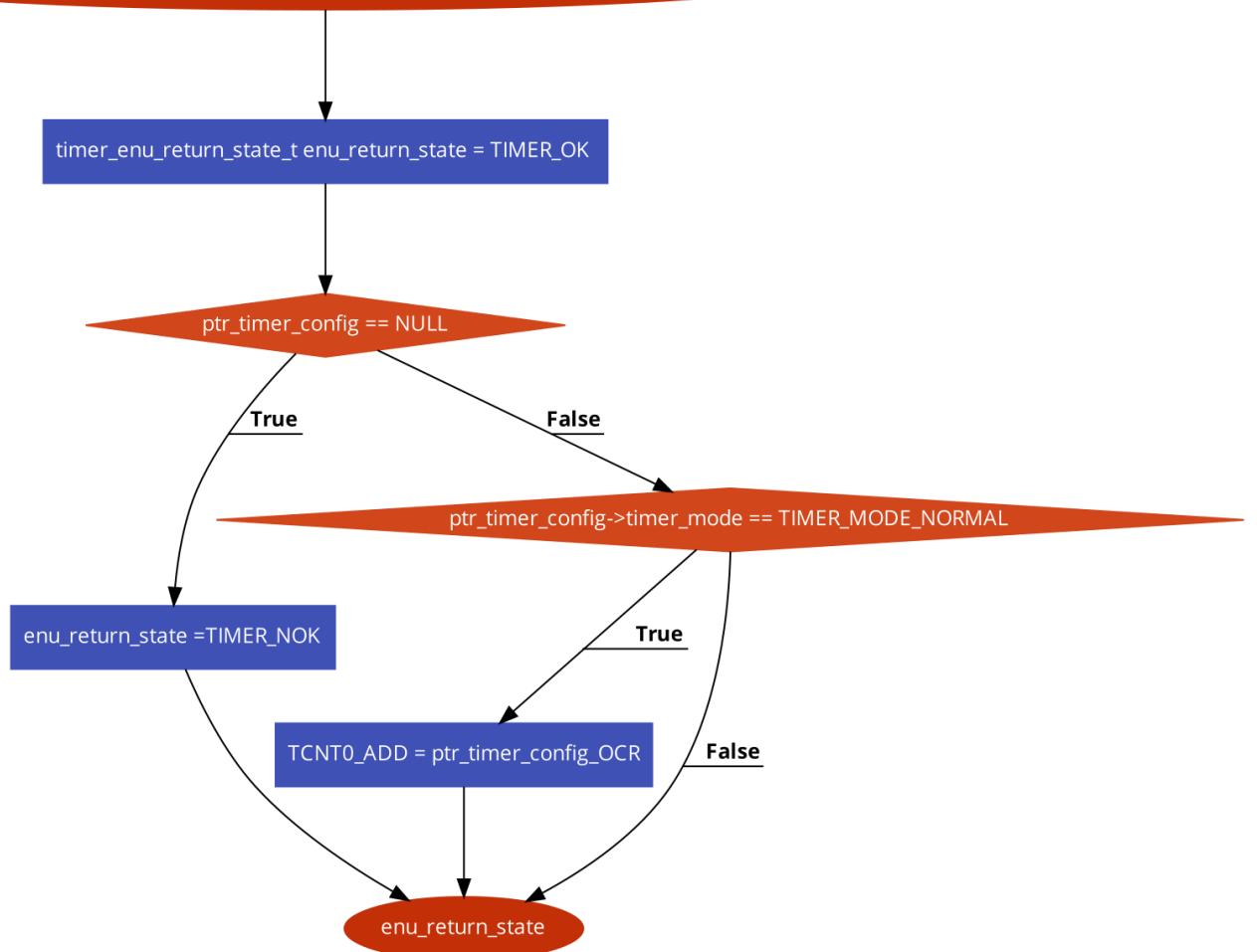


Figure 12 `timer_x_set_tcnt()`

timer\_enu\_return\_state\_t timer\_initialize\_callback\_OVF(void (ptr\_func)(void))

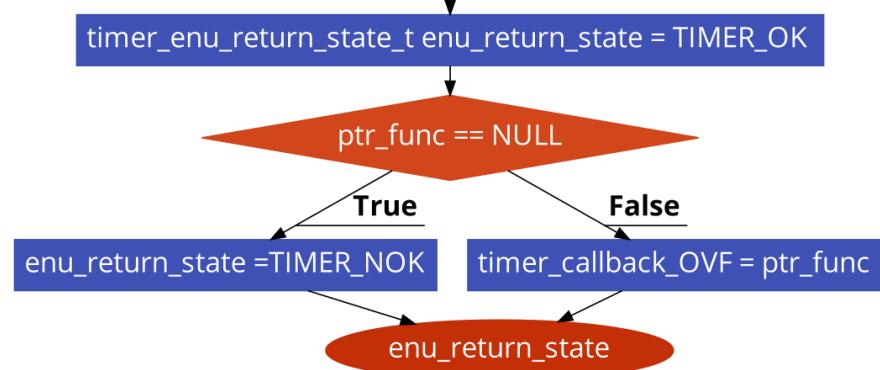


Figure 13 `timer_x_initialize_callback_OVF()`

timer\_enu\_return\_state\_t timer\_initialize\_callback\_COMP(void (ptr\_func)(void))

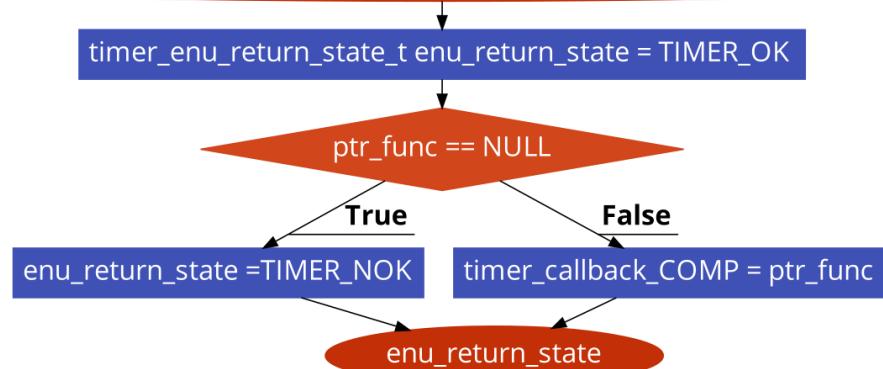


Figure 14 `timer_x_initialize_callback_COMP()`

# HAL

## ULTRASONIC API:

Flowcharts:

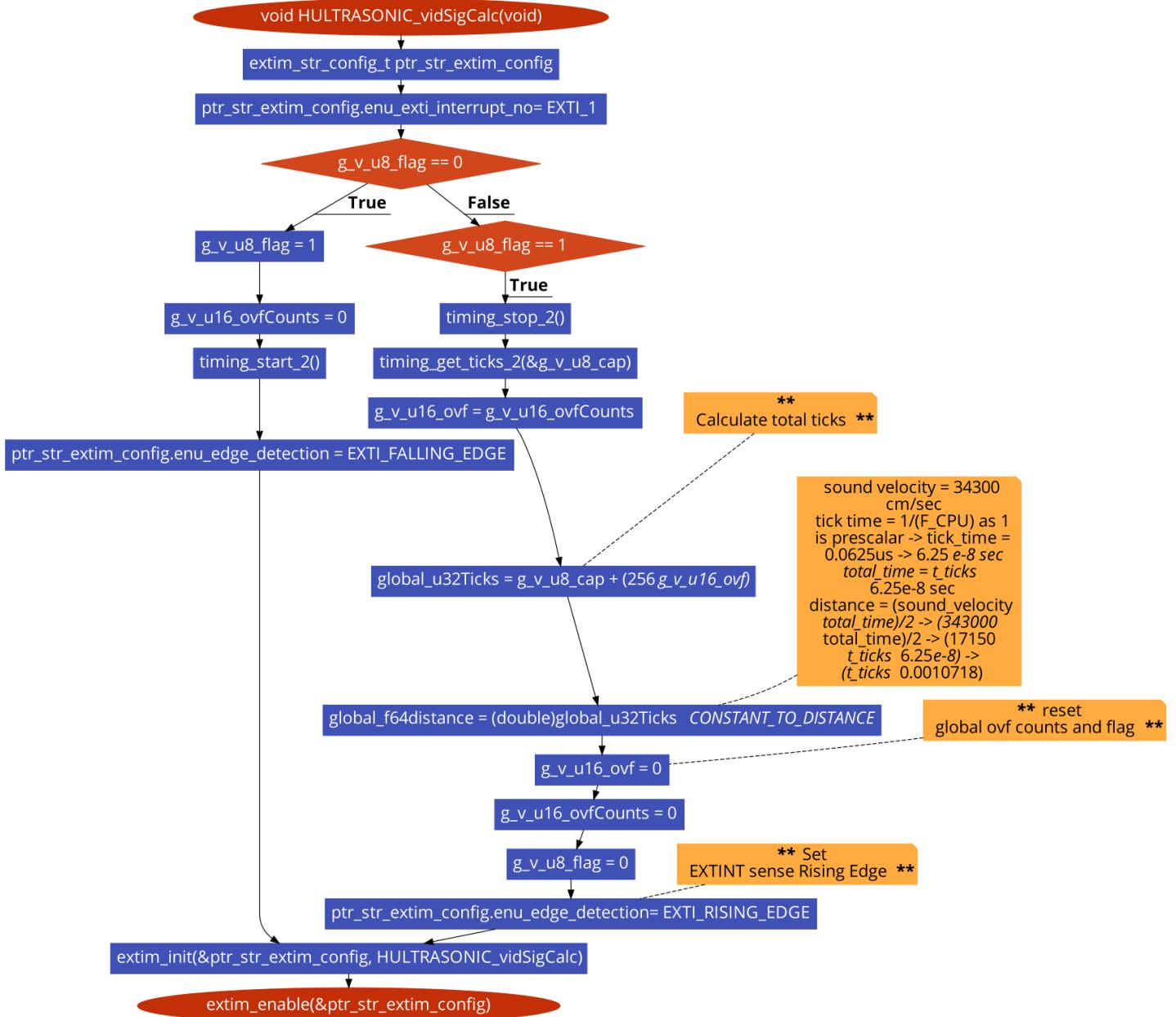


Figure 15 HULTRASONIC\_vidSigCalc()

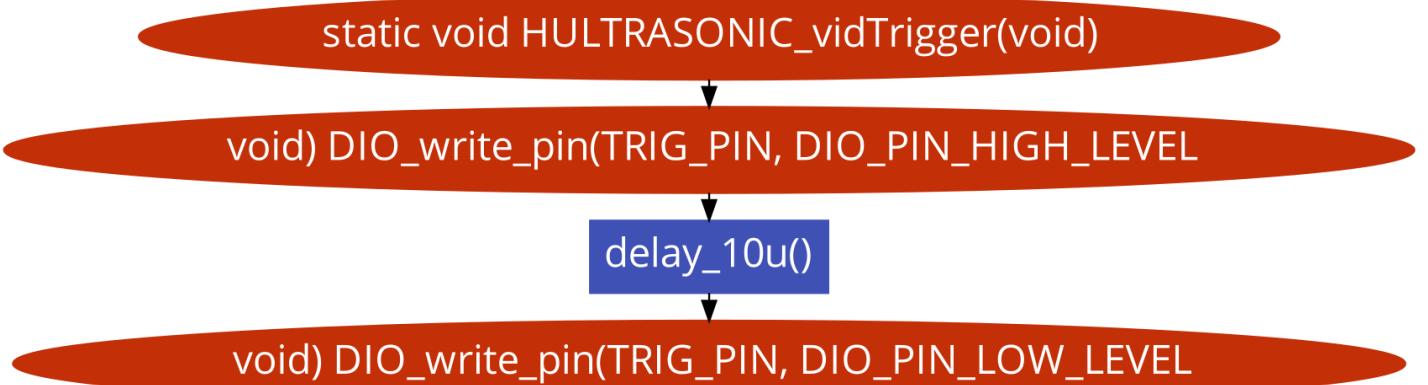


Figure 16 `HULTRASONIC_vidTrigger()`

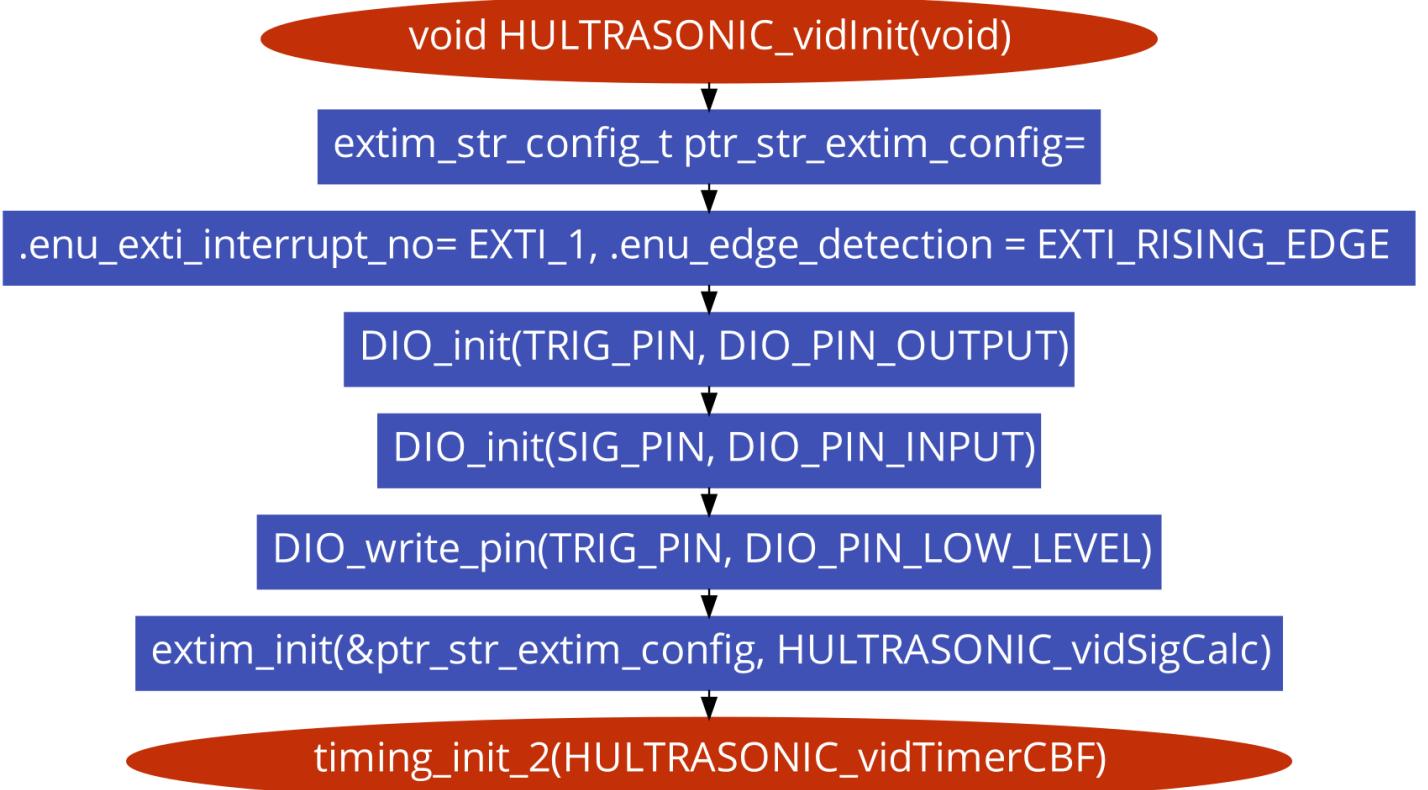


Figure 17 `HULTRASONIC_vidInit()`

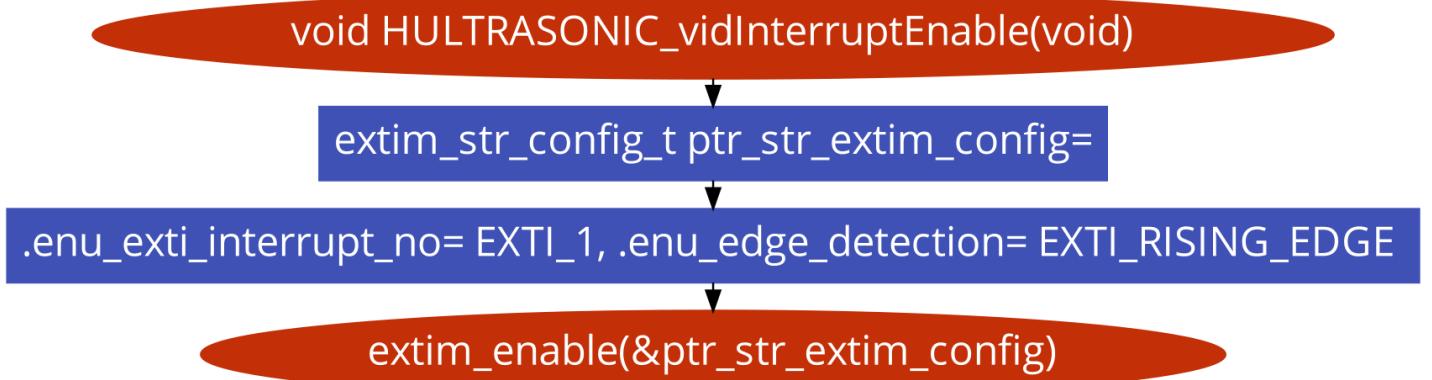


Figure 18 HULTRASONIC\_vidInterruptEnable()

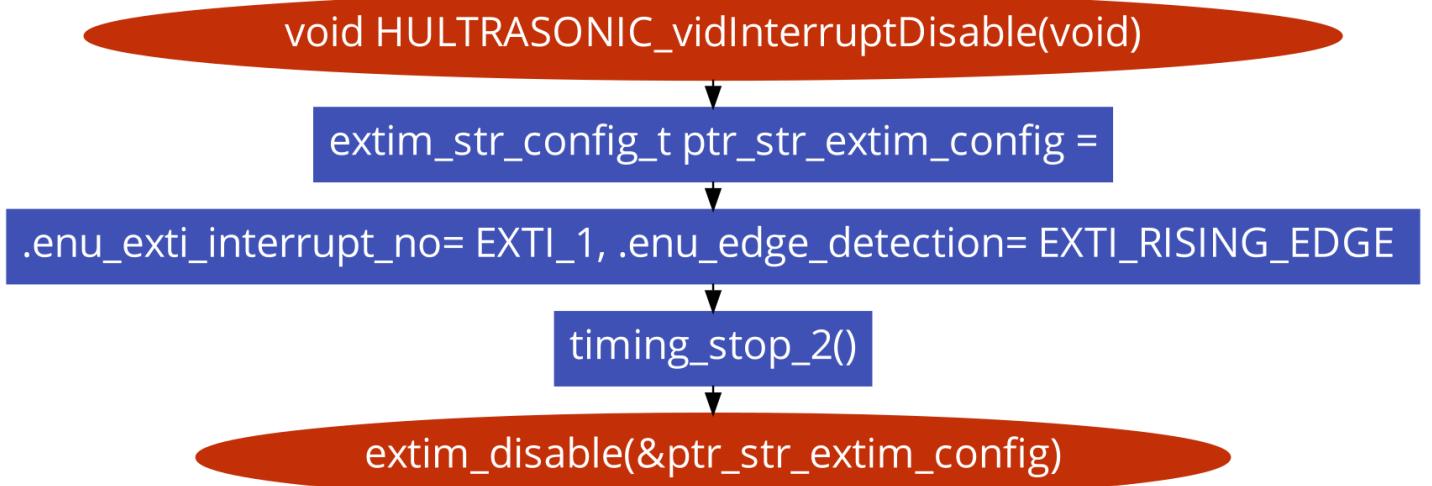


Figure 20 HULTRASONIC\_vidInterruptDisable()

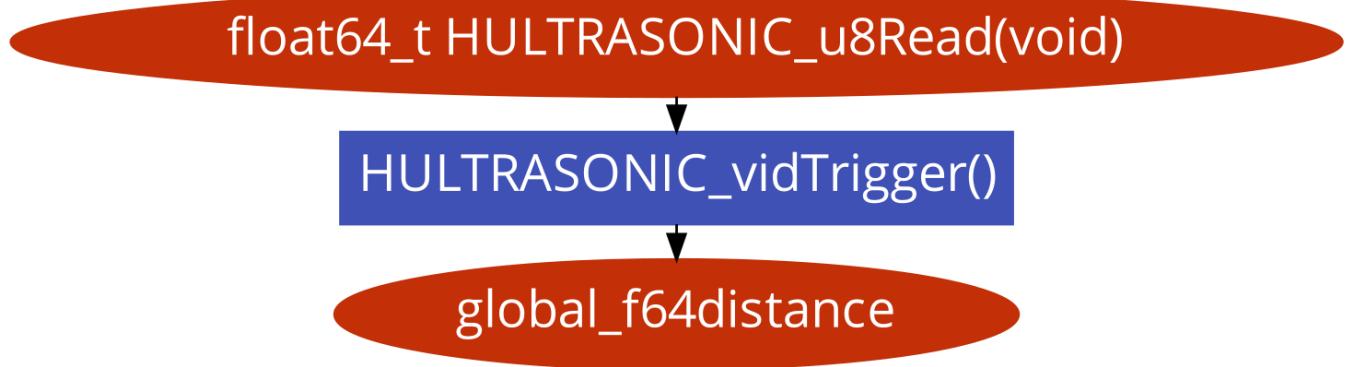


Figure 19 HULTRASONIC\_u8Read()

## LCD API:

Flowcharts:

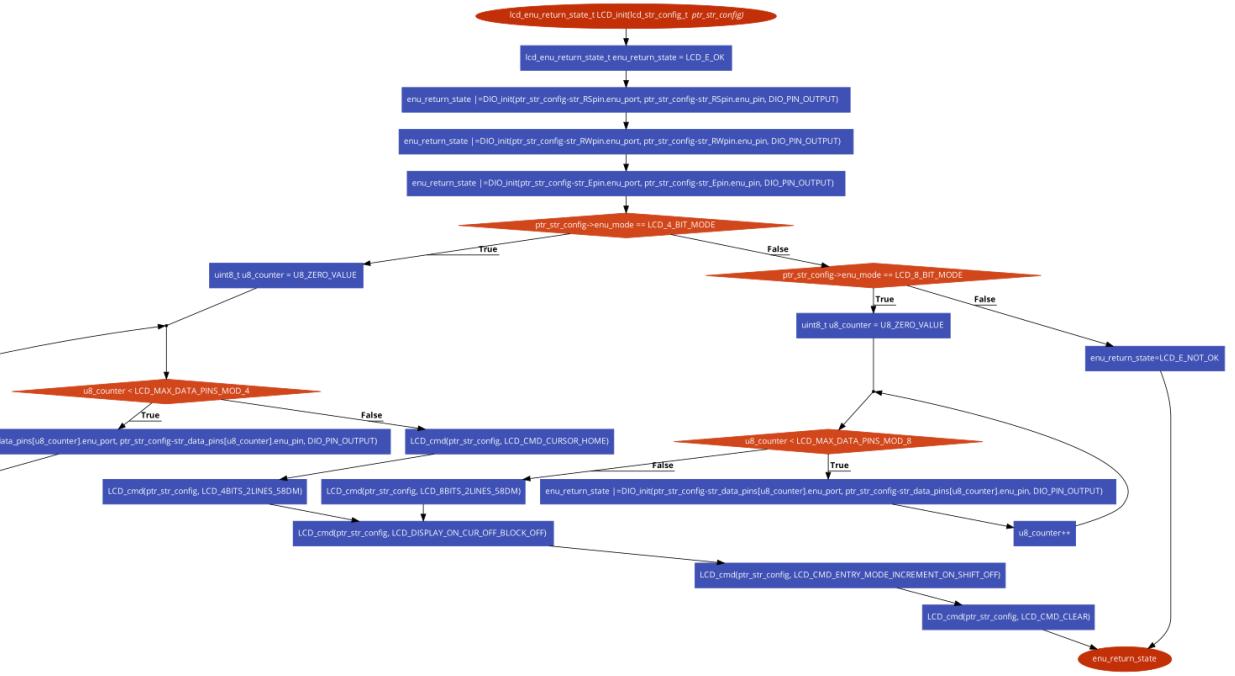


Figure 22 LCD\_init()

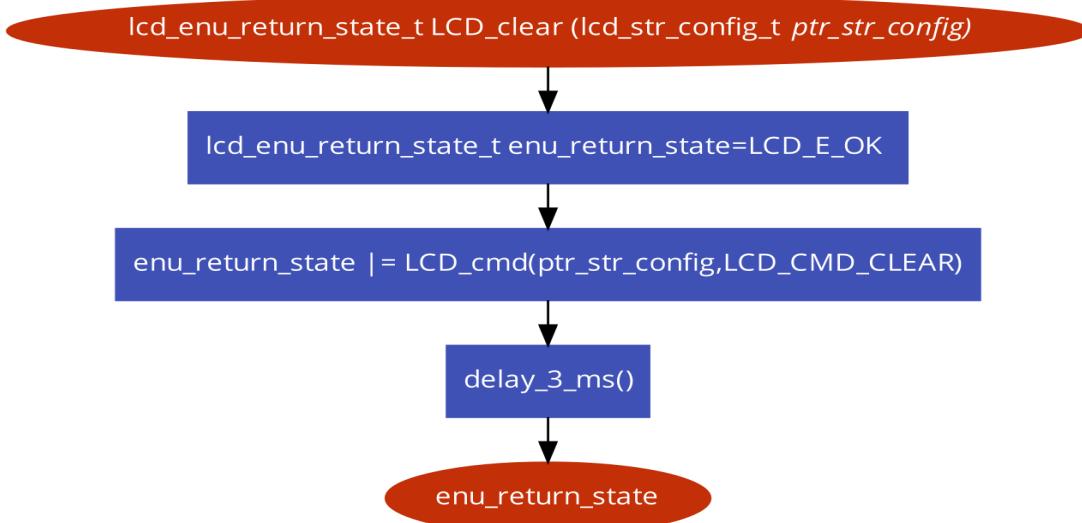


Figure 21 LCD\_clear()

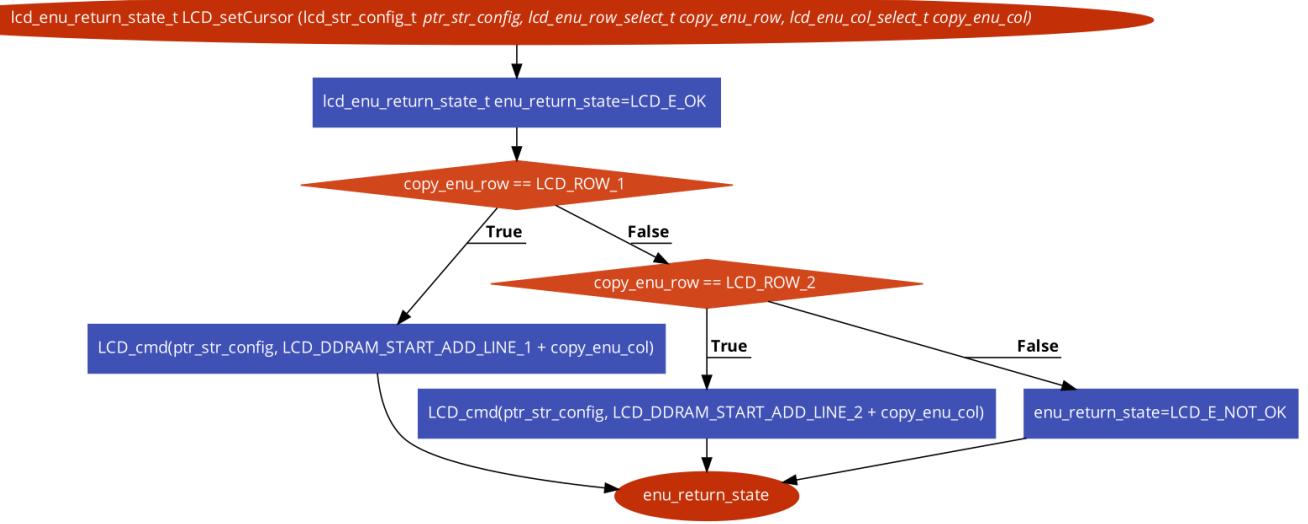


Figure 24 `LCD_setCursor()`

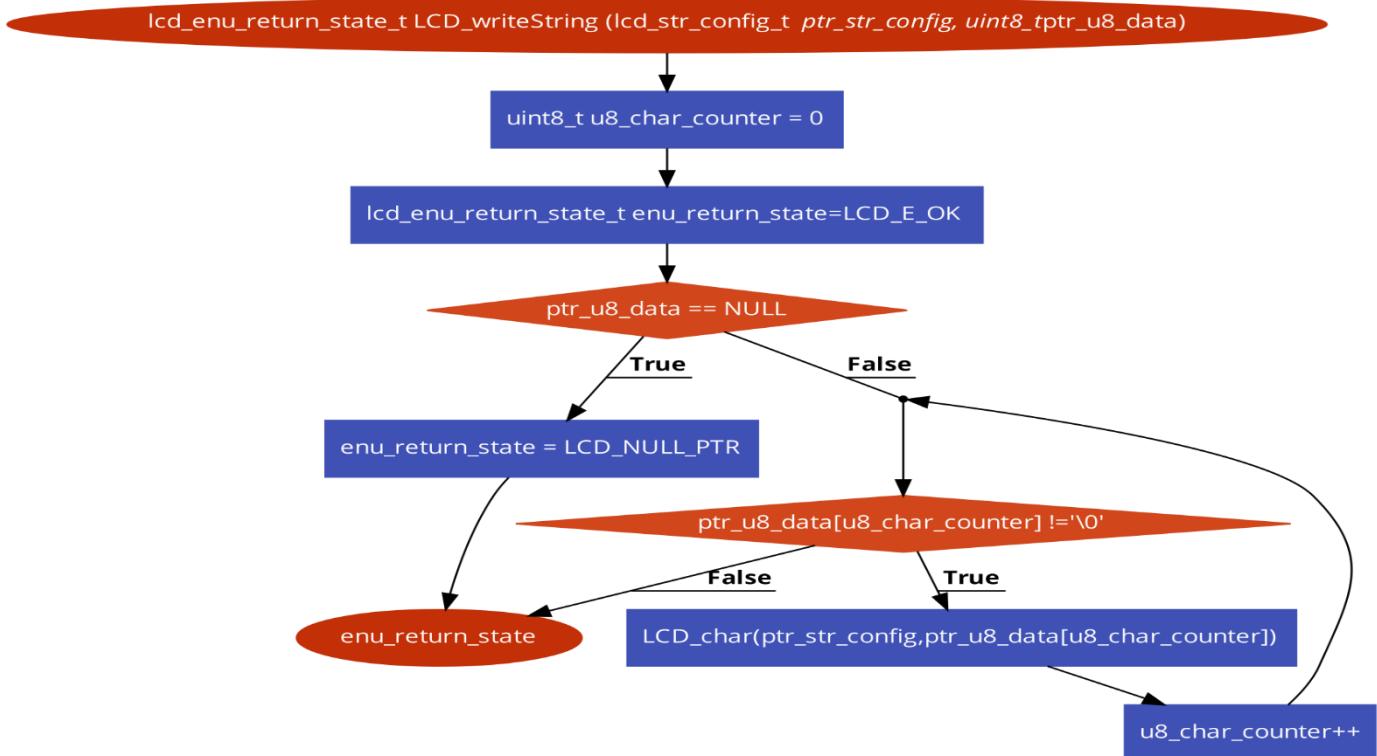


Figure 23 `LCD_writeString()`

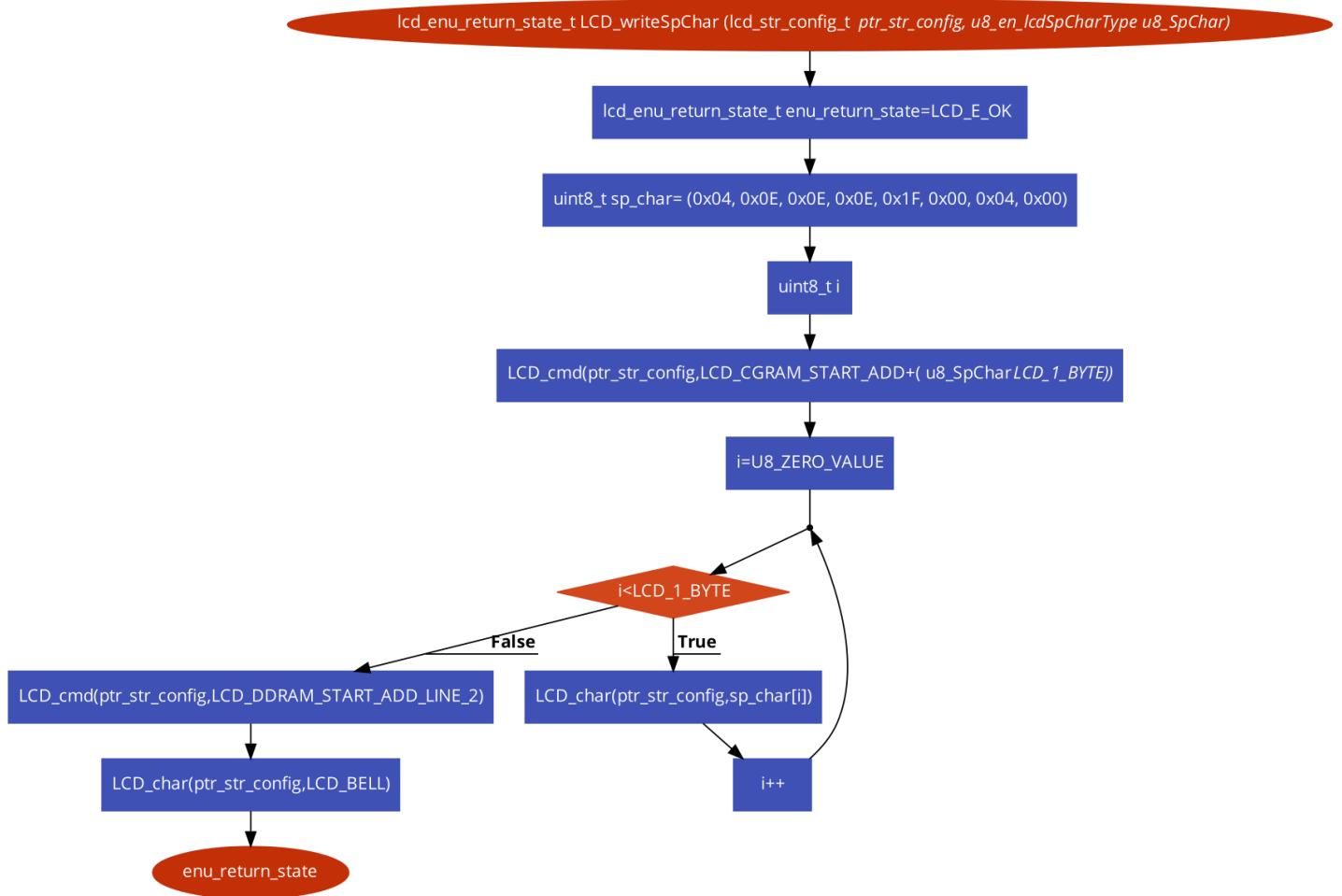


Figure 25 `LCD_writeSpChar()`

## Motor API:

Flowcharts:

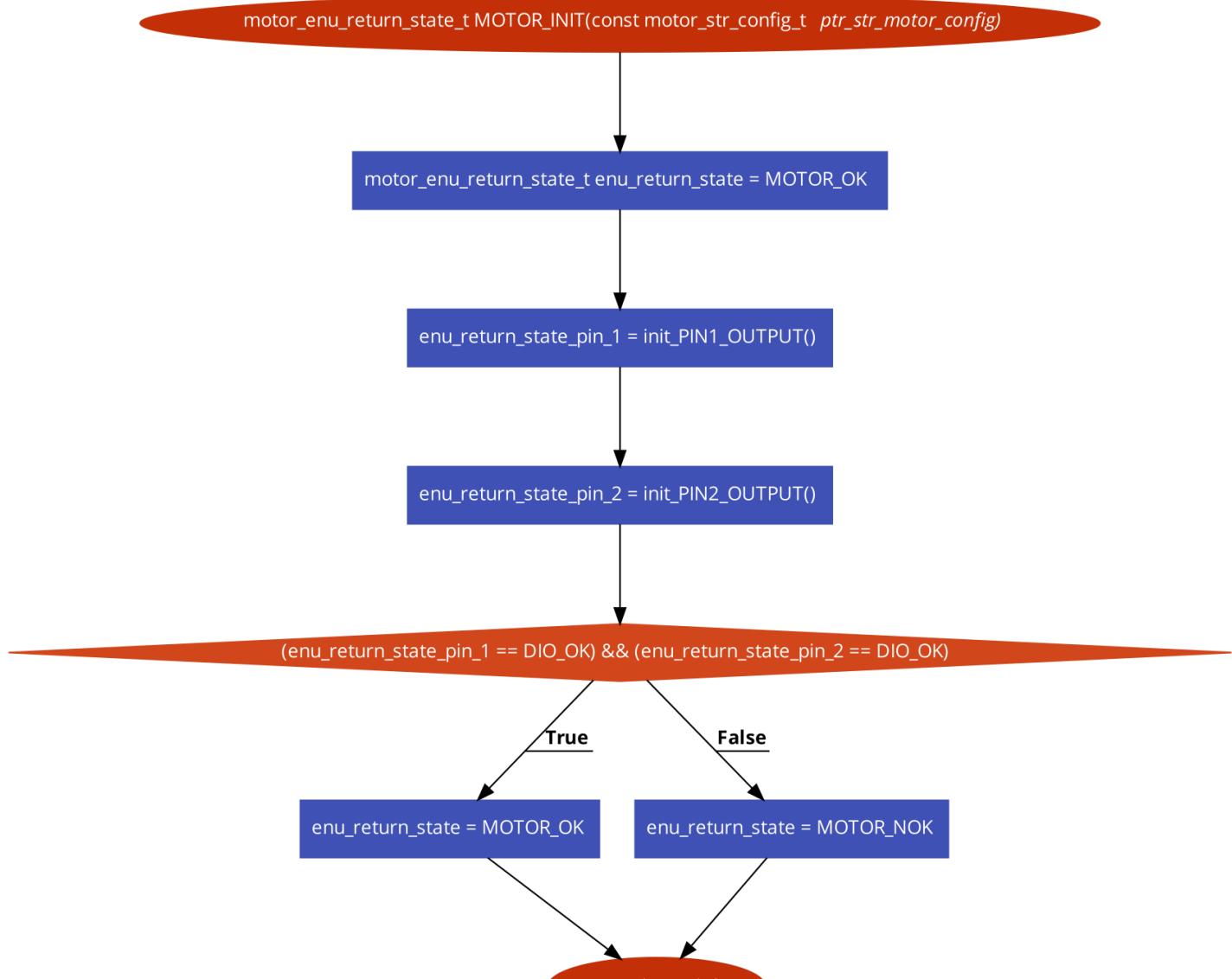


Figure 26 MOTOR\_INIT ()

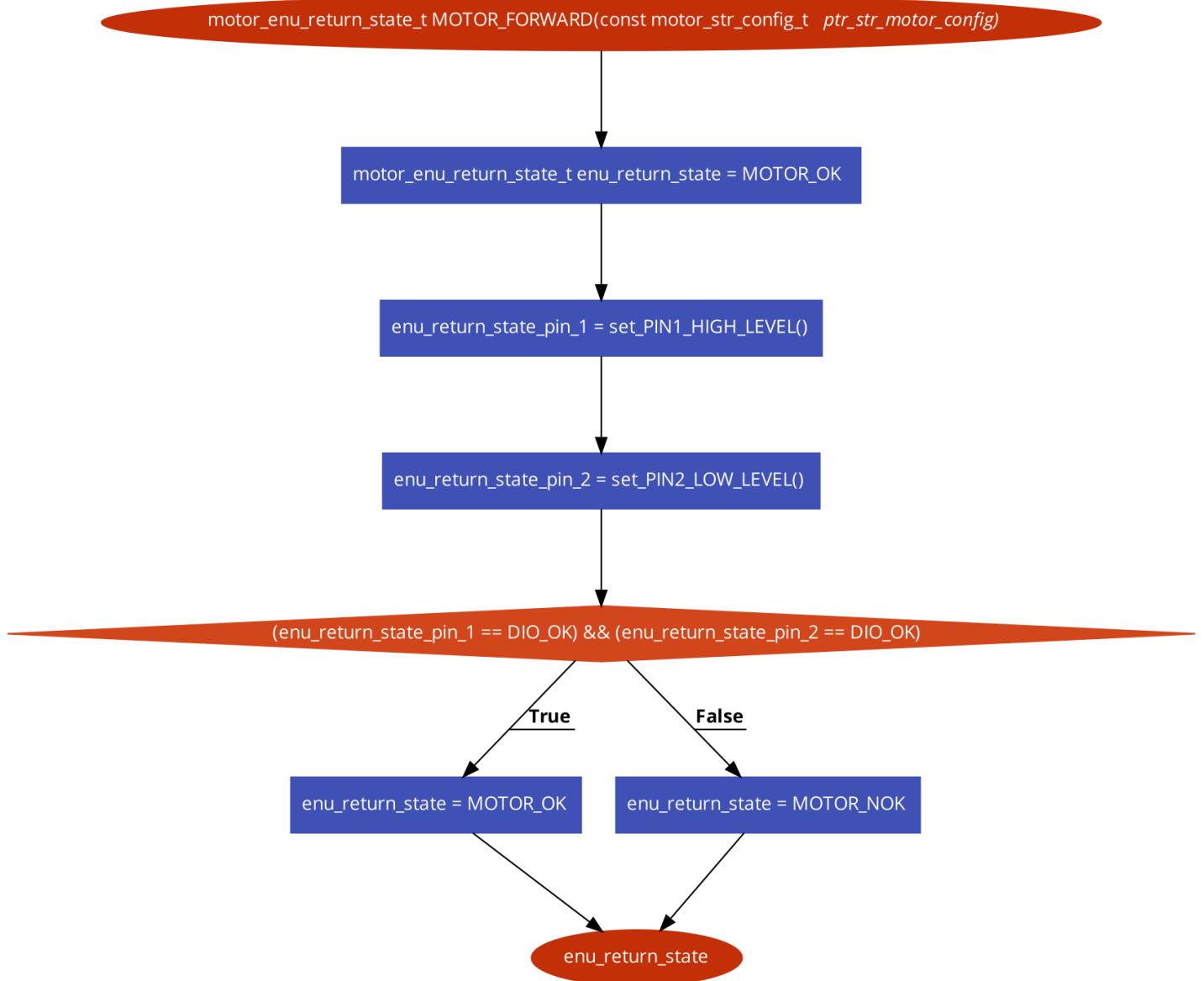


Figure 27 MOTOR\_FORWARD()

```
motor_enu_return_state_t MOTOR_BACKWARD(const motor_str_config_t *ptr_str_motor_config)
```

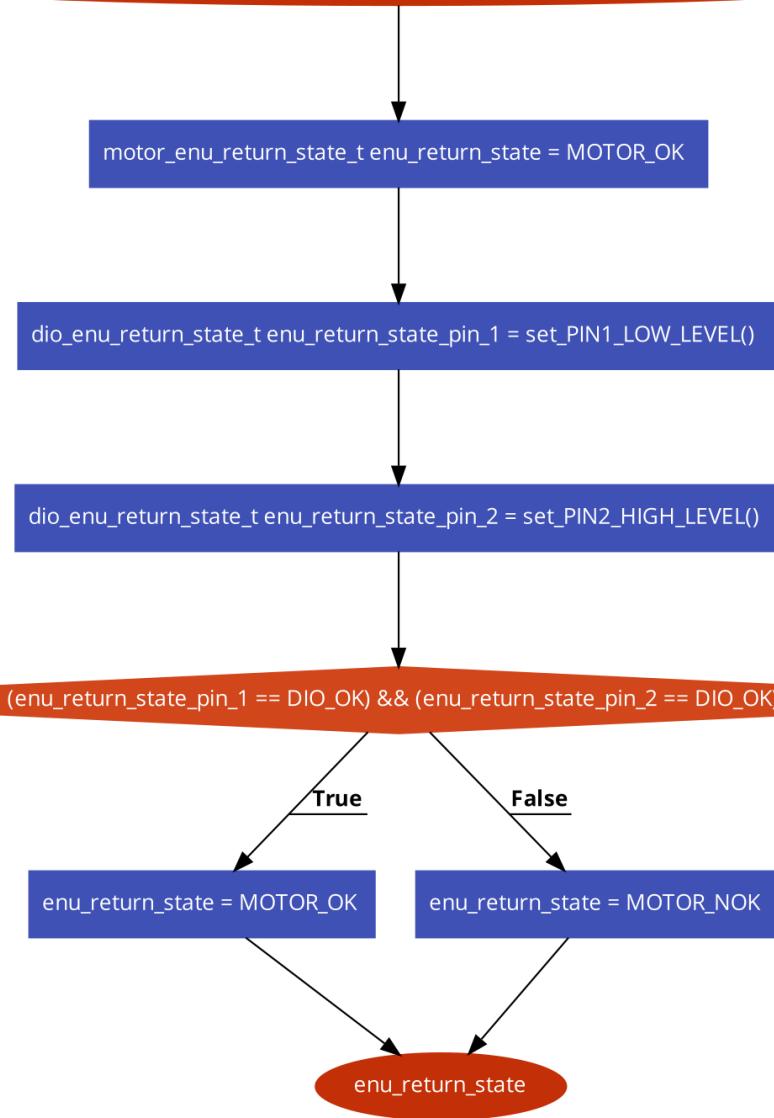
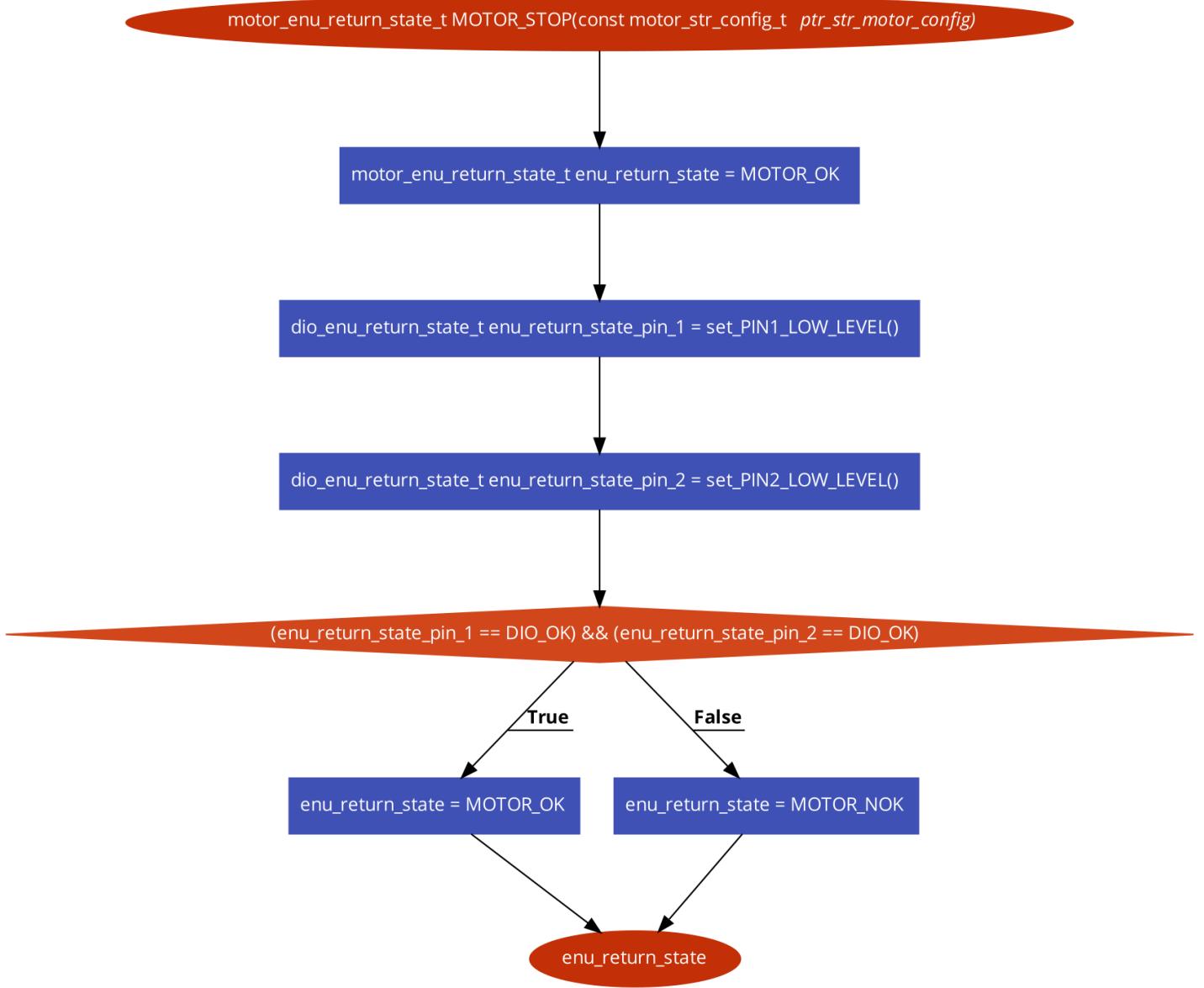


Figure 28 MOTOR\_BACKWARD()



*Figure 29 MOTOR\_STOP()*

## Car Control API :

Flowcharts:

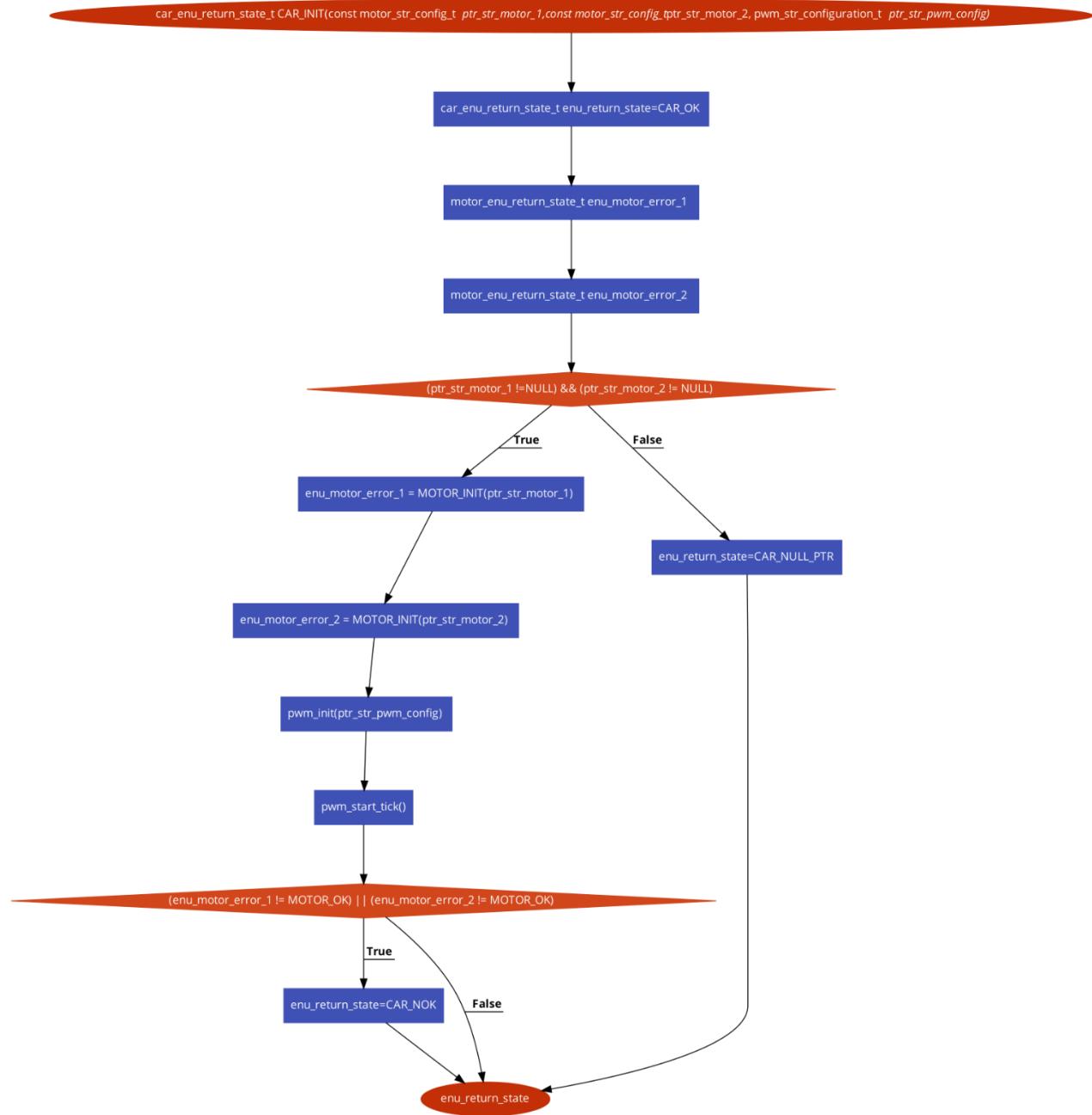


Figure 30 CAR\_INIT()

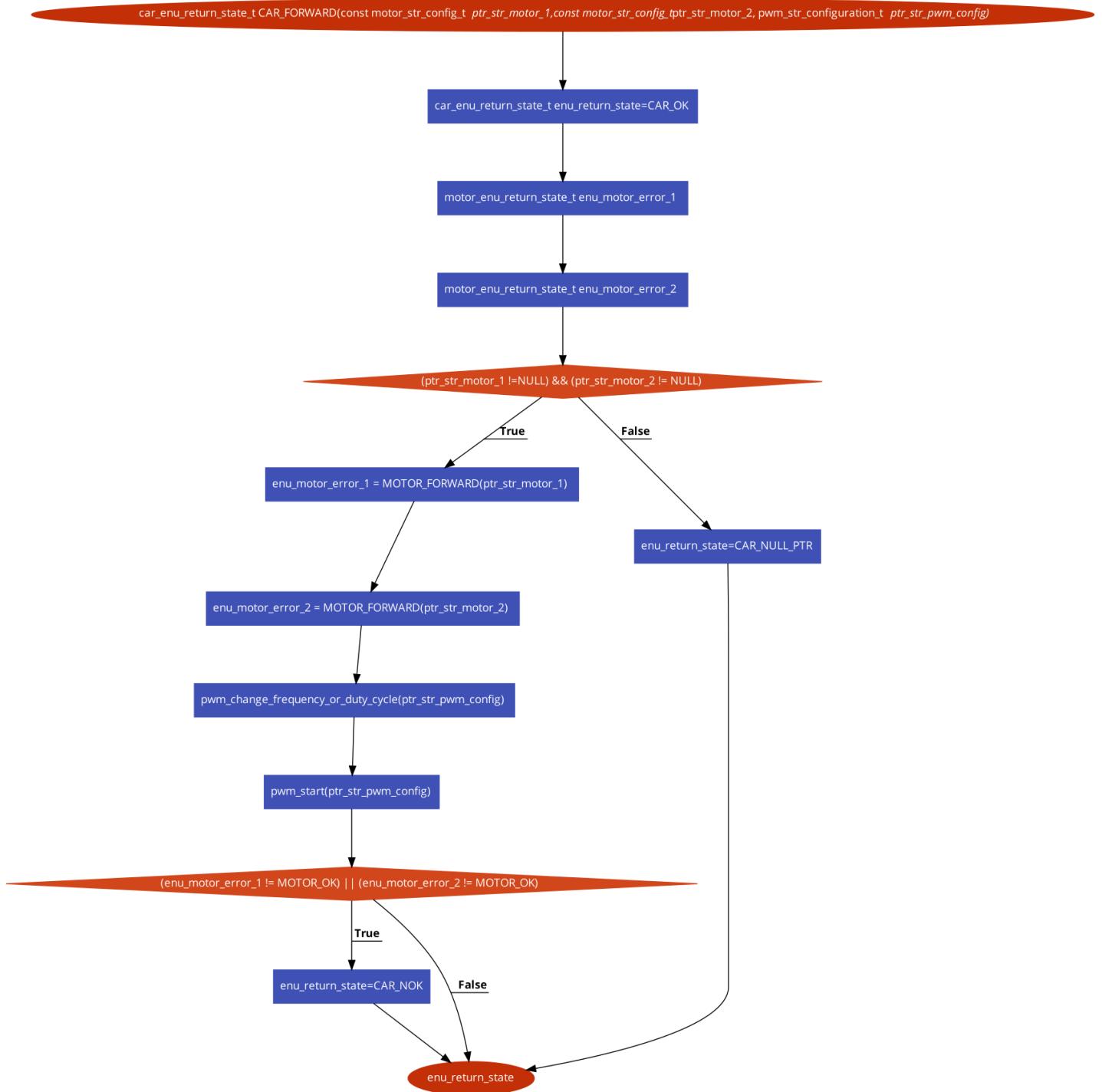


Figure 31 CAR\_FORWARD()

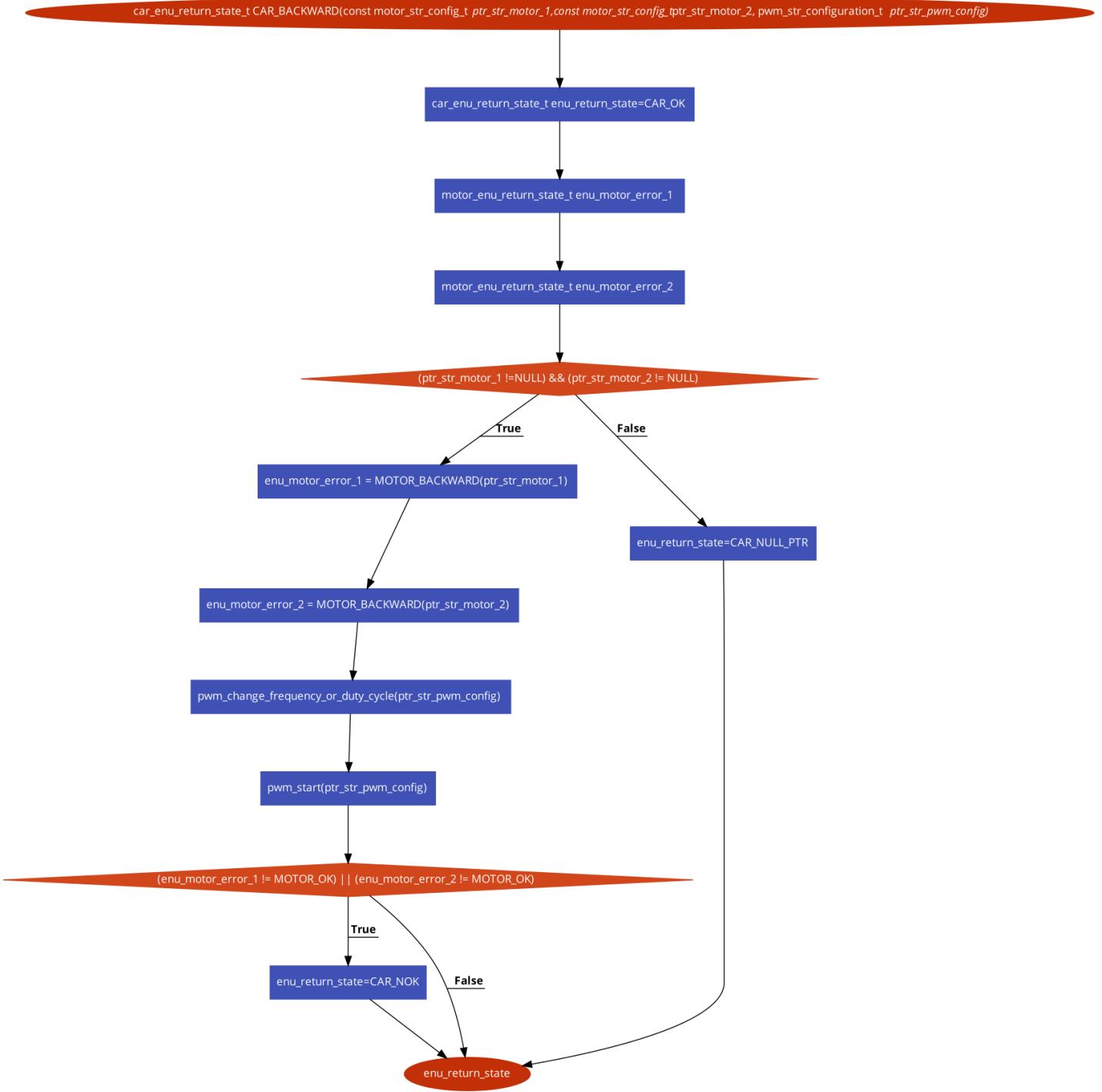


Figure 32 CAR\_BACKWARD()

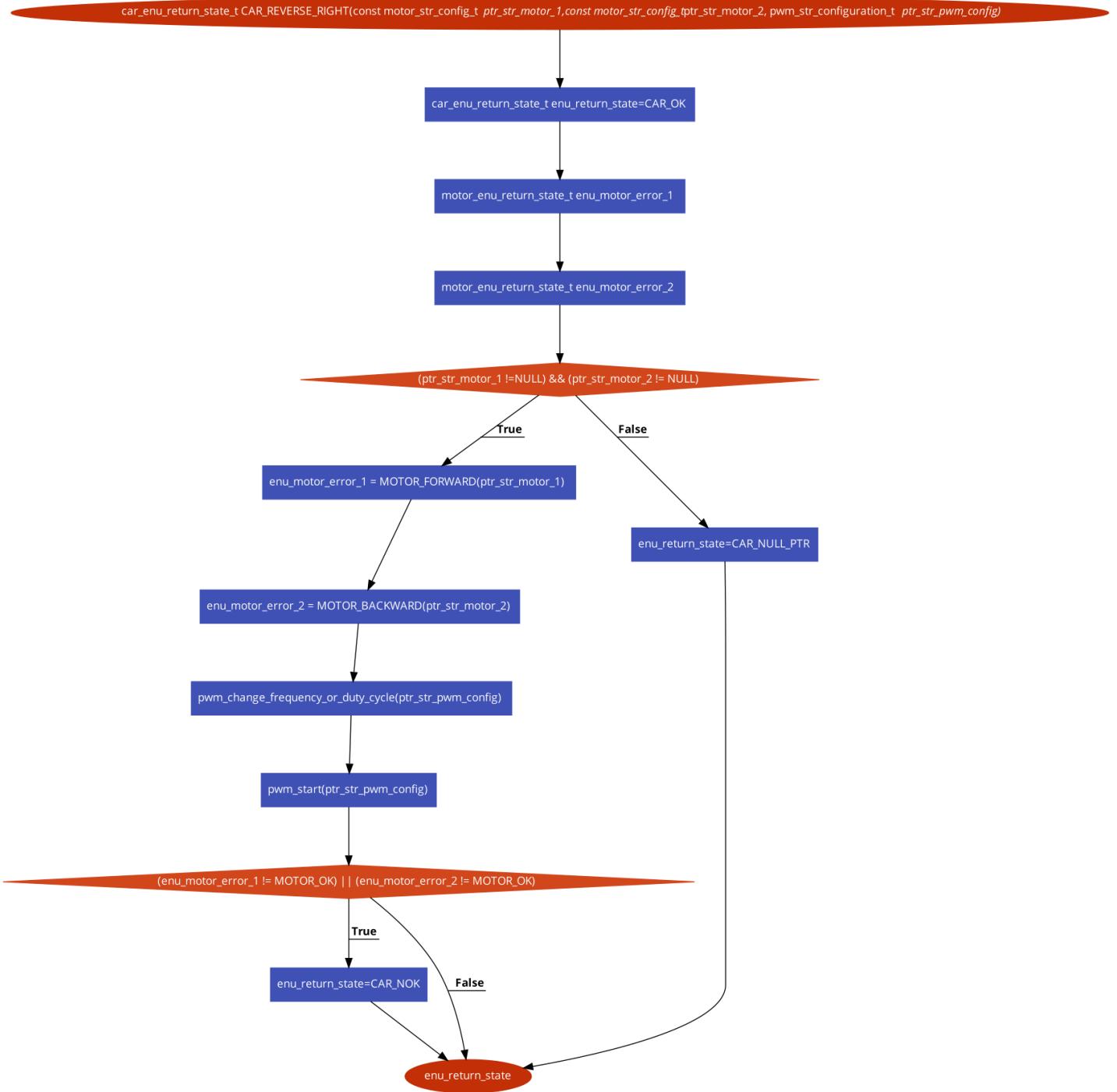


Figure 33 CAR\_REVERSE\_RIGHT()

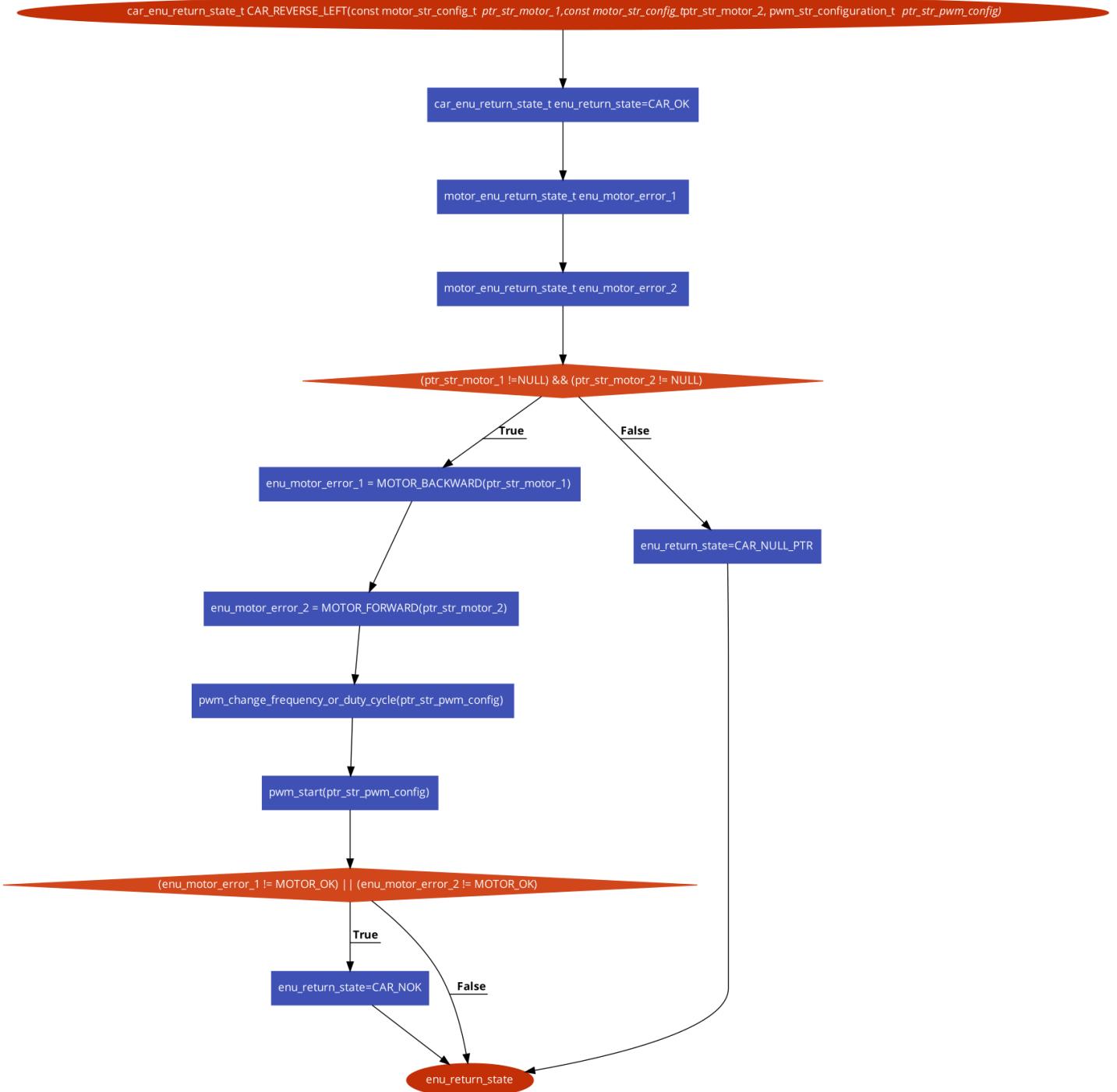


Figure 34 CAR.REVERSE.LEFT()

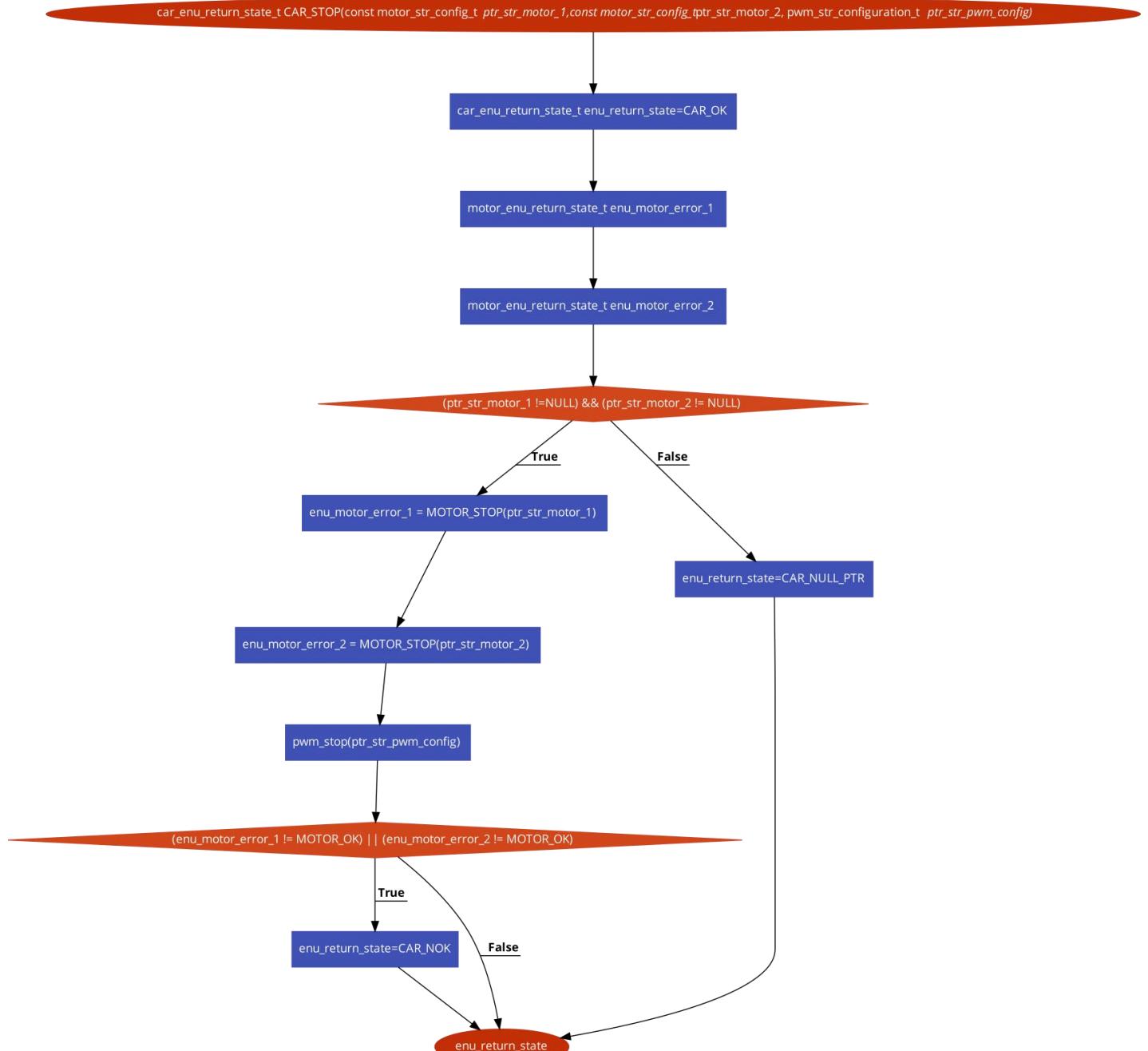


Figure 35 CAR\_STOP

## TIMING API :

Flowcharts:

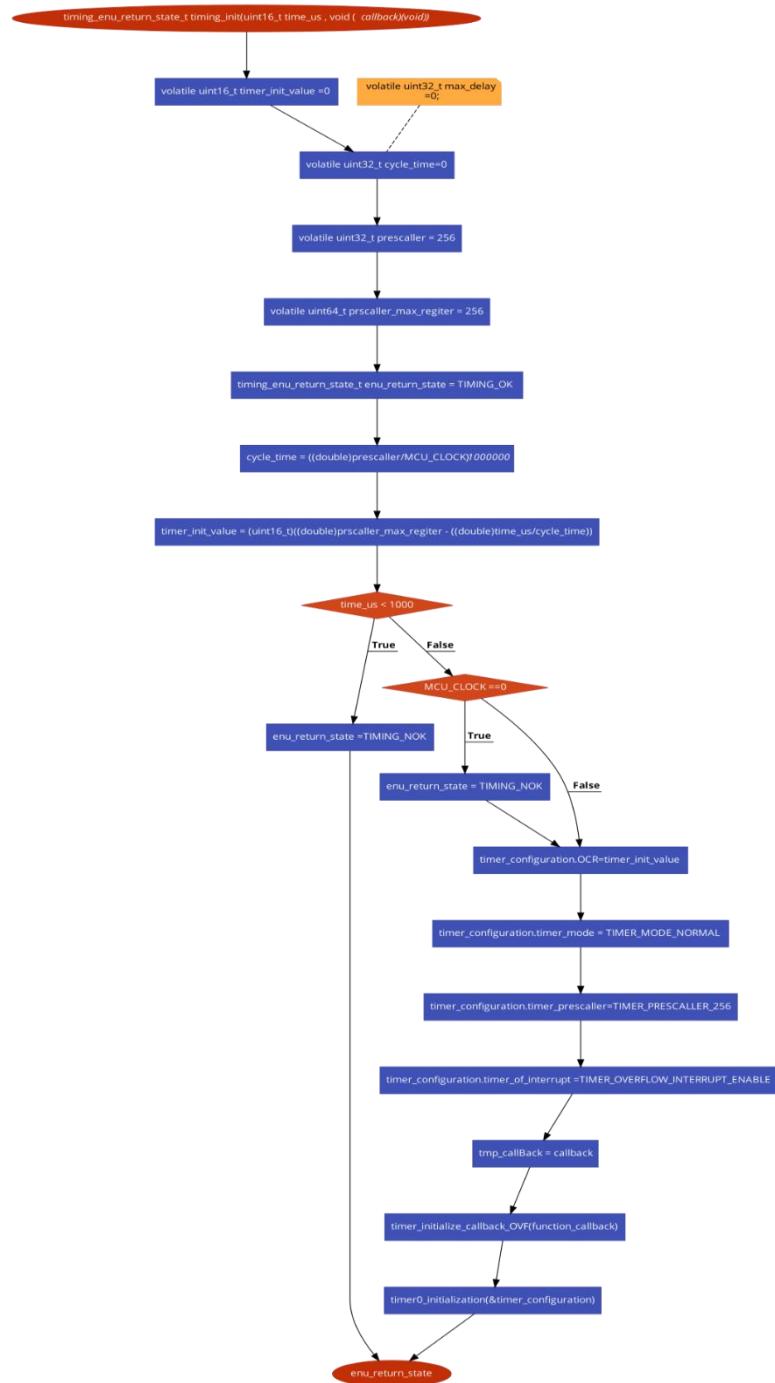


Figure 36 `timing_init()`

void timing\_start(void)

Start the timer using the  
configured timer settings

timer\_start(&timer\_configuration)

Figure 37 timing\_start()

void timing\_stop(void)

Stop the timer that was  
previously started

timer\_stop(&timer\_configuration)

Figure 38 timing\_stop

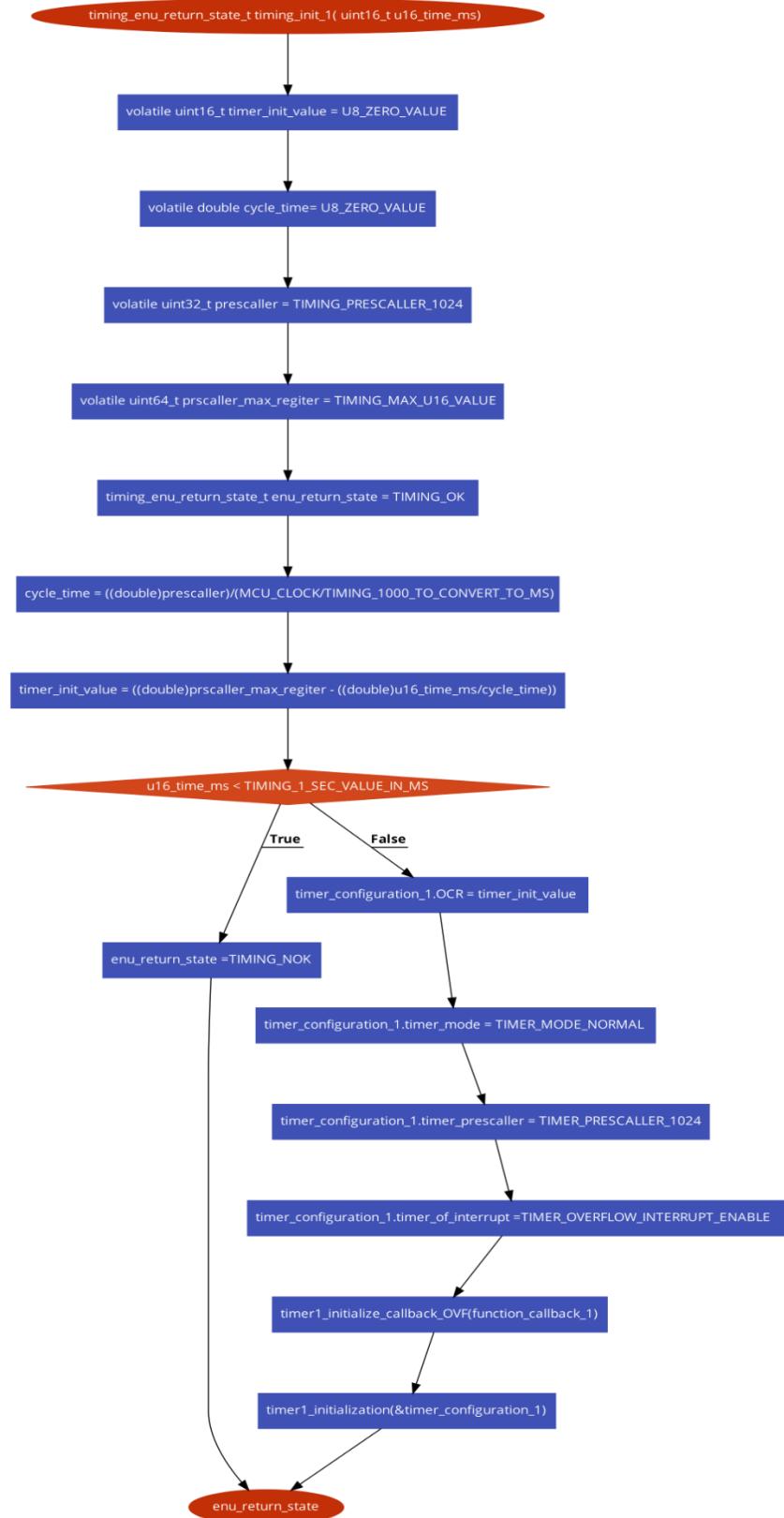


Figure 39 `timing_init_1()`

void timing\_start\_1(void)



timer1\_start(&timer\_configuration\_1)

Figure 40 timing\_start\_1()

void timing\_stop\_1(void)



timer1\_stop(&timer\_configuration\_1)

Figure 42 timing\_stop\_1()

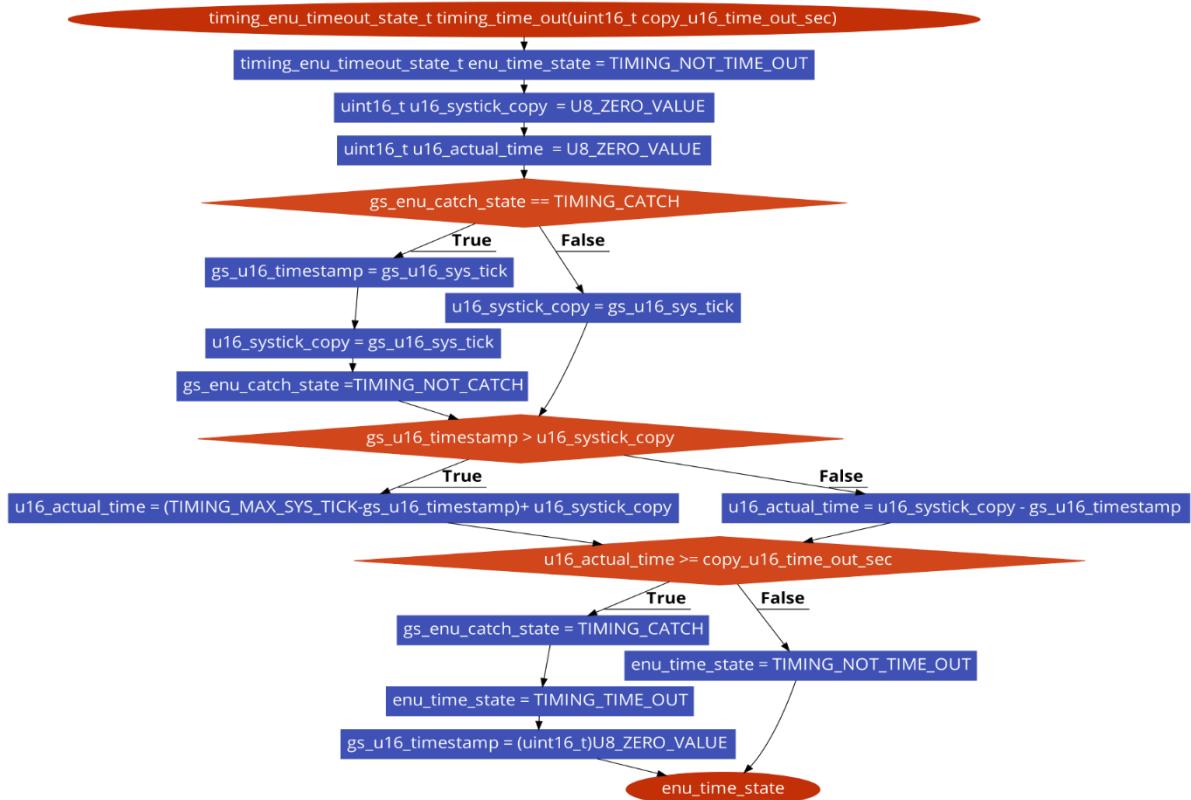


Figure 41 timing\_time\_out()

```
void timing_break_time_out(void)
```

```
    gs_enu_catch_state = TIMING_CATCH
```

```
    gs_u16_timestamp = (uint16_t)U8_ZERO_VALUE
```

Figure 43 timing\_break\_time\_out()

```
void delay_s(uint16_t copy_u16_delay)
```

Loop until the specified  
delay time has passed

True

```
timing_time_out(copy_u16_delay) != TIMING_TIME_OUT
```

Figure 44 delay\_s()

```
timing_enu_return_state_t timing_init_2(void (callback)(void))
```

```
    timing_enu_return_state_t ret = TIMING_OK
```

callback == NULL

```
    timer_configuration_2.OCR= U8_ZERO_VALUE
```

True

```
    ret =TIMING_NOK
```

False

```
    timer_configuration_2.timer_mode = TIMER_MODE_NORMAL
```

```
    timer_configuration_2.timer_prescaller=TIMER_PRESCALLER_0
```

```
    timer_configuration_2.timer_of_interrupt =TIMER_OVERFLOW_INTERRUPT_ENABLE
```

```
    timer2_initialize_callback_OVF(callback)
```

```
    timer2_initialization(&timer_configuration_2)
```

```
ret
```

Figure 45 timing\_init\_2()

void timing\_start\_2(void)



timer2\_start(&timer\_configuration\_2)

Figure 46 timing\_start\_2()

void timing\_stop\_2(void)



timer2\_stop(&timer\_configuration\_2)

Figure 47 timing\_stop\_2()

void timing\_get\_ticks\_2(uint8\_t \*ptr\_u8\_ticks)



timer2\_get\_tcnt(&timer\_configuration\_2,ptr\_u8\_ticks)

Figure 48 timing\_get\_ticks\_2()

## Interrupt Manager API :

Flowcharts:

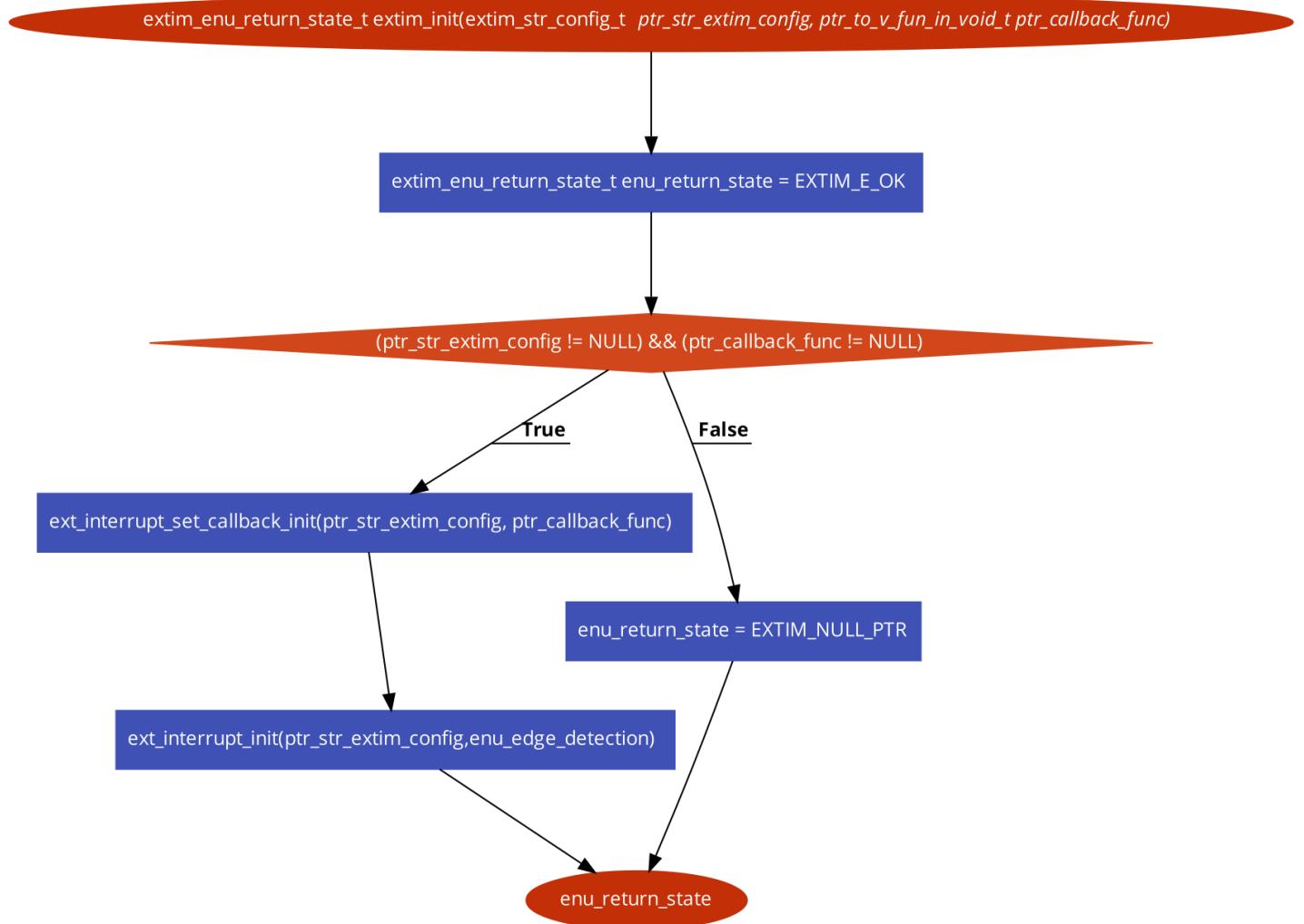


Figure 49 extim\_init()

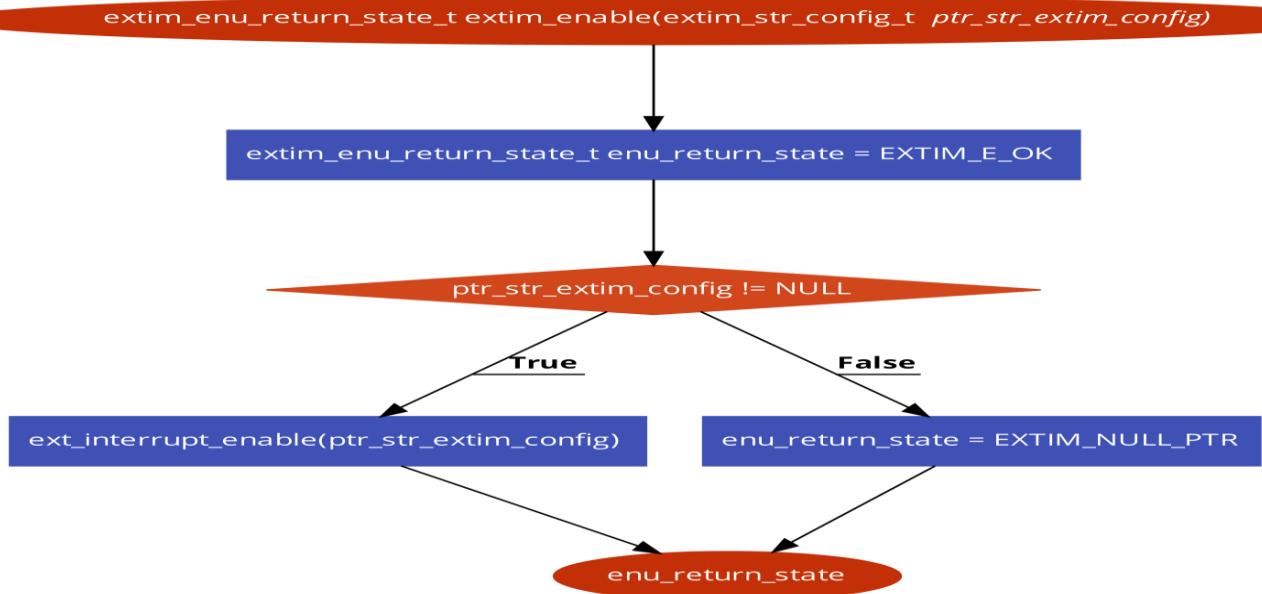


Figure 50 extim\_enable()

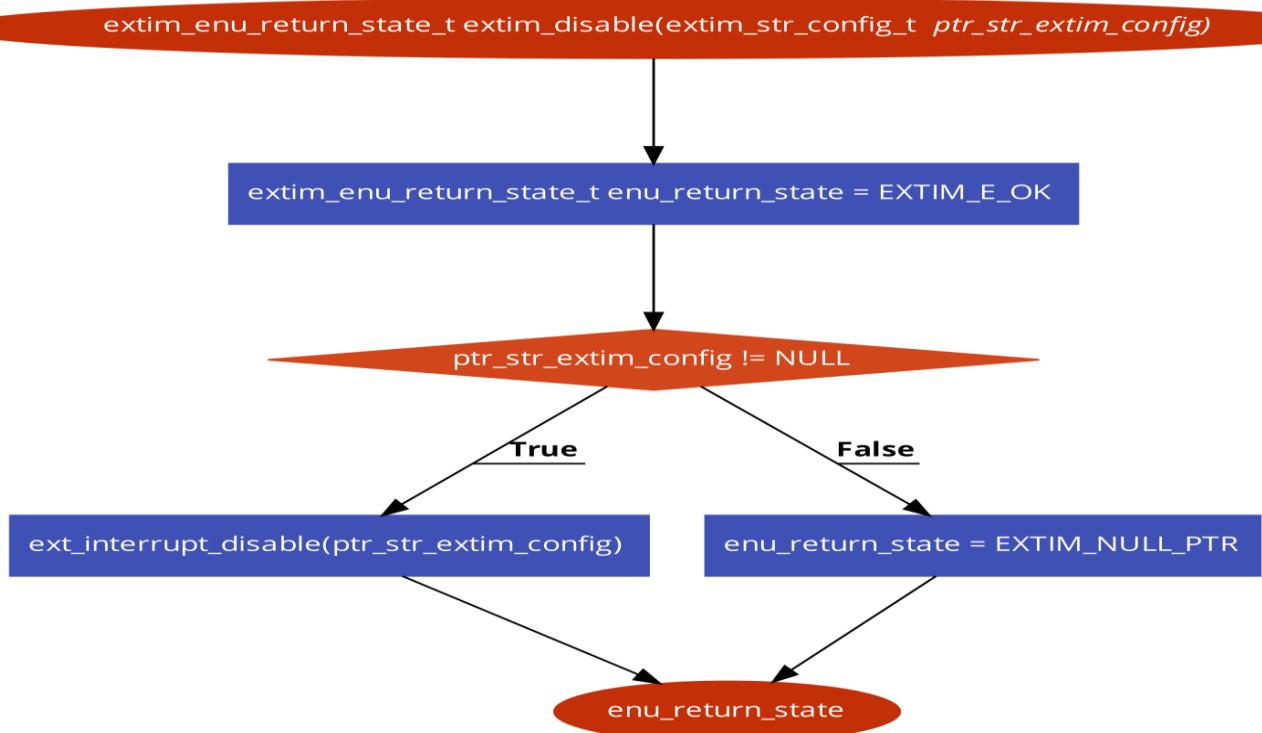


Figure 51 extim\_disable()

## PWM API:

Flowcharts:

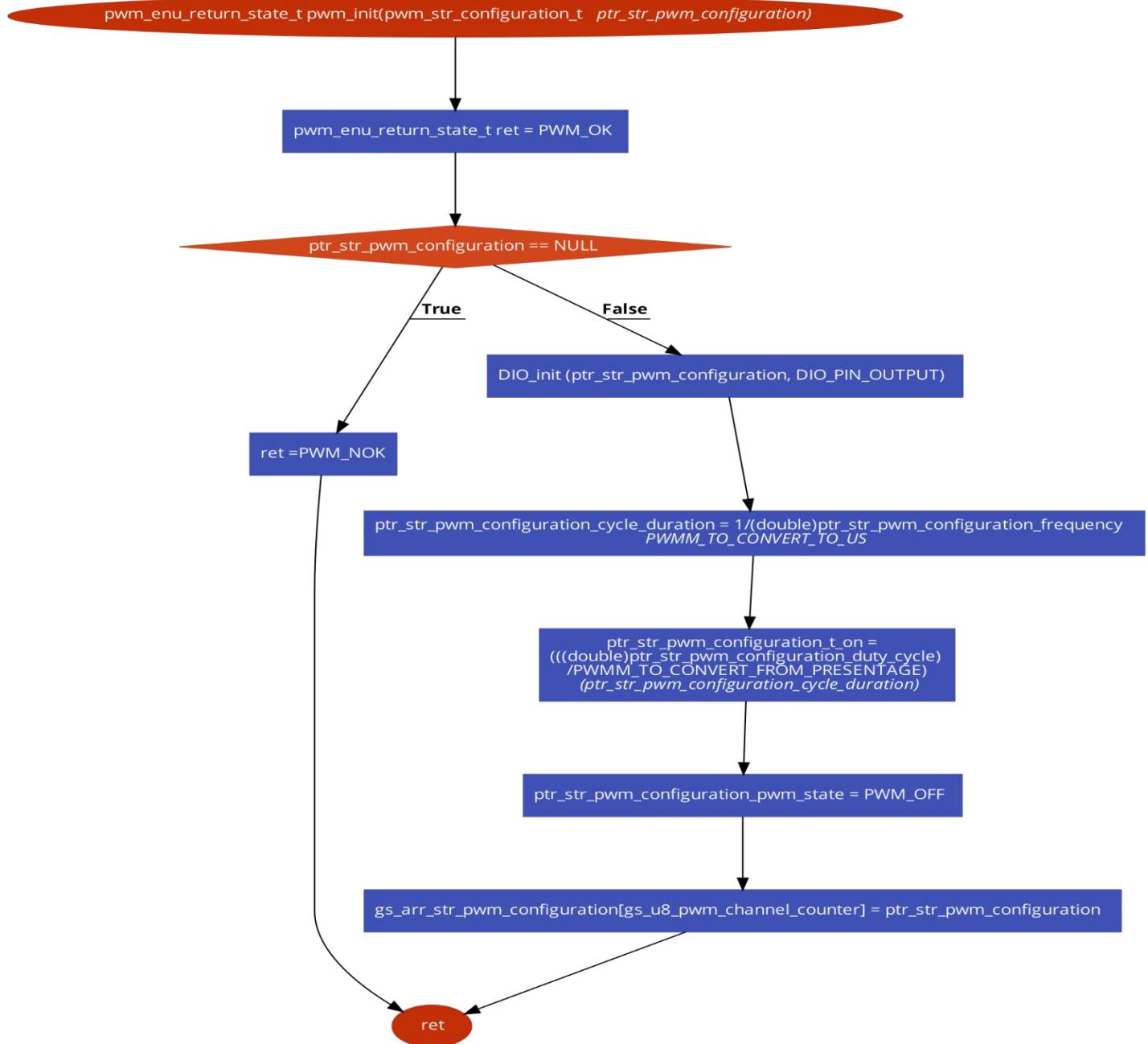


Figure 52 `pwm_init`

void pwm\_start\_tick(void)

timing\_init(1000, pwm\_tick\_counter)

timing\_start()

Figure 54 pwm\_start\_tick()

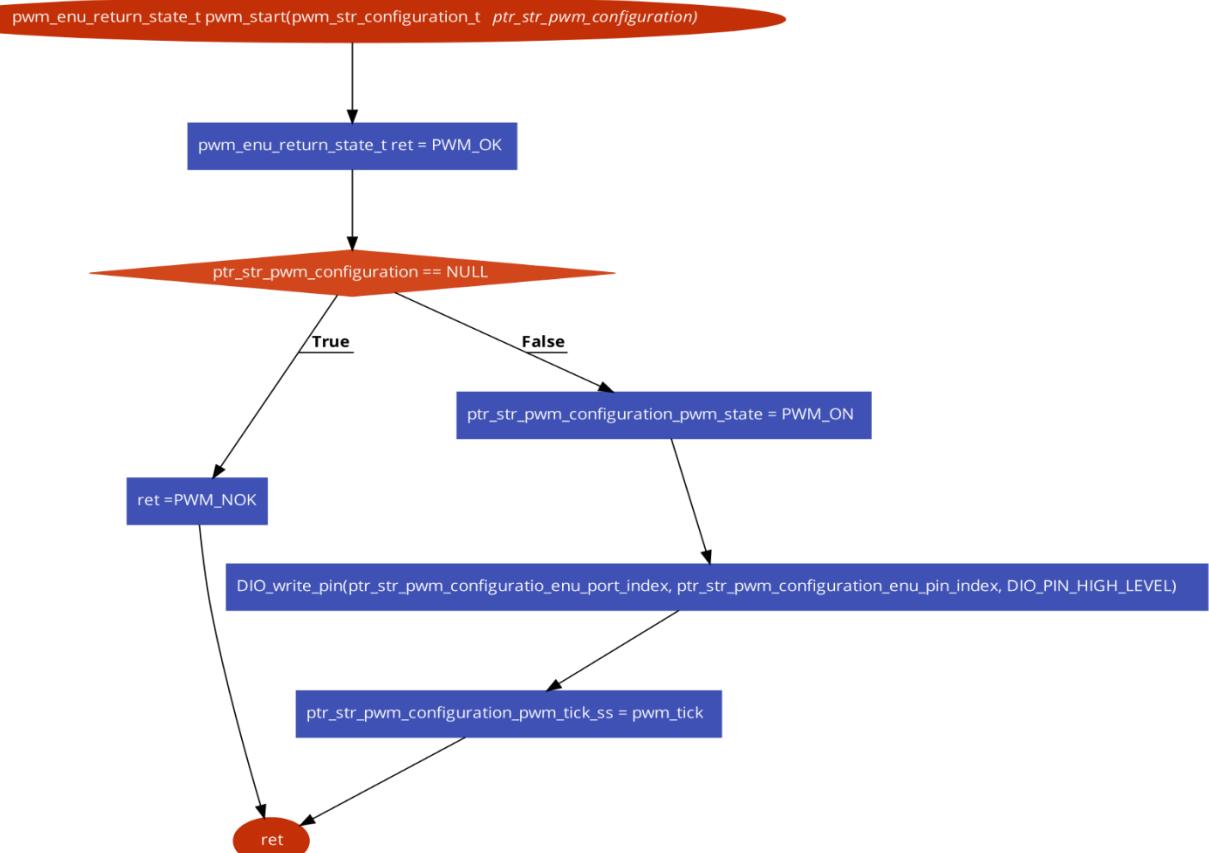


Figure 53 pwm\_start()

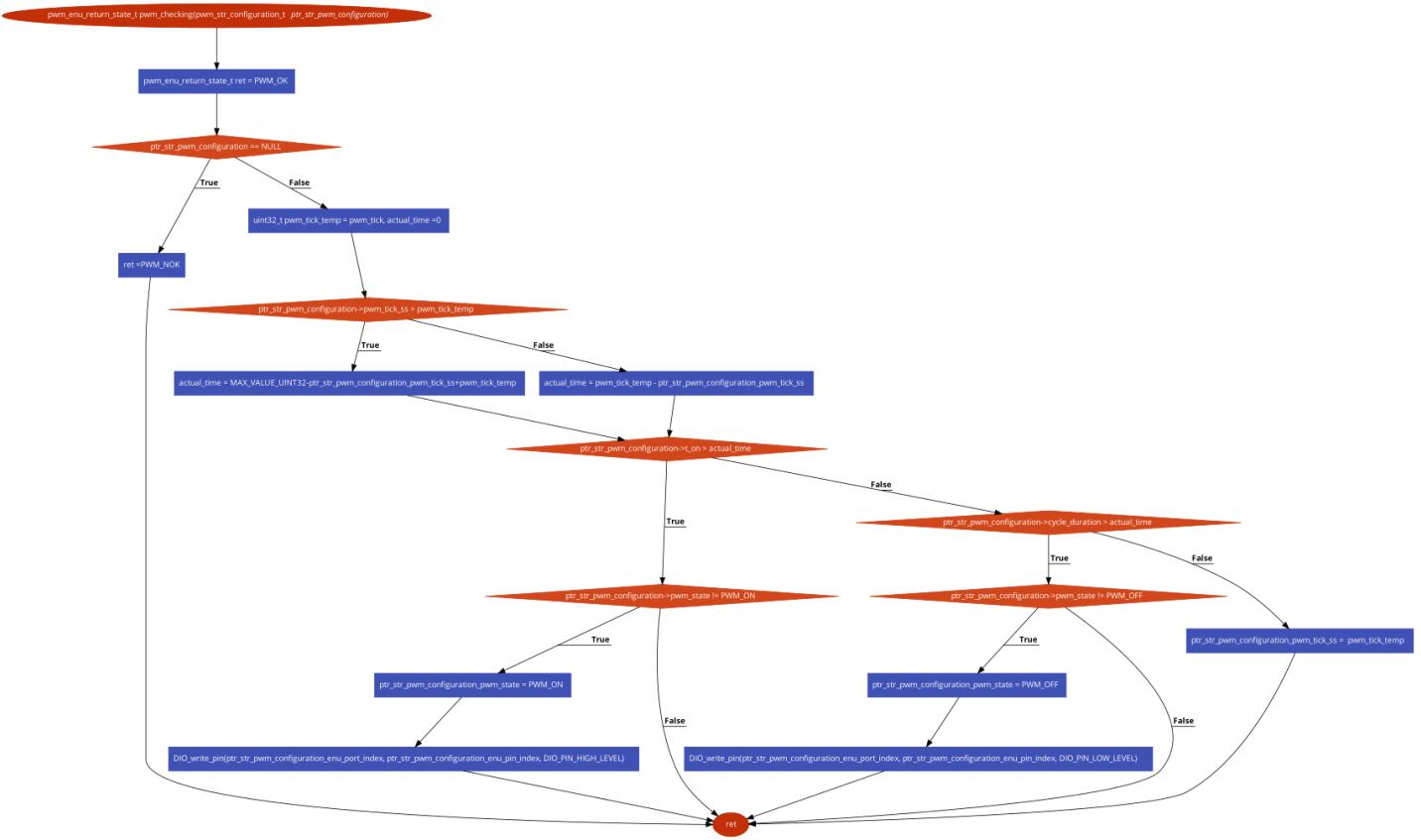


Figure 55 `pwm_checking()`

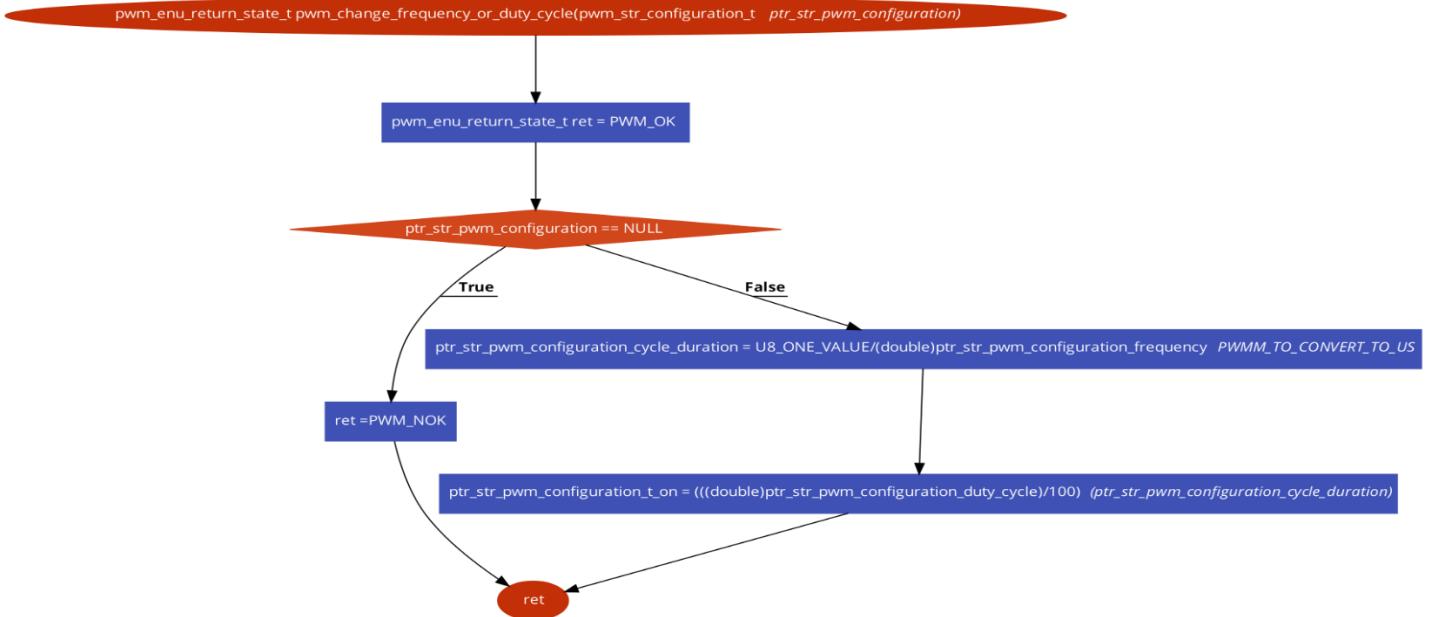


Figure 56 `pwm_change_frequency_or_duty_cycle()`

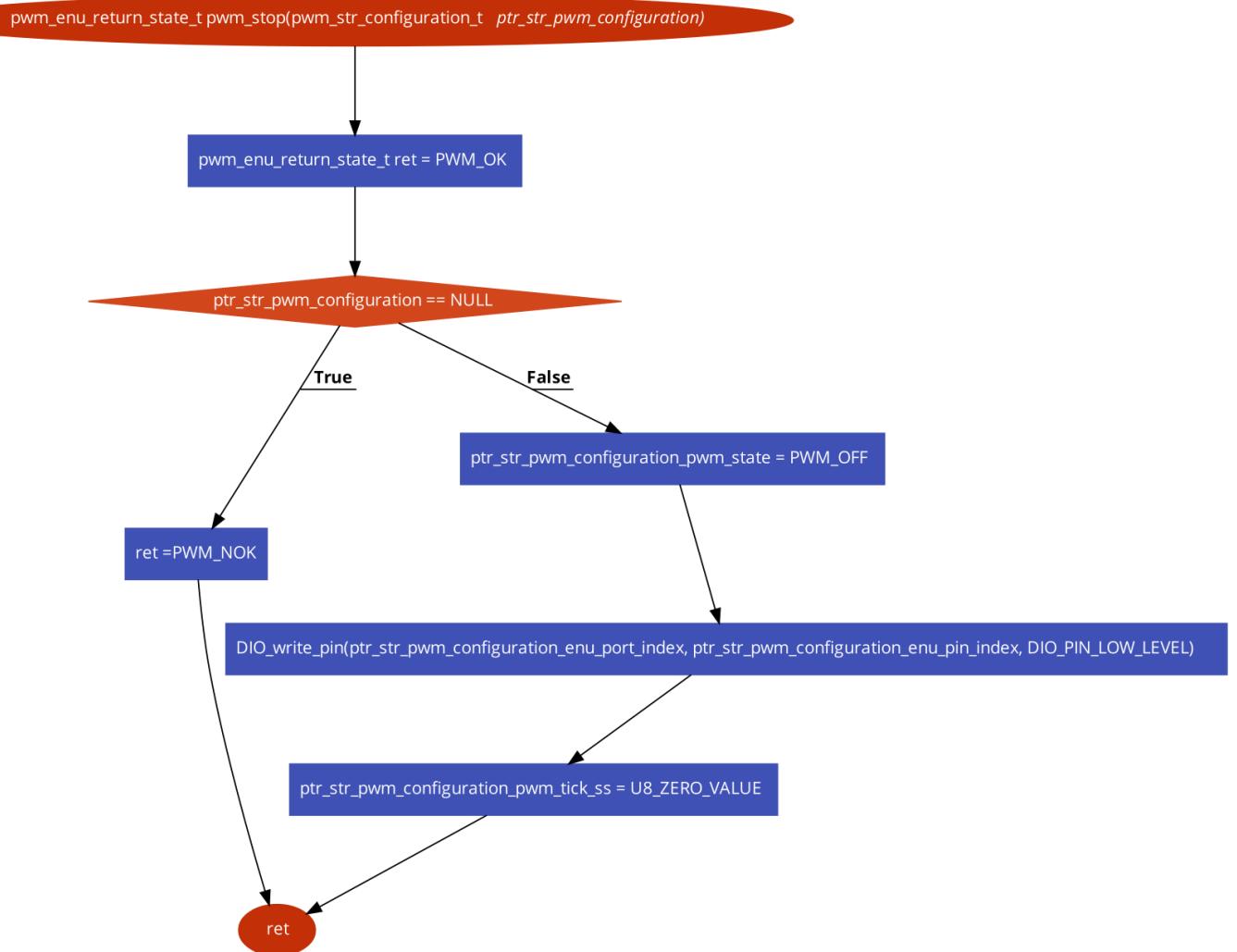


Figure 57 `pwm_stop()`

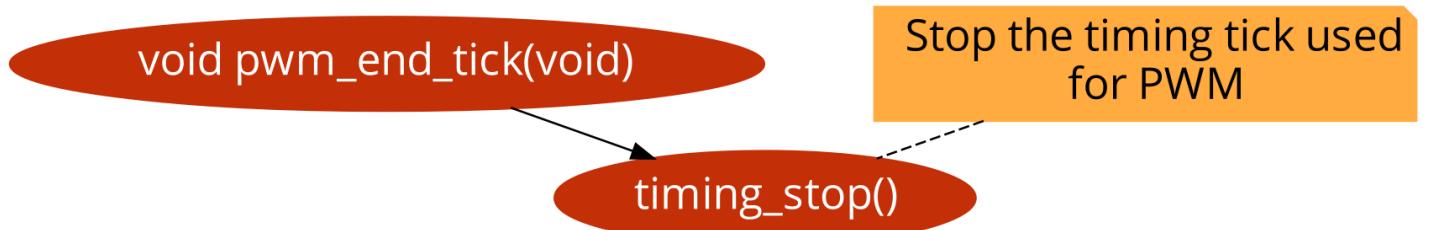


Figure 58 `pwm_end_tick()`



## Button API:

Flowcharts:

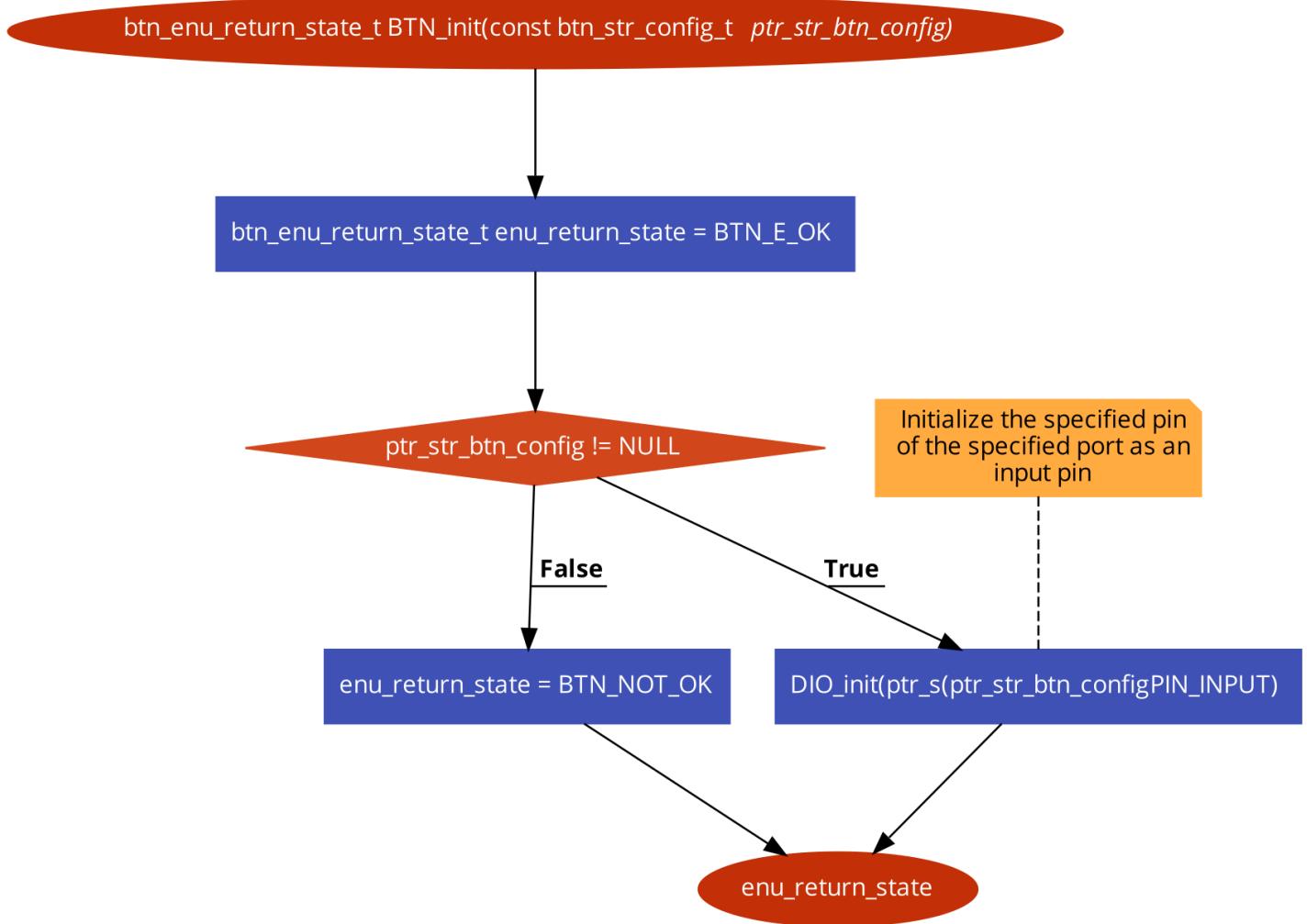
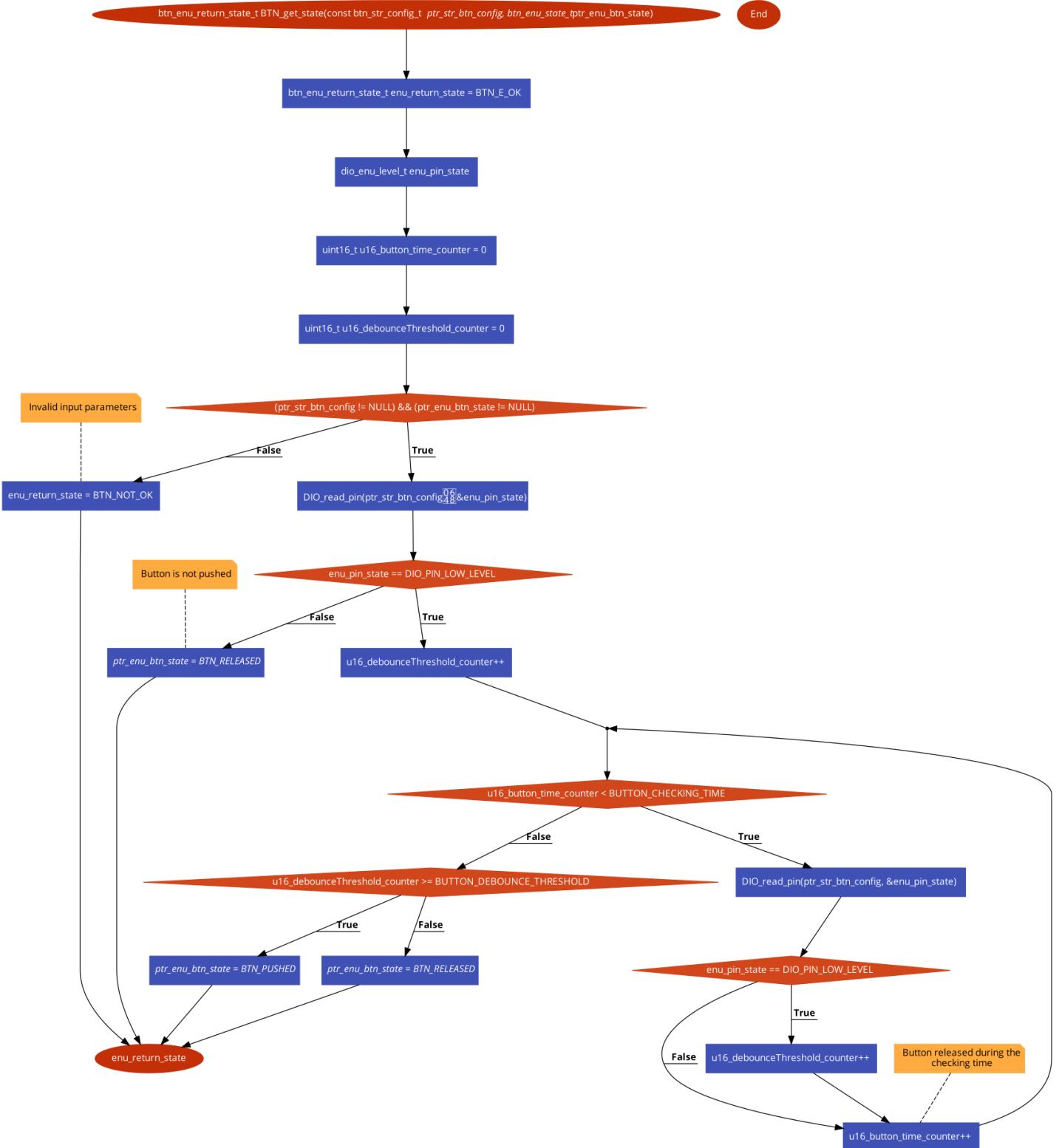


Figure 59 `BTN_init()`





# App

App API:

Flowcharts:

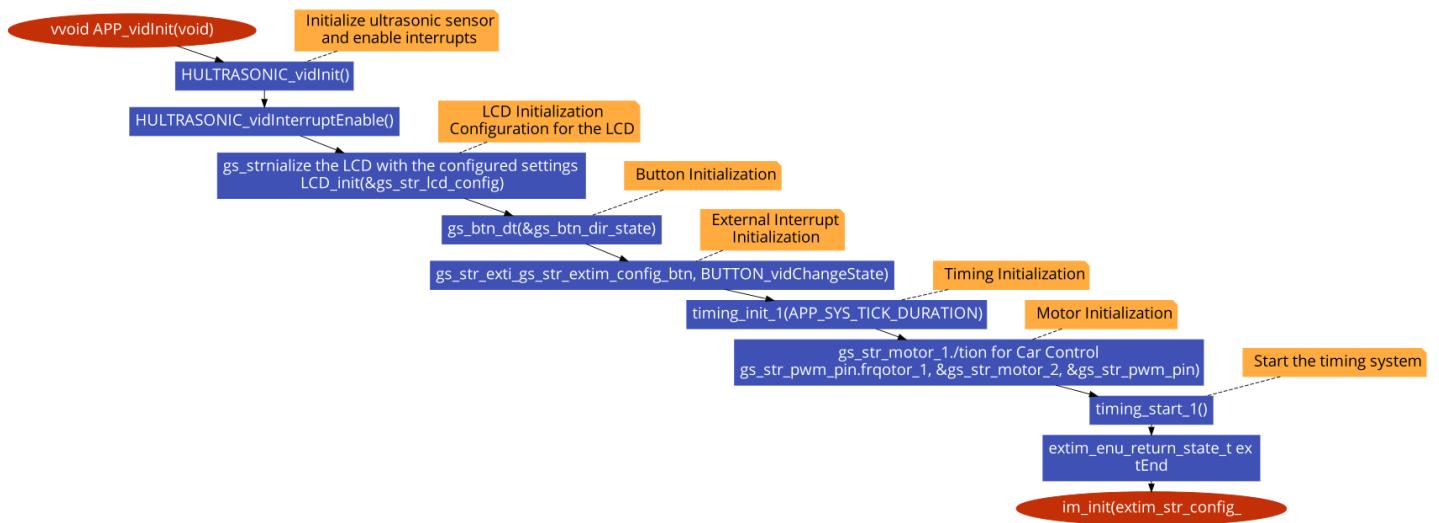


Figure 61 APP\_init()

Figure 62 APP\_start() added besides the pdf as SVG

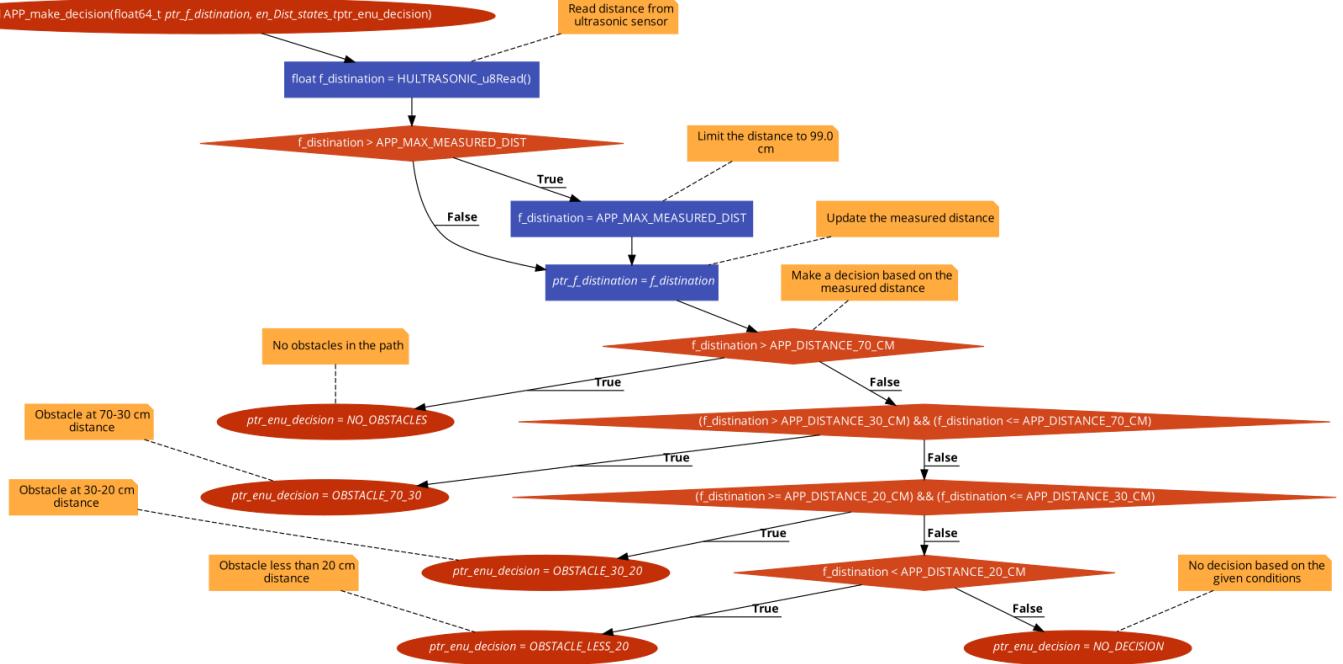


Figure 63 APP\_make\_decision()

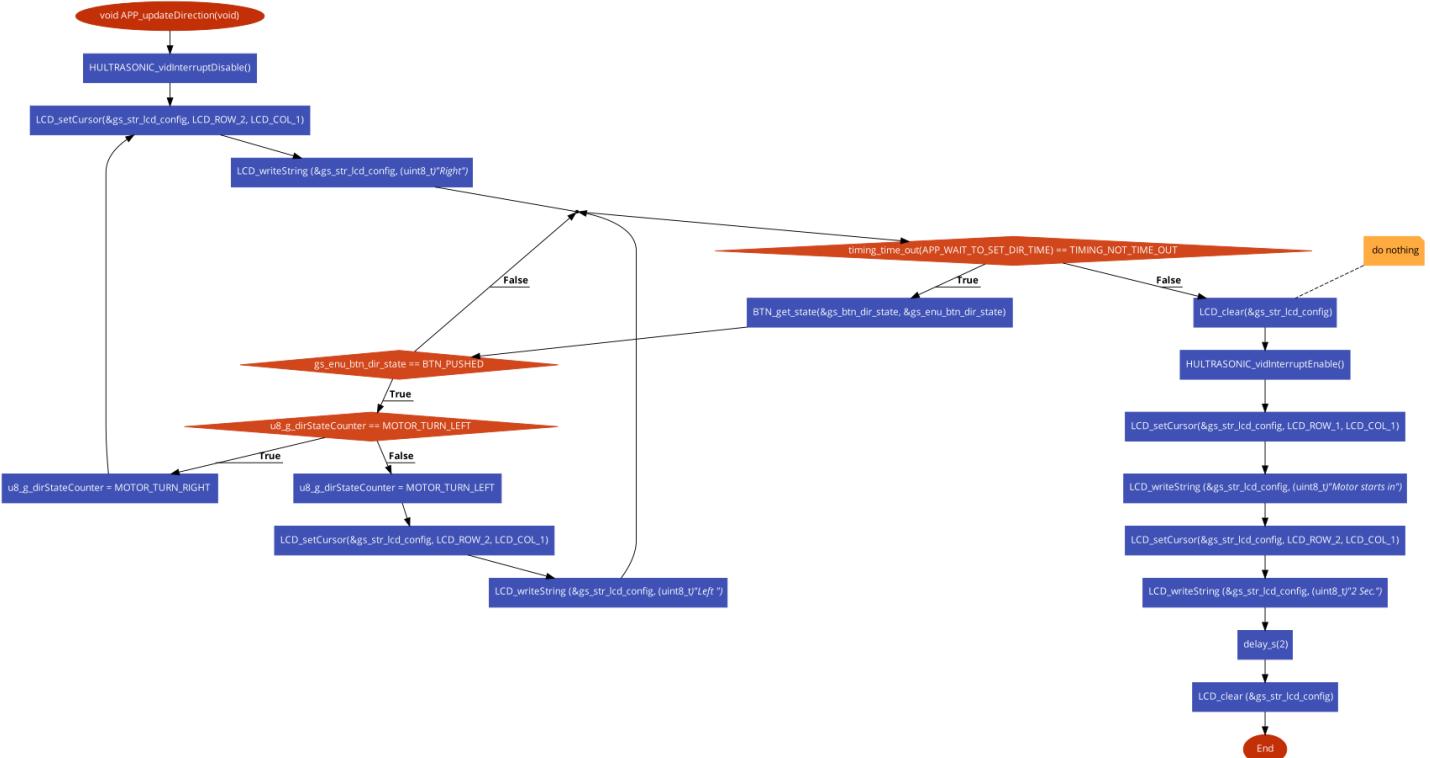


Figure 64 APP\_updateDirection()

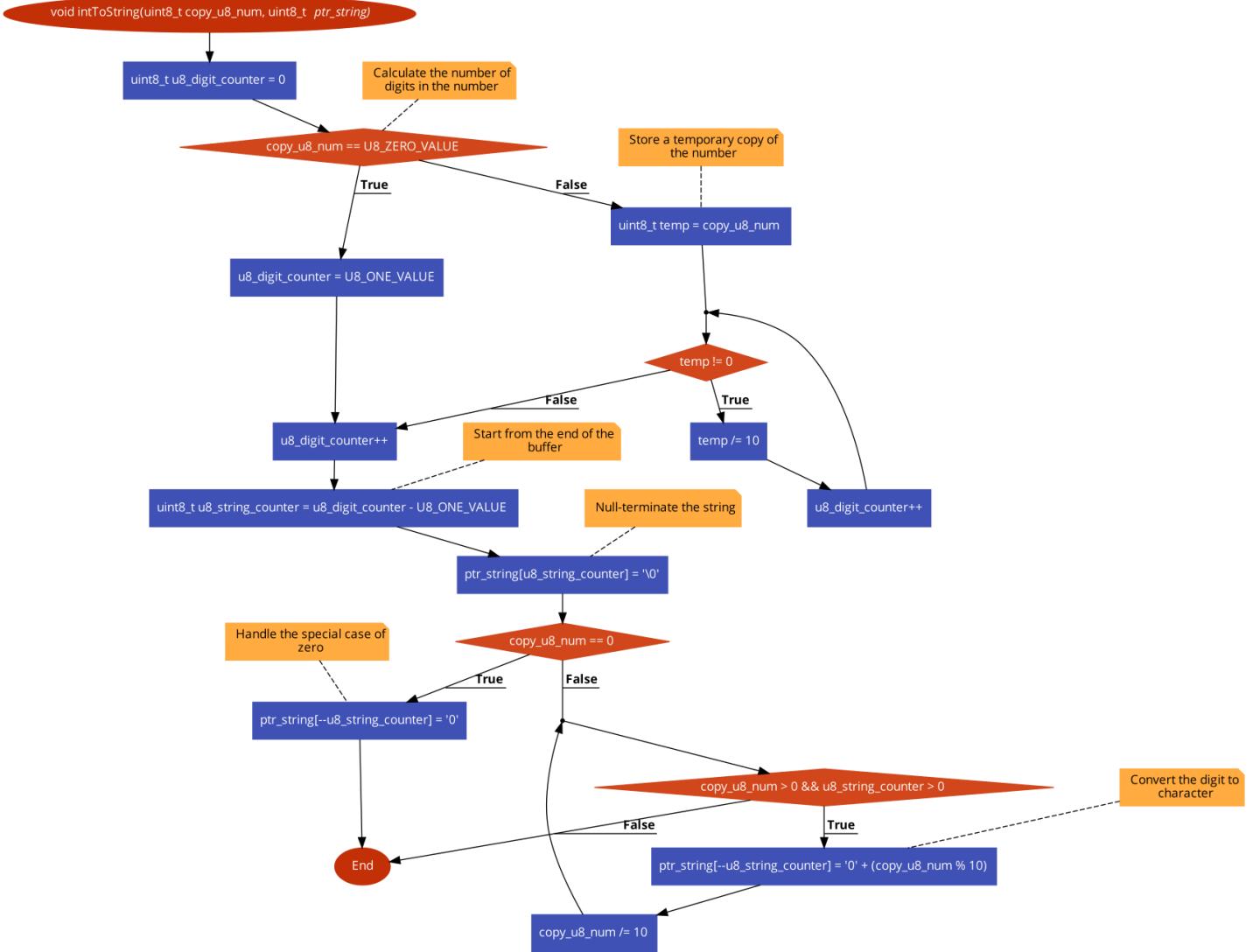


Figure 65 `intToString()`

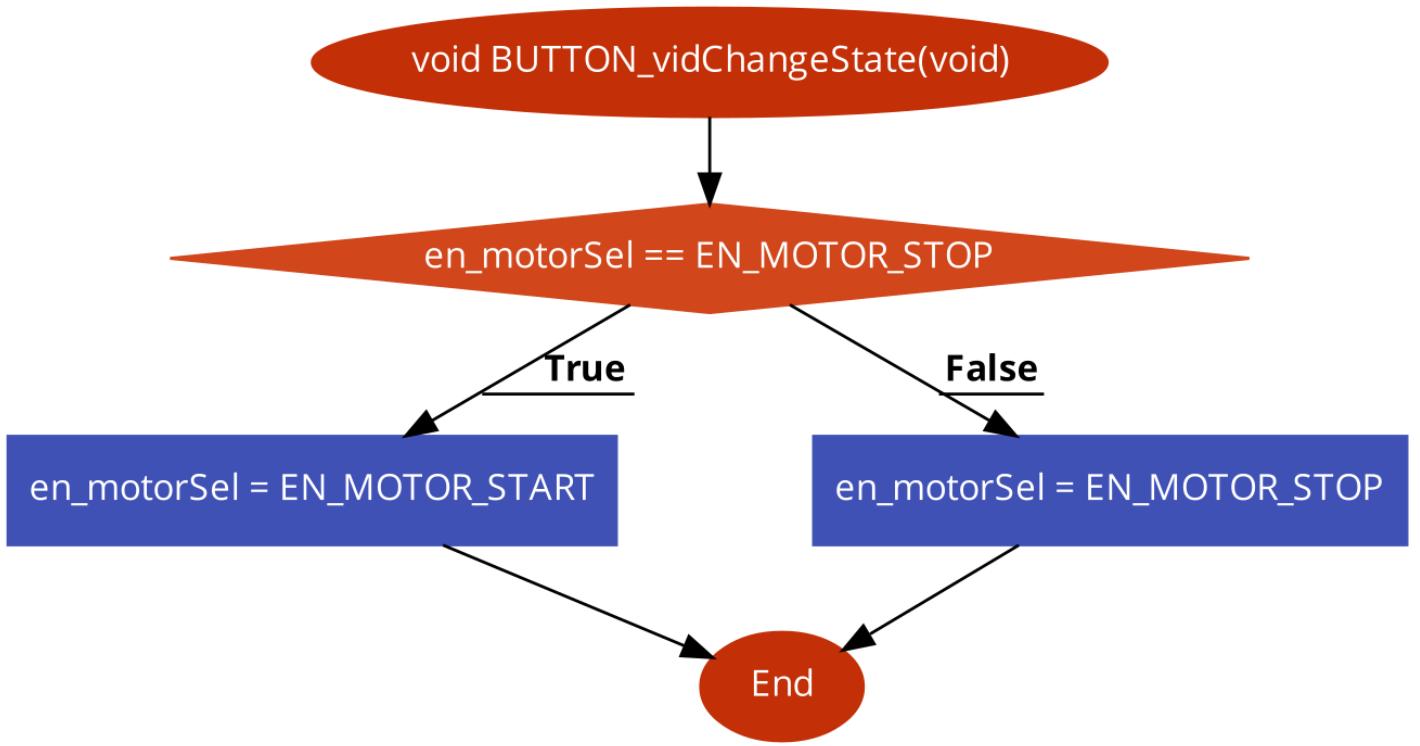


Figure 66 `BUTTON_vidChangeState()`