Sprints

Avoid Obstacle Car

Static Design

Bassel Yasser Mahmoud

2023

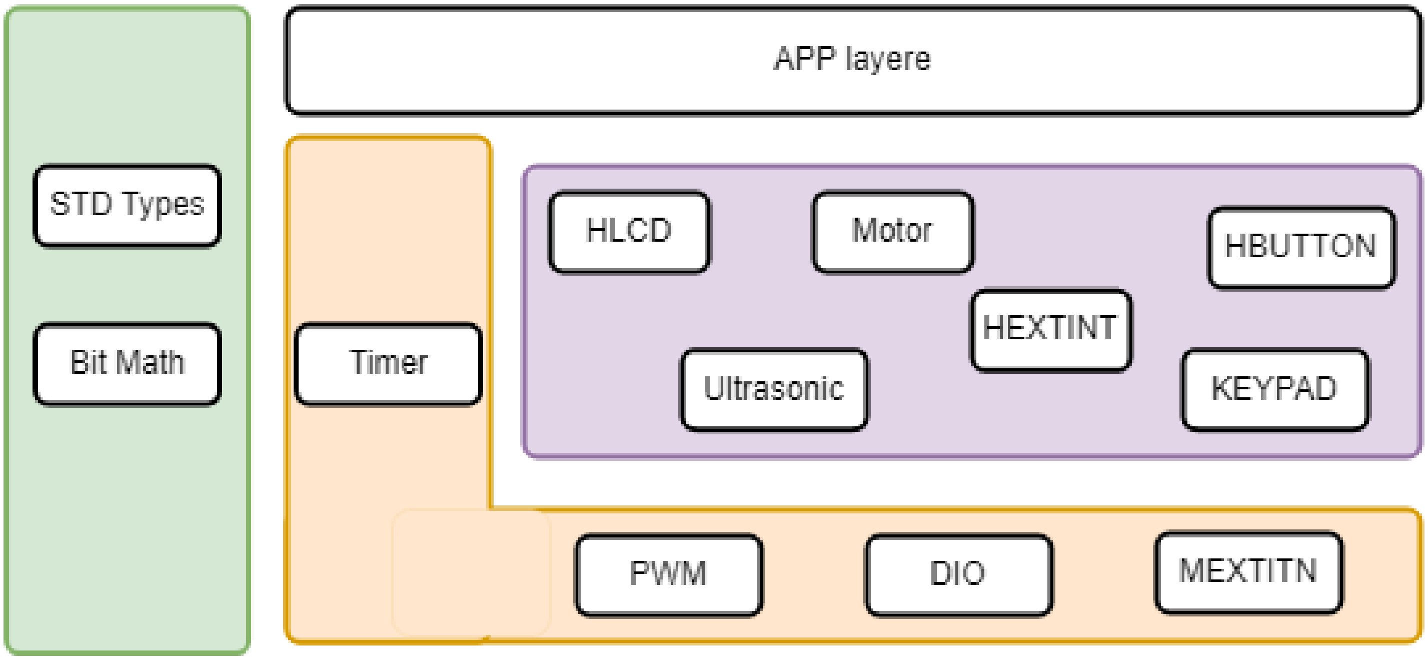
Contents

Introduction

Are you looking for a way to protect your property from damage or casualties caused by robots? Our obstacle-avoiding robot or car is the perfect solution! Our device is in motion and can detect obstacles ahead of it and take necessary action to avoid them. This will help you reduce costs associated with damages and casualties caused by robots. With our device, you can rest assured that your property is protected from any potential harm.

Obstacle detection is the primary requirement of this autonomous robot. The robot gets the information from the surrounding area through mounted sensors on the robot. Some sensing devices used for obstacle detection like bump sensors, infrared sensors, ultrasonic sensors, etc. The ultrasonic sensor is most suitable for obstacle detection and it is of low cost and has a high ranging capability.

Layered Architecture



HAL Layer

APIs

HLCD

```
/*
 * function          : HLCD_vidInit
 * description       : func to set LCD initialization
 * input param       : void
 * return            : void
 * */
void HLCD_vidInit(void)
```

```
/*
 * function          : HLCD_vidWritecmd
 * description       : func to configure some commands on lcd
 * input param       :
 *                               u8commandCopy --> take lcd cmd instructions from
instruction table
<https://components101.com/sites/default/files/component_datasheet/16x2%20LCD%20Datas
heet.pdf>
 * return            : void
 * */
void HLCD_vidWritecmd (Uint8_t u8commandCopy)
```

```
/*
 * function          : HLCD_vidWriteChar
 * description       : func to write char on lcd
 * input param       : u8CharCopy -> take ascii code of char   or   char address on
CGROM
 * return            : void
 * */
void HLCD_vidWriteChar (Uint8_t u8CharCopy)
```

```
/*
 * function          : HLCD_ClrDisplay
 * description       : func to clear anything on lcd
 * input param       : void
 * return            : void
 * */
void HLCD_ClrDisplay(void)
```

```
/*
 * function         : HLCD_ShiftLeft
 * description      : func to shift the lcd display from right to left
 * input param      : void
 * return           : void
 * */
void HLCD_ShiftLeft(void)
```

```
/*
 * function         : HLCD_gotoXY
 * description      : func to determine position which char print at this position on
lcd   ### NOTE : (2rows x 16coloms)
 * input param      :
 *                          row -> take row number 0 or 1
 *                          pos -> take colom number from 0 ~ 16
 * return           : void
 * */

void HLCD_gotoXY (Uint8_t row, Uint8_t pos)
```

```
/*
 * function         : HLCD_WriteString
 * description      : func to write string on lcd
 * input param      : str --> which take string as argument
 * return           : void
 * */
void HLCD_WriteString (Uint8_t* str)
```

```
/*
 * function         : HLCD_WriteInt
 * description      : func to write integer number on lcd
 * input param      : number --> which take number as argument
 * return           : void
 * */
void HLCD_WriteInt (Uint32_t number)
```

```
/*
 * function         : HLCD_vidCreatCustomChar
 * description      : func to store new patterm on CGRAM
 * input param      :
 *                          pu8custom  -> take pointer to array which having LCD
Custom Character Generated data ### take only 8 characters
 *                          u8Location -> determine location on CGRAM [0 ~ 8]
 * return           : void
 * */
void HLCD_vidCreatCustomChar (Uint8_t* pu8custom, Uint8_t u8Location)
```

HButton

```
/*
* AUTHOR                   : Bassel Yasser Mahmoud
* FUNCTION                 : HButton_Init
* DESCRIPTION       : Initialize specified pin as input and pull up
* RETURN                   : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_Init (enu_pin en_pinx)
```

```
/*
* AUTHOR                   : Bassel Yasser Mahmoud
* FUNCTION                 : HButton_ExtIntInit
* DESCRIPTION       : Initialize specified as pull up for external interrupt
* RETURN                   : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_ExtIntInit (enu_pin en_pinx)
```

```
/*
* AUTHOR                   : Bassel Yasser Mahmoud
* FUNCTION                 : HButton_getPinVal
* DESCRIPTION       : Get pin status if it is high or low
* RETURN                   : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_getPinVal (enu_pin en_pinx, Uint8_t* pu8_refVal )
```

Motor

```
/*
 * Author              : Bassel Yasser Mahmoud
 * function            : HMOTOR_vidInit
 * description         : Motor Initialization as DIO dir output
 * input param         : void
 * return              : void
 * */
void HMOTOR_vidInit(void);

/*
 * Author              : Bassel Yasser Mahmoud
 * function            : HMOTOR_vidStart
 * description         : Start Motor To move
 * input param         : Copy_u8DutyCycle : PWM duty cycle
 * return              : void
 * */
void HMOTOR_vidStart (Uint8_t Copy_u8DutyCycle);

/*
 * Author              : Bassel Yasser Mahmoud
 * function            : HMOTOR_vidStop
 * description         : Motor movement stop
 * input param         : void
 * return              : void
 * */
void HMOTOR_vidStop(void);

/*
 * Author              : Bassel Yasser Mahmoud
 * function            : HMOTOR_vidTurnRight
 * description         : Motor turn direction to right
 * input param         : void
 * return              : void
 * */
void HMOTOR_vidTurnRight(void);

/*
 * Author              : Bassel Yasser Mahmoud
 * function            : HMOTOR_vidTurnLeft
 * description         : Motor turn direction to left
 * input param         : void
 * return              : void
 * */
void HMOTOR_vidTurnLeft(void);
```

HEXTINT

```
/*
 * Author           : Bassel Yasser Mahmoud
 * function         : HExtInt_enInit
 * description      : func to write integer number on lcd
 * in[1]            : enExtint : Interrupt type [INT0, INT1. INT2]
 * in[2]            : snsCtrl  : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
 * return           : void
 * */
enu_HExtIntError_t HExtInt_enInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl);
```

```
/*
 * Author           : Bassel Yasser Mahmoud
 * function         : HExtInt_enCBF
 * description      : Take pointer to function to be executed in ISR when it fires
 * input param      : pointer to function
 * return           : void
 * */
enu_HExtIntError_t HExtInt_enCBF (ptr_func pFunc);
```

```
/*
 * Author           : Bassel Yasser Mahmoud
 * function         : HExtInt_enCBFInt1
 * description      : Take pointer to function to be executed in ISR when it fires
 * input param      : pointer to function
 * return           : void
 * */
enu_HExtIntError_t HExtInt_enCBFInt1(ptr_func pFunc);
```

Ultrasonic

```
/*
 * Author            : Bassel Yasser Mahmoud
 * function          : HULTRASONIC_vidInit
 * description       : func to write integer number on lcd
 *                     Set trig pin as output
 *                      Initialize external interrupt
 * in[1]             : enExtint : Interrupt type [INT0, INT1. INT2]
 * in[2]             : snsCtrl  : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
 * return            : void
 * */
void HULTRASONIC_vidInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl);
```

```
/*
 * Author            : Bassel Yasser Mahmoud
 * function          : HULTRASONIC_vidTrigger
 * description       : Sending pulse
 * input param       : void
 * return            : void
 * */
void HULTRASONIC_vidTrigger(void);
```

```
/*
 * Author            : Bassel Yasser Mahmoud
 * function          : HULTRASONIC_u8Read
 * description       : Read distance from ultrasonic sensor
 * input param       : void
 * return            : void
 * */
Uint8_t HULTRASONIC_u8Read(void);
```

KEYPAD

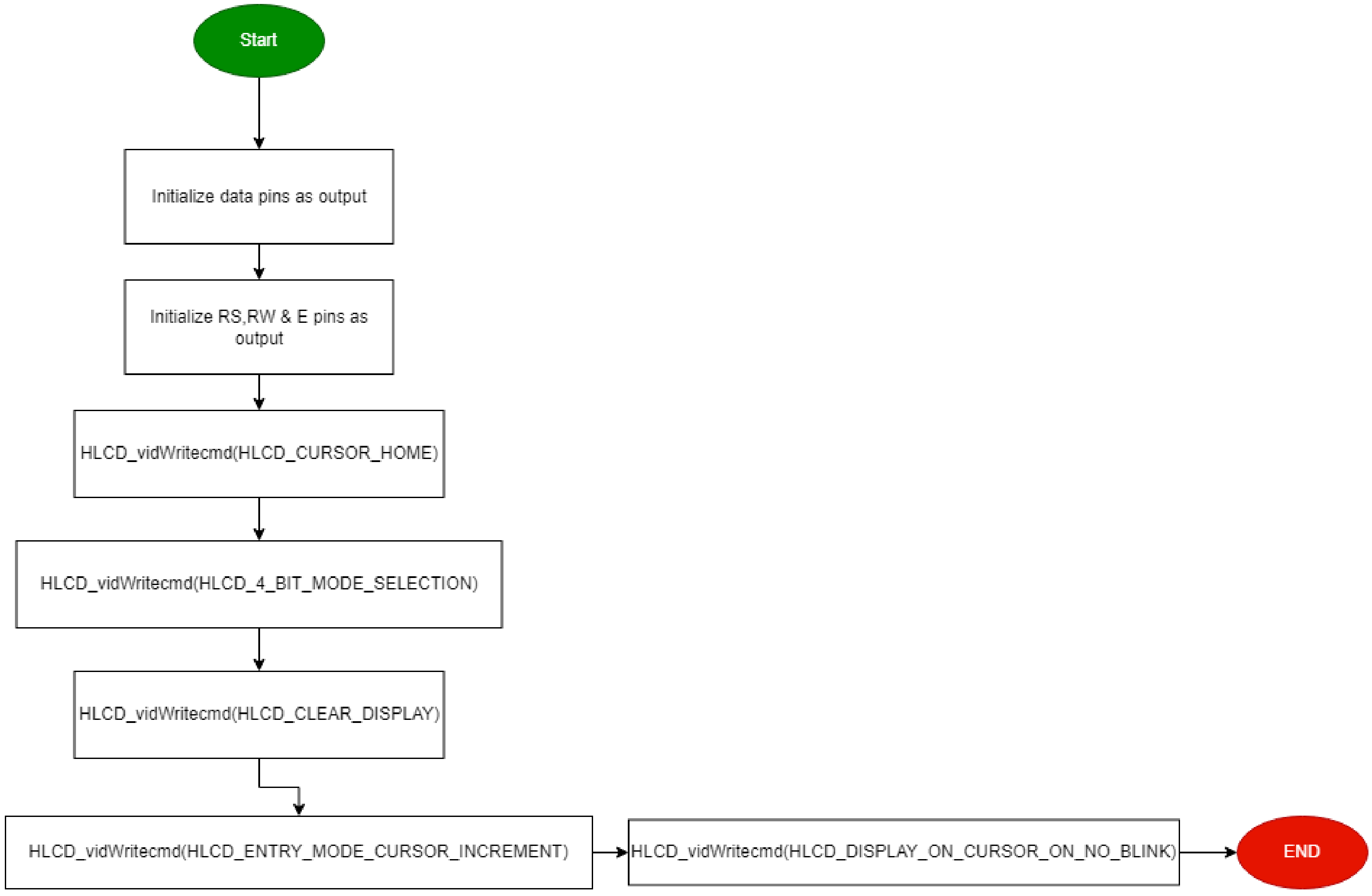
```
/*
 * Author          : Bassel Yasser Mahmoud
 * Function        : KEYPAD_vidInit_V2
 * Description     : KEYPAD Initialization
 * in[1]           : void
 * Return          : void
 */
void KEYPAD_vidInit_V2(void);
```

```
/*
 * Author          : Bassel Yasser Mahmoud
 * Function        : KEYPAD_u8GetPressed_V2
 * Description     : KEYPAG get pin status
 * in[1]           : void
 * Return          : Uint8_t {Pin Status}
 */
Uint8_t KEYPAD_u8GetPressed_V2(void);
```

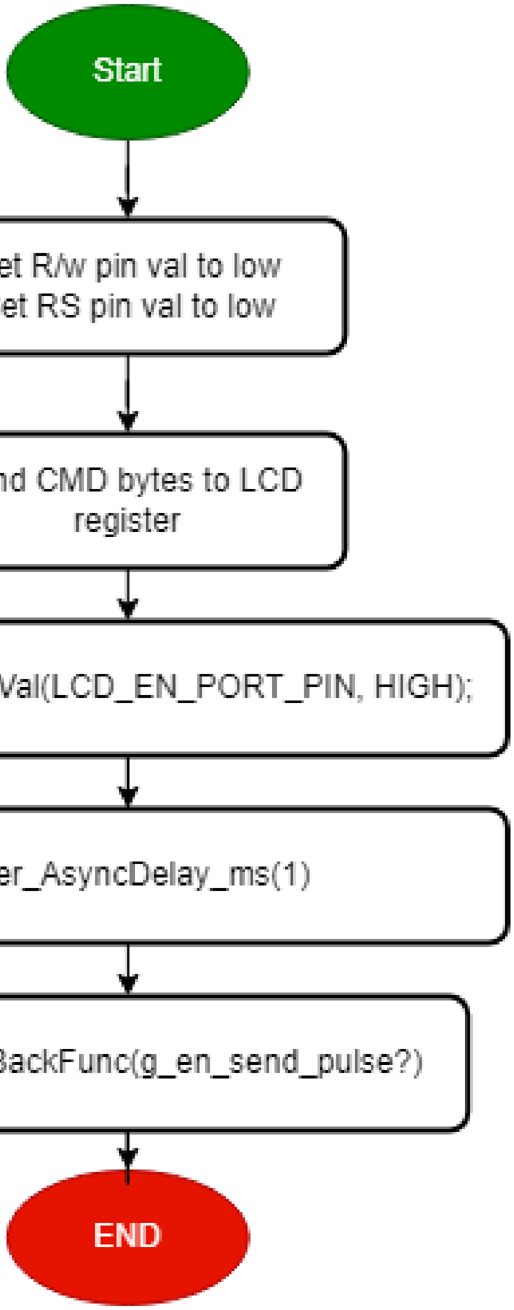
Flowchart

HLCD

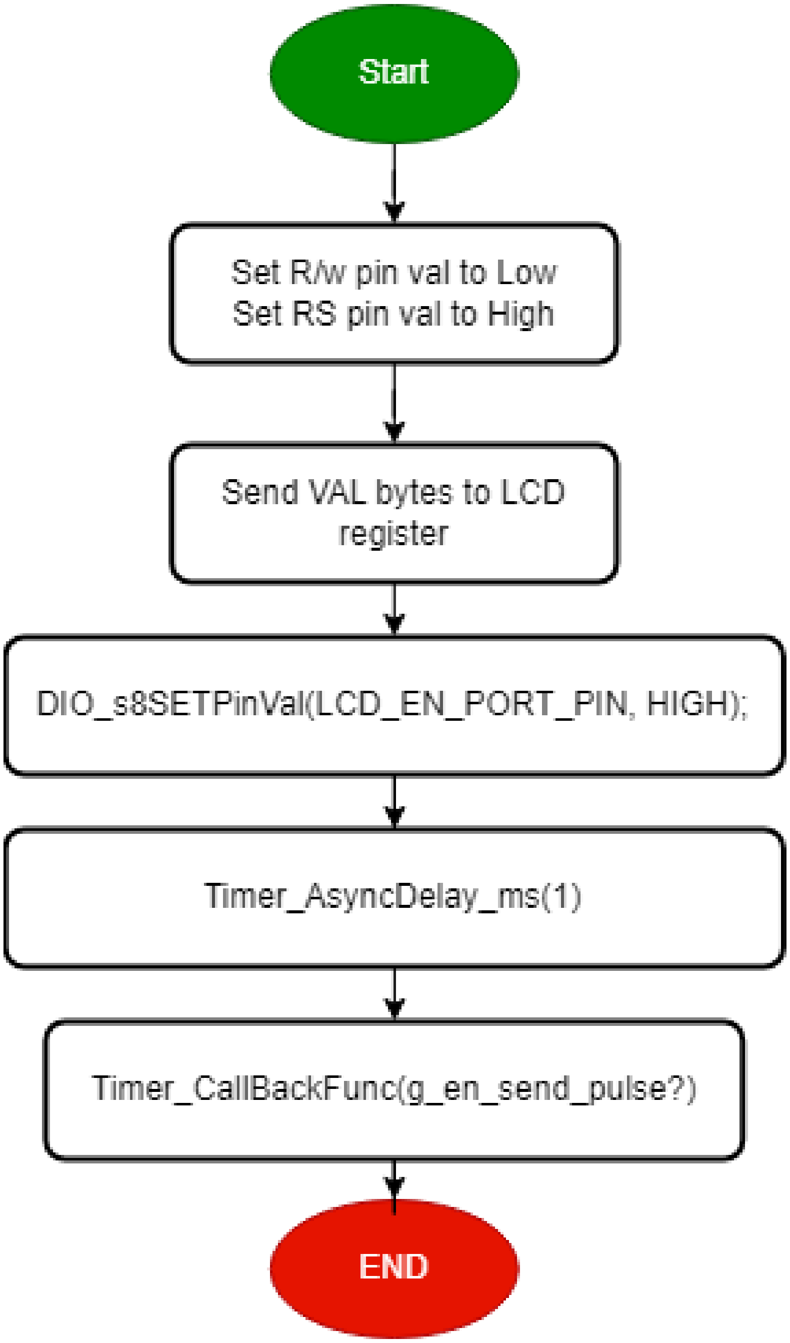
void HLCD_vidInit(void)


Start
Initialize data pins as output
Initialize RS,RW & E pins as output
HLCD_vidWritecmd(HLCD_CURSOR_HOME)
HLCD_vidWritecmd(HLCD_4_BIT_MODE_SELECTION)
HLCD_vidWritecmd(HLCD_CLEAR_DISPLAY)
HLCD_vidWritecmd(HLCD_ENTRY_MODE_CURSOR_INCREMENT)
HLCD_vidWritecmd(HLCD_DISPLAY_ON_CURSOR_ON_NO_BLINK)
END

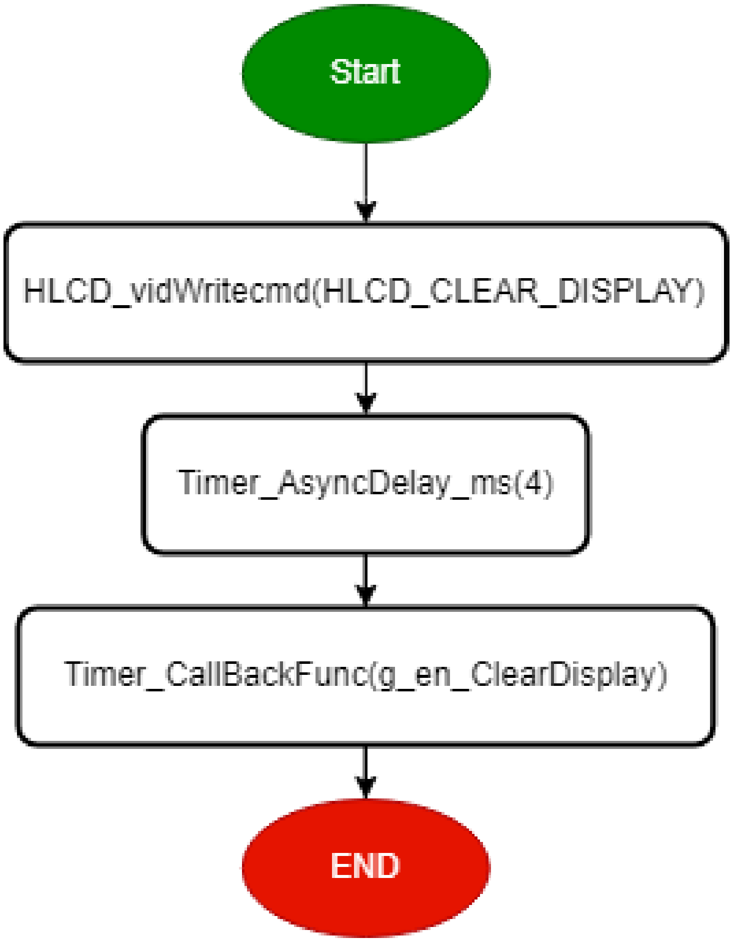
void HLCD _vidWritecmd (Uint8_t u8commandCopy)


Start
Set R/w pin val to low
Set RS pin val to low
Send CMD bytes to LCD register
DIO_s8SETPinVal(LCD_EN_PORT_PIN, HIGH);
Timer_AsyncDelay_ms(1)
Timer_CallBackFunc(g_en_send_pulse?)
END

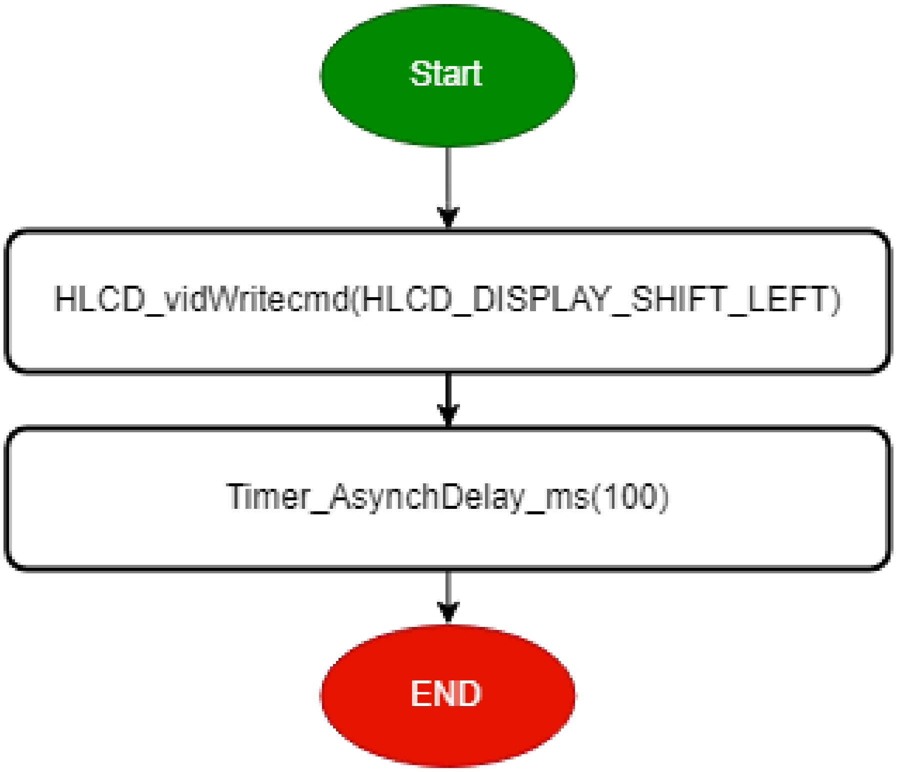
void HLCD_vidWriteChar (Uint8_t u8CharCopy)


Start
Set R/w pin val to Low
Set RS pin val to High
Send VAL bytes to LCD register
DIO_s8SETPinVal(LCD_EN_PORT_PIN, HIGH);
Timer_AsyncDelay_ms(1)
Timer_CallBackFunc(g_en_send_pulse?)
END

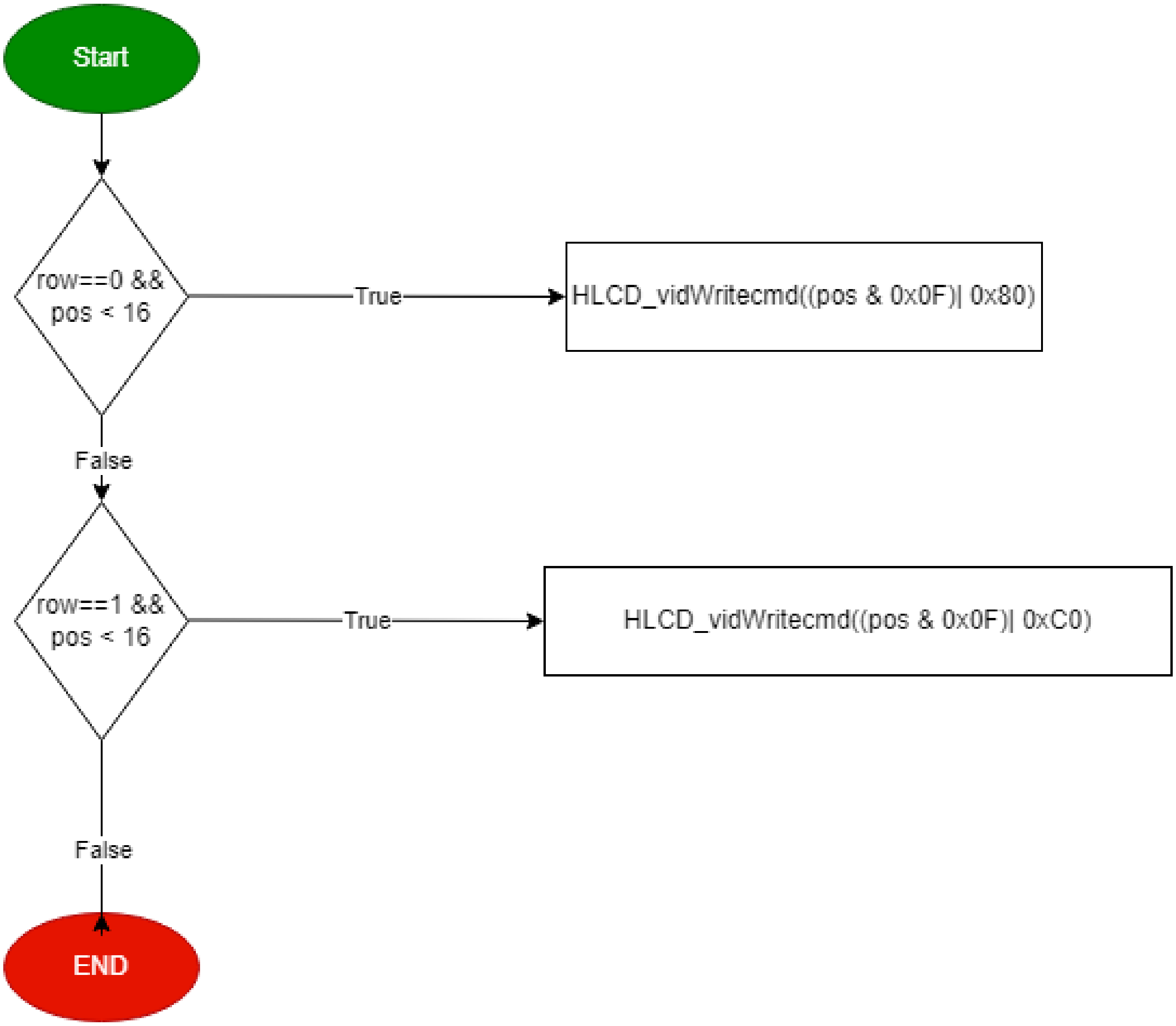
void HLCD_ClrDisplay(void)


Start
HLCD_vidWritecmd(HLCD_CLEAR_DISPLAY)
Timer_AsyncDelay_ms(4)
Timer_CallBackFunc(g_en_ClearDisplay)
END

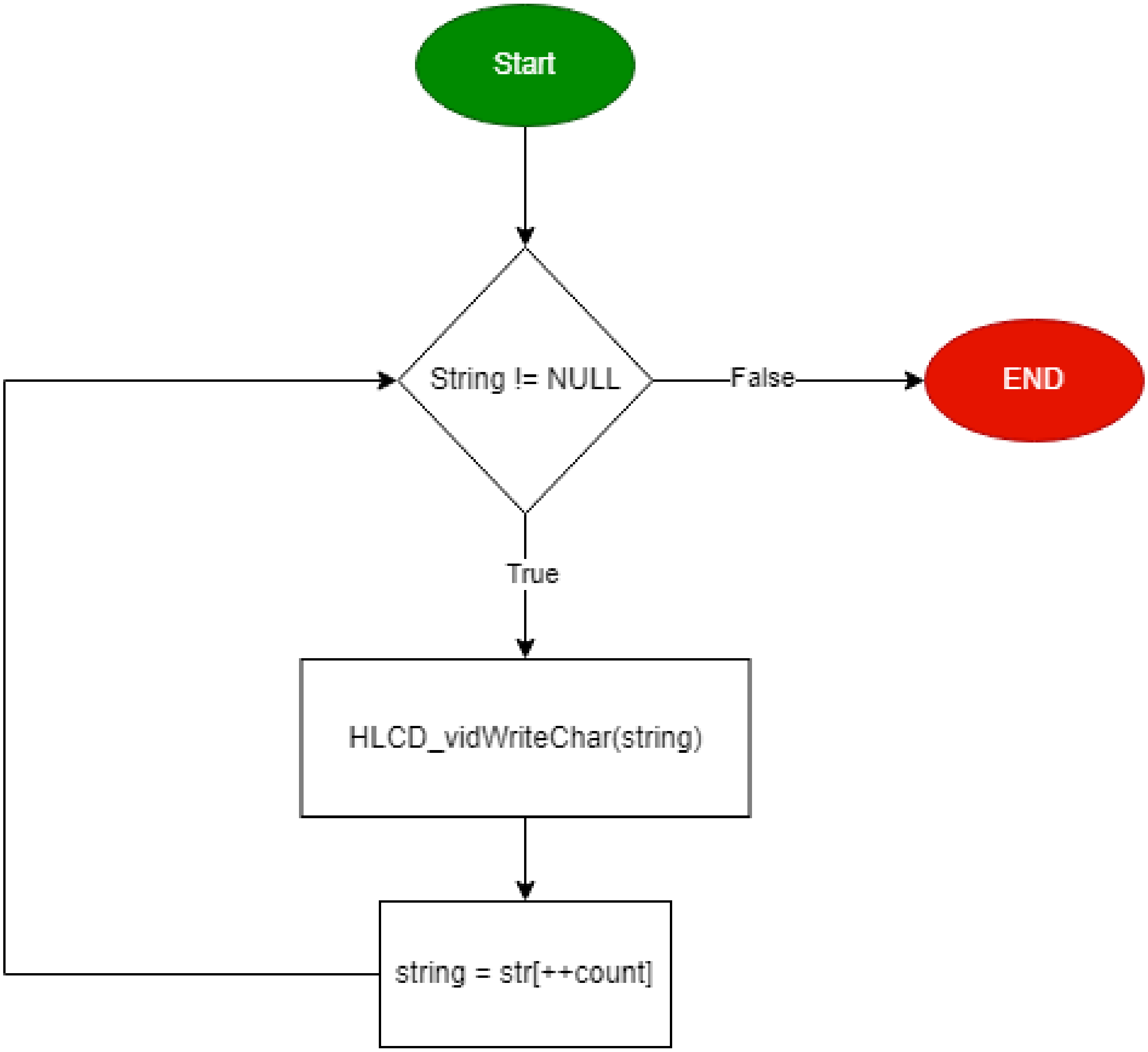
void HLCD_ShiftLeft(void)


Start
HLCD_vidWritecmd(HLCD_DISPLAY_SHIFT_LEFT)
Timer_AsynchDelay_ms(100)
END

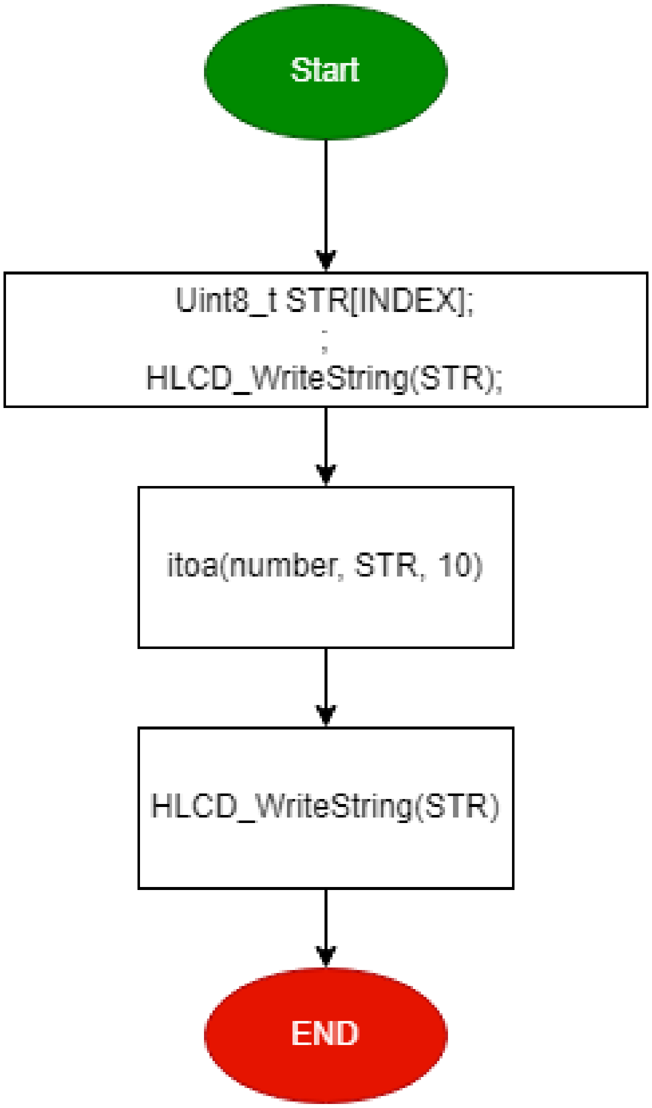
void HLCD_gotoXY (Uint8_t row, Uint8_t pos)


Start
row==0 &&
pos < 16
True
HLCD_vidWritecmd((pos & 0x0F)| 0x80)
False
row==1 &&
pos < 16
True
HLCD_vidWritecmd((pos & 0x0F)| 0xC0)
False
END

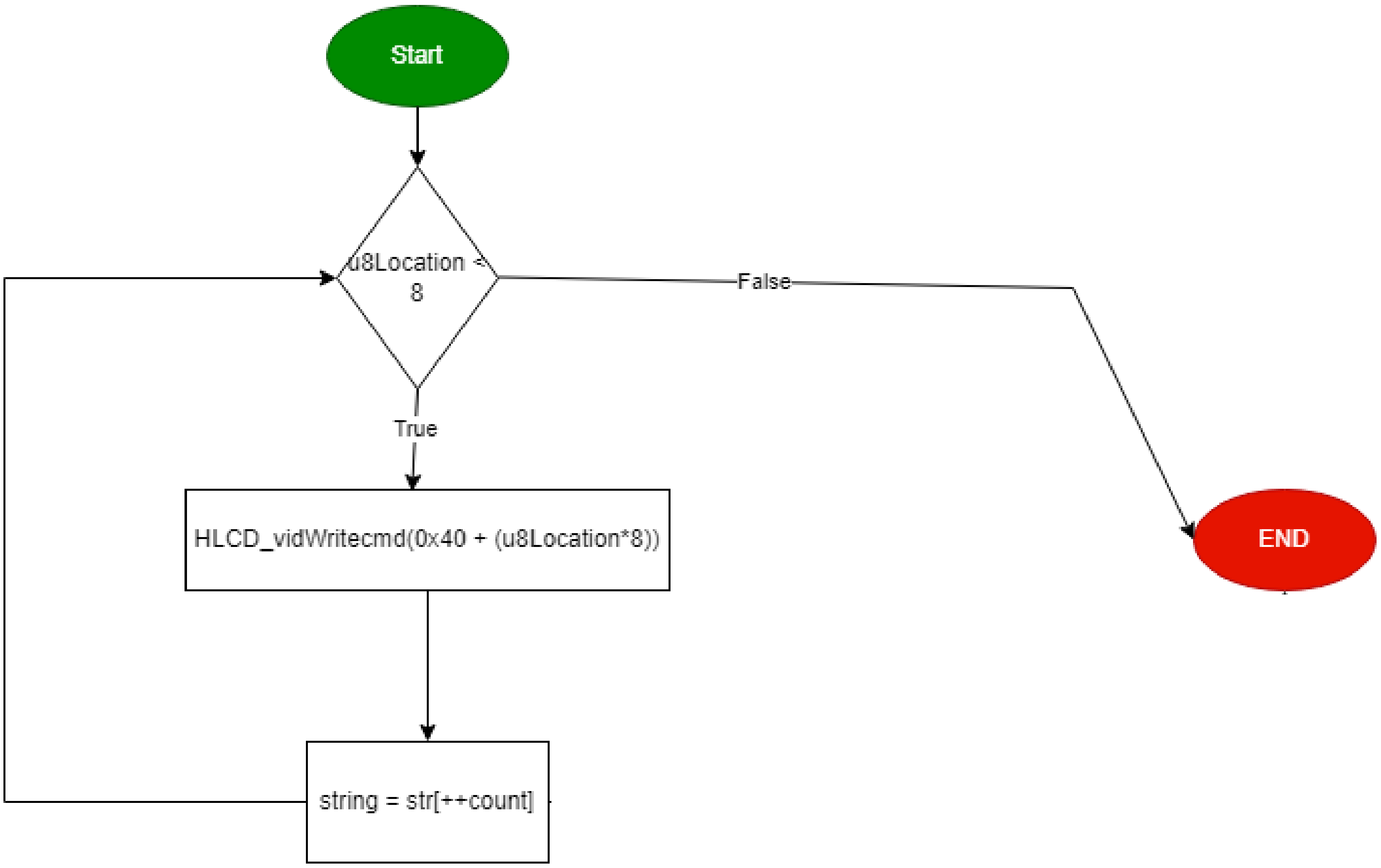
void HLCD_WriteString (Uint8_t* str)



void HLCD_WriteInt (Uint32_t number)

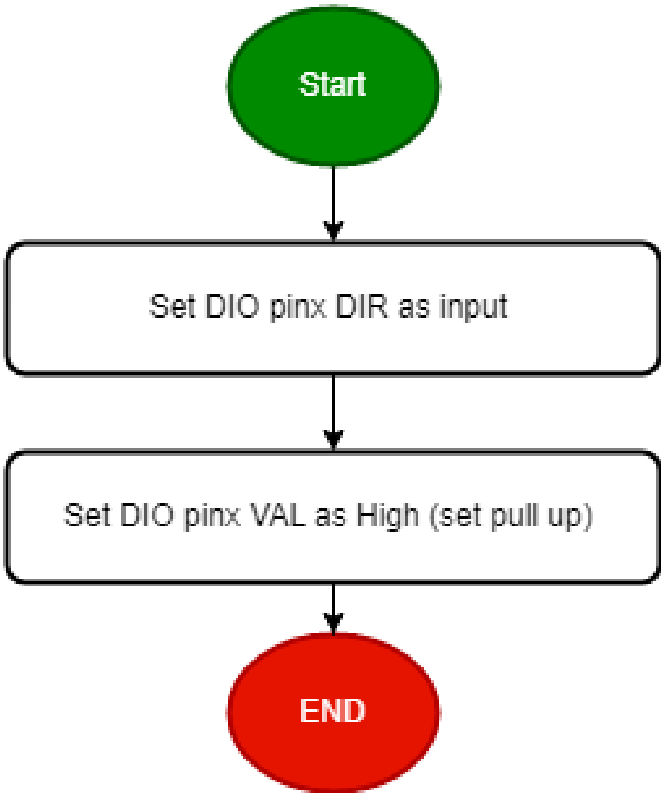


void HLCD_vidCreatCustomChar (Uint8_t* pu8custom, Uint8_t u8Location)

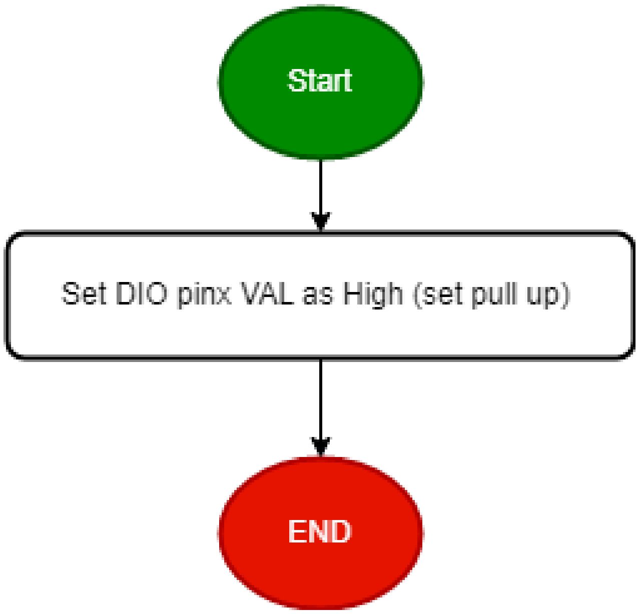


HButton

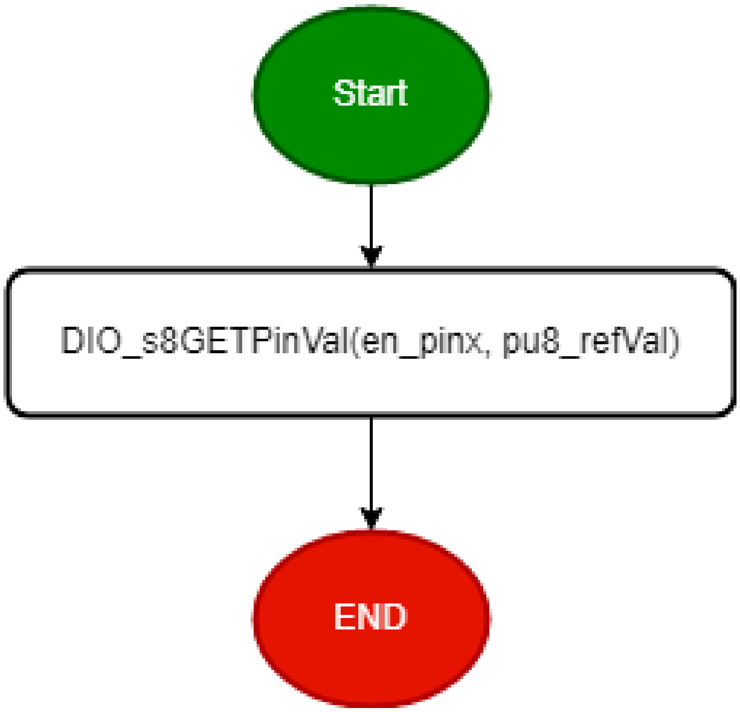
enu_buttonError_t HButton_Init (enu_pin en_pinx)


Start
Set DIO pinx DIR as input
Set DIO pinx VAL as High (set pull up)
END

enu_buttonError_t HButton_ExtIntInit (enu_pin en_pinx)

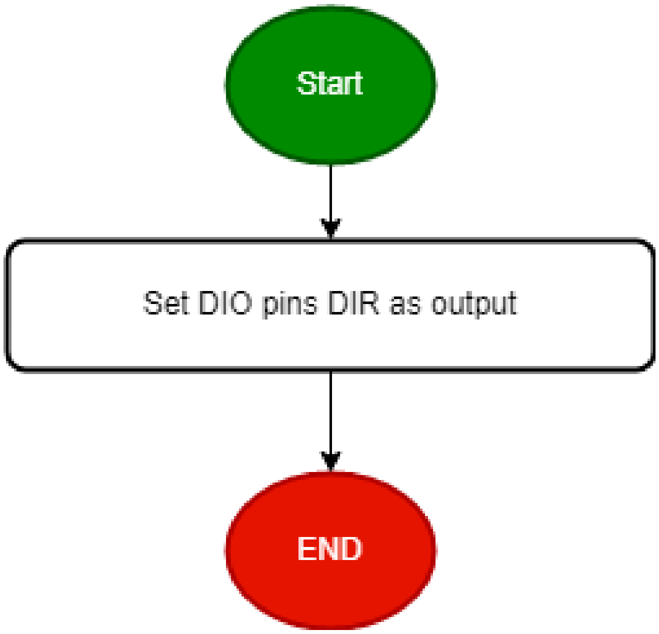


enu_buttonError_t HButton_getPinVal (enu_pin en_pinx, Uint8_t* pu8_refVal)

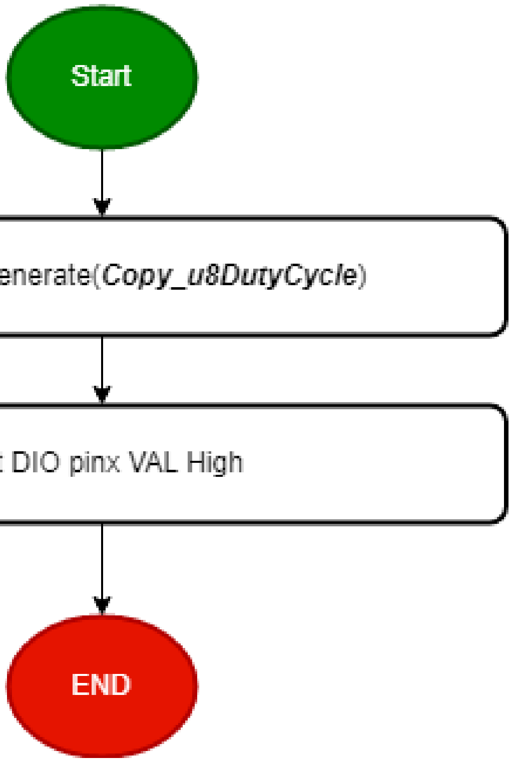


Motor

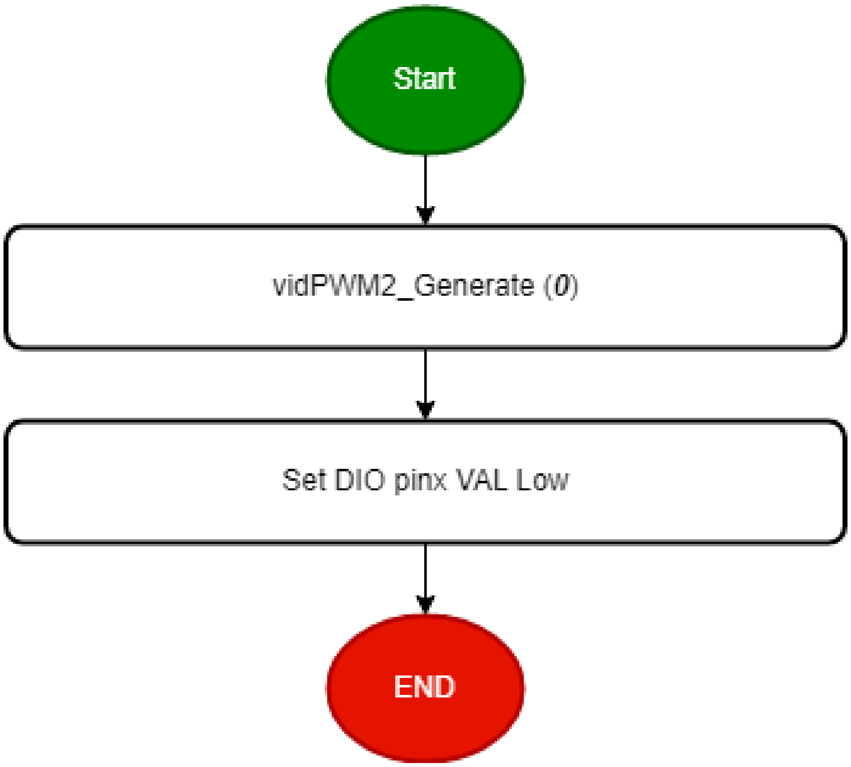
void HMOTOR_vidInit(void)



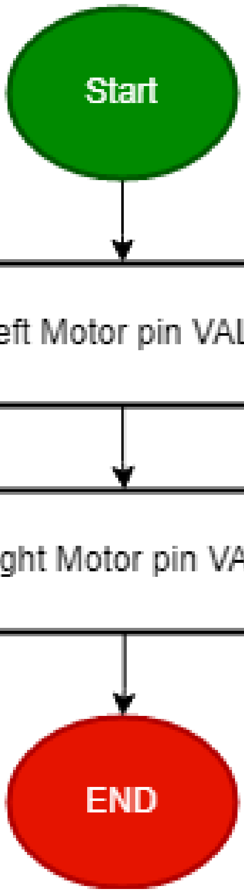
void HMOTOR_vidStart (Uint8_t Copy_u8DutyCycle)



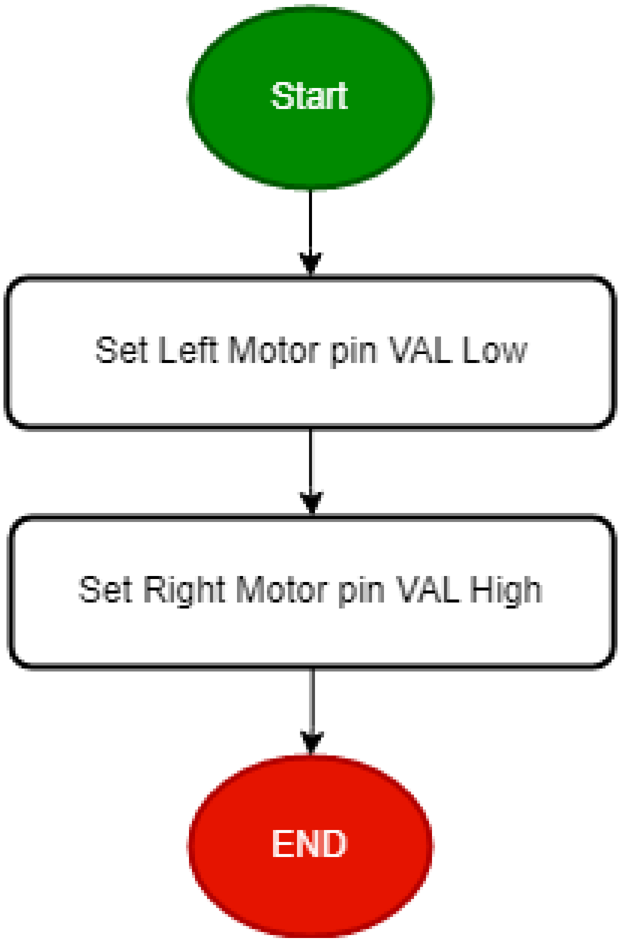
void HMOTOR_vidStop(void)



void HMOTOR_vidTurnRight(void)

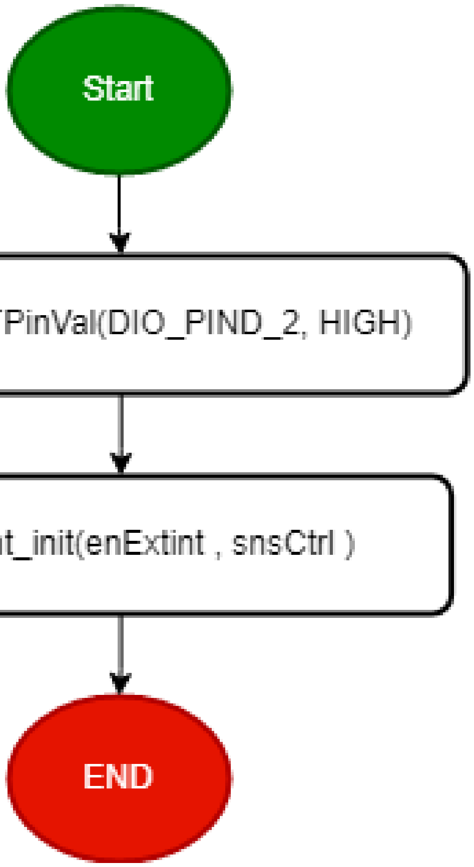


void HMOTOR_vidTurnLeft(void)

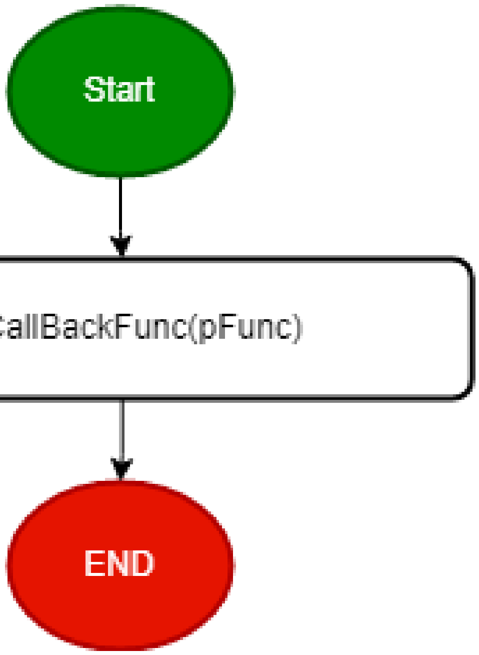


HEXTINT

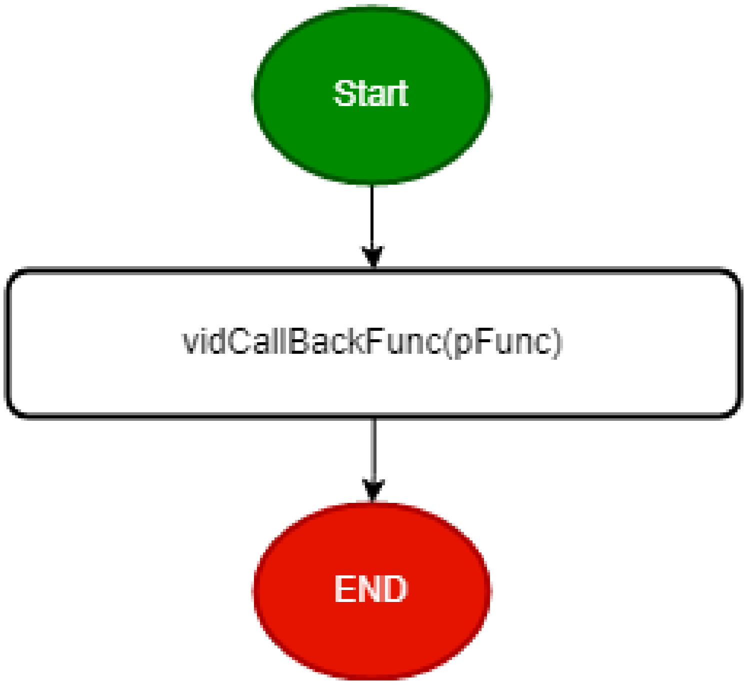
enu_HExtIntError_t HExtInt_enInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl)


Start
DIO_s8SETPinVal(DIO_PIND_2, HIGH)
vidExtInt_init(enExtint , snsCtrl)
END

enu_HExtIntError_t HExtInt_enCBF (ptr_func pFunc)

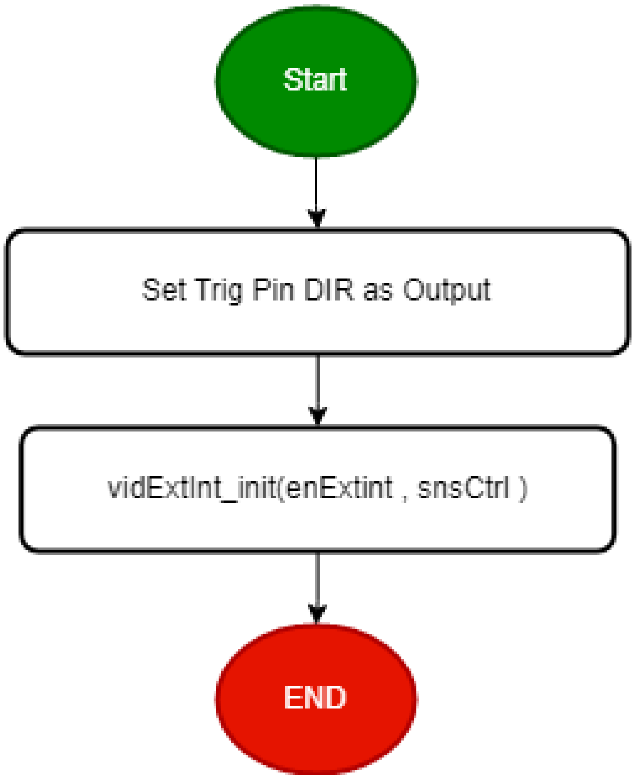


enu_HExtIntError_t HExtInt_enCBFInt1(ptr_func pFunc)

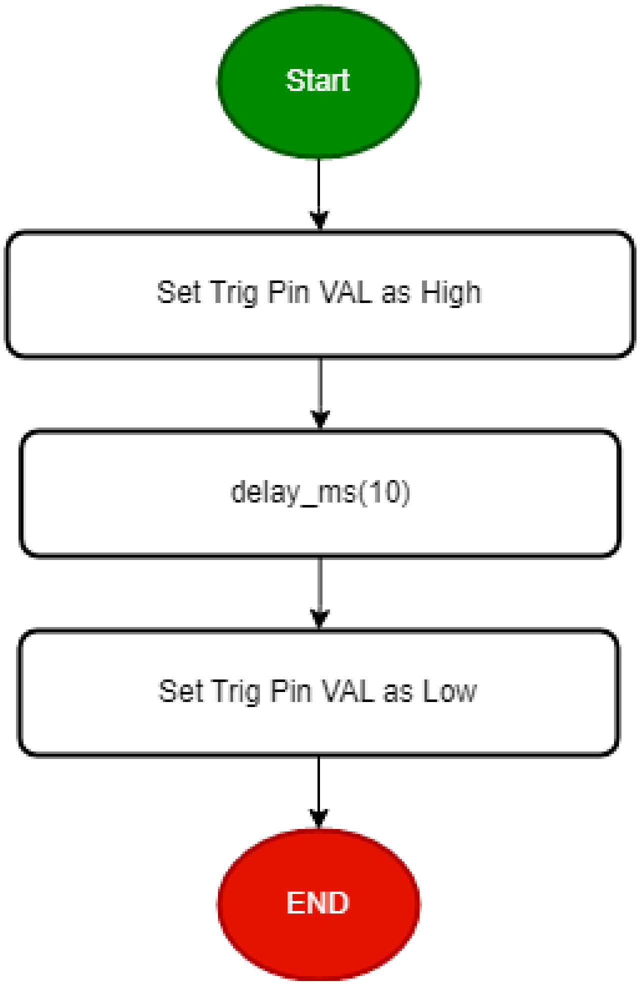


Ultrasonic

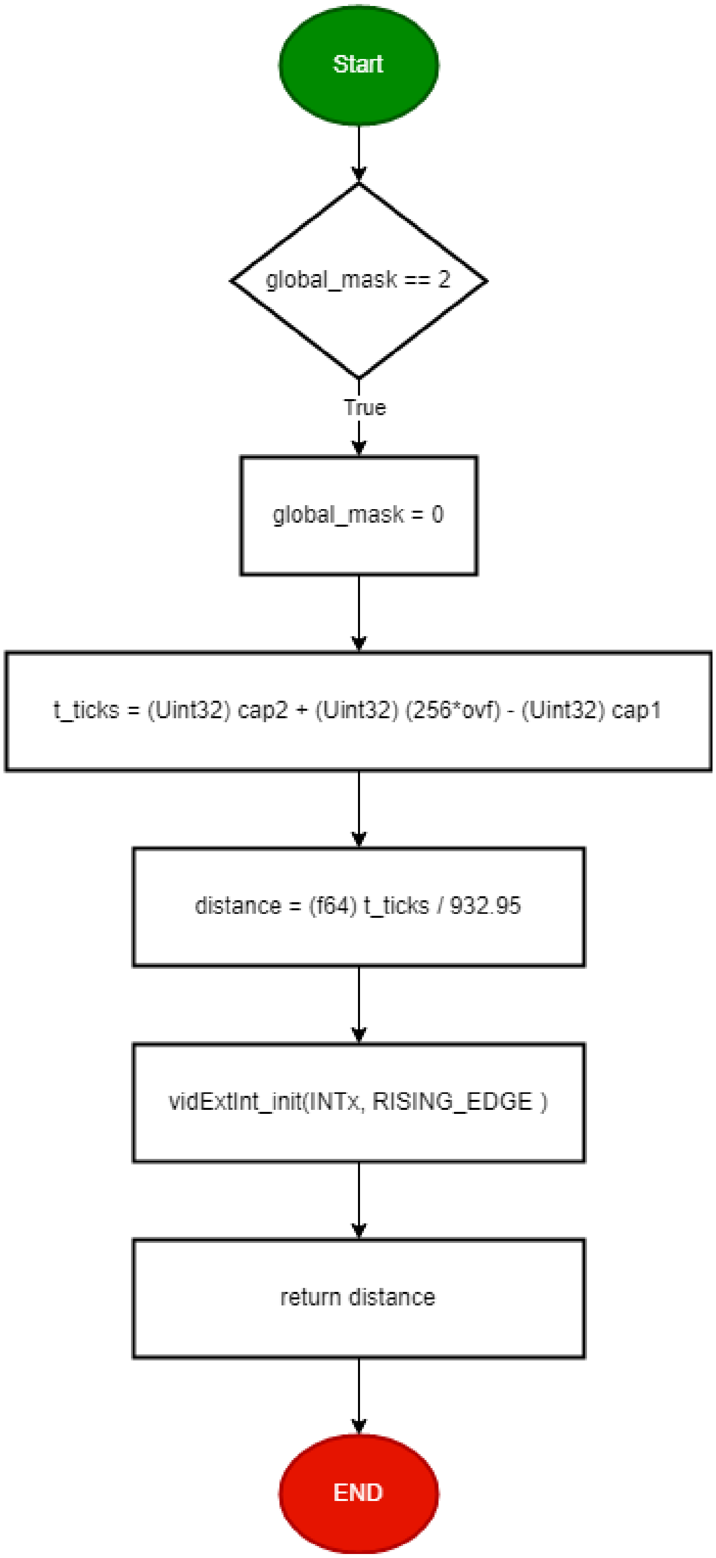
void HULTRASONIC_vidInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl)


Start
Set Trig Pin DIR as Output
vidExtInt_init(enExtint , snsCtrl)
END

void HULTRASONIC_vidTrigger(void)

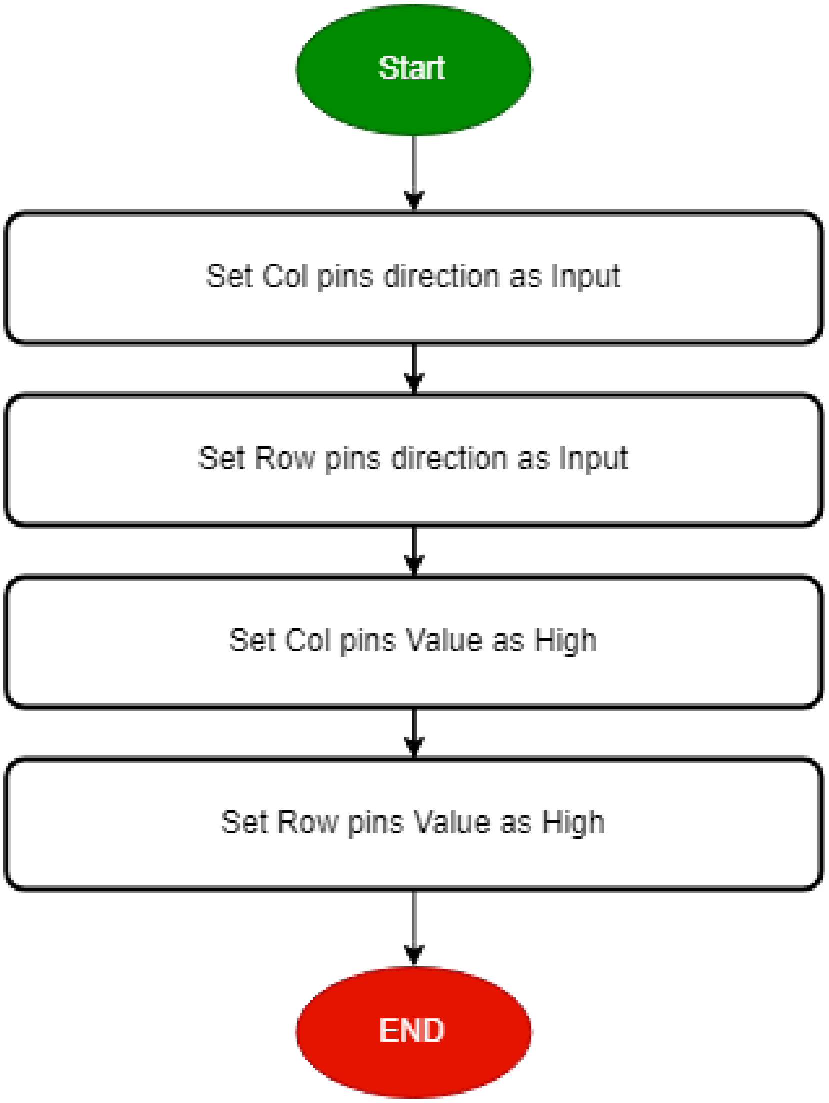


Uint8_t HULTRASONIC_u8Read(void)

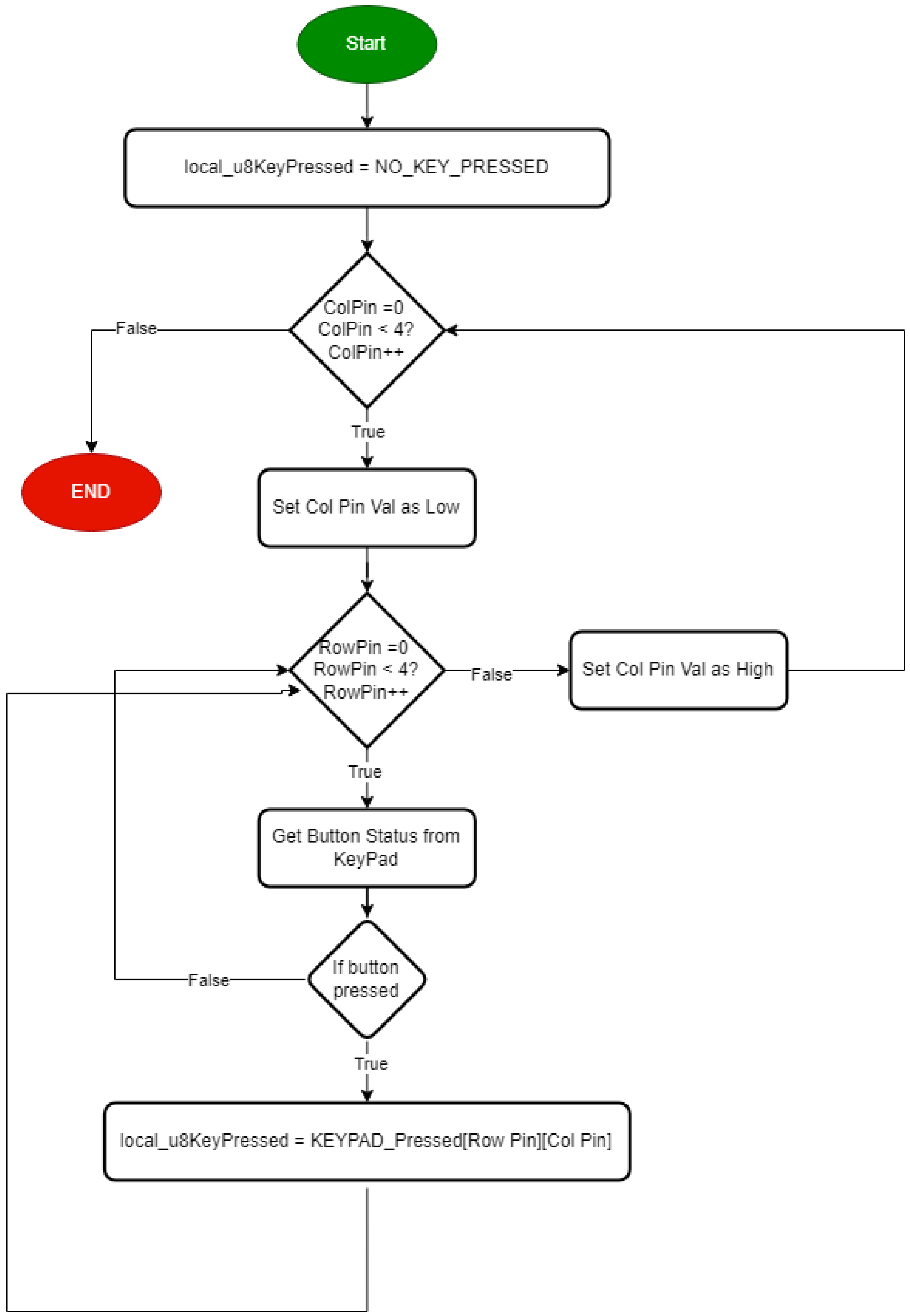

Start
global_mask == 2
True
global_mask = 0
t_ticks = (Uint32) cap2 + (Uint32) (256*ovf) - (Uint32) cap1
distance = (f64) t_ticks / 932.95
vidExtInt_init(INTx, RISING_EDGE)
return distance
END

KEYPAD

void KEYPAD_vidInit_V2(void)


Start
Set Col pins direction as Input
Set Row pins direction as Input
Set Col pins Value as High
Set Row pins Value as High
END

Uint8_t KEYPAD_u8GetPressed_V2(void)



MCAL Layer

APIs

DIO

```
  /*
   * AUTHOR         : Bassel Yasser
   * Function             : DIO_s8SETPinDir
   * Description    : Set Pin Direction
   * Arguments      :
   *                             - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
   *                             - enPortDir {INPUT , OUTPUT}
   * Return         :   Sint8_t
   */
```

Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)

```
/*
 * AUTHOR             : Bassel Yasser
 * Function           : DIO_s8SETPinVal
 * Description        : Set Pin Value
 * Arguments :
 *               - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
 *               - enPortDir {HIGH, LOW}
 * Return             :   Sint8_t
 */
```

Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_val enPortVal)

```
/*
 * AUTHOR             : Bassel Yasser
 * Function           : DIO_s8GETPinVal
 * Description        : Set Pin Value
 * Arguments :
 *                    - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
 *                    - pu8Val address of variable that u want to save value on it
 * Return             :   Sint8_t
 */
```

Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)

EXTINT

```
/*
 * Author            : Bassel Yasser Mahmoud
 * function          : vidExtInt_init
 * description       : func to write integer number on lcd
 * in[1]             : enExtint : Interrupt type [INT0, INT1. INT2]
 * in[2]             : snsCtrl  : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
 * return            : Uint8_t : return error Status  {E_INT_OK ,E_INT_NOK }
 * */
Uint8_t vidExtInt_init (enu_int_type_t, enu_sns_ctrl_t);

/*
 * Author            : Bassel Yasser Mahmoud
 * function          : vidCallBackFunc
 * description       : Take pointer to function to be executed in ISR when it fires
 * input param       : pointer to function
 * return            : void
 * */
void vidCallBackFunc (ptr_func funcCopy);


/*
 * Author            : Bassel Yasser Mahmoud
 * function          : vidCallBackFuncInt1
 * description       : Take pointer to function to be executed in ISR when it fires
 * input param       : pointer to function
 * return            : void
 * */
void vidCallBackFuncInt1(ptr_func funcCopy);
```

Timer

```
/*
 * Author          : Bassel Yasser Mahmoud
 * function        : enuTimer2_init
 * description     : Timer Initialization
 * input param     : enTimerMode { OVF_MODE, PHASE_CORRECT_PWM_MODE, CTC_MODE,
FAST_PWM_MODE}
 * return          : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t enuTimer2_init (enu_timerMode_t enTimerMode);
```

```
/*
 * Author          : Bassel Yasser Mahmoud
 * function        : u8Timer2_setPrescallar
 * description     : Timer Initialization
 * input param     : Copy_enPrescal {  TIMER_NO_CLK_SRC,
                                       TIMER_PRE_1,
                                       TIMER_PRE_8,
                                       TIMER_PRE_64,
                                       TIMER_PRE_256,
                                       TIMER_PRE_1024,
                                       TIMER_EXT_CLK_FALLING,
                                       TIMER_EXT_CLK_RISING,}
 * return          : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t u8Timer2_setPrescallar (enu_timerPrescalar_t Copy_enPrescal);
```

```
/*
 * Author          : Bassel Yasser Mahmoud
 * function        : vidTimer2_OvfIrqEnable
 * description     : Timer2 Interrupt Enable
 * input param     : void
 * return          : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t vidTimer2_OvfIrqEnable(void);
```

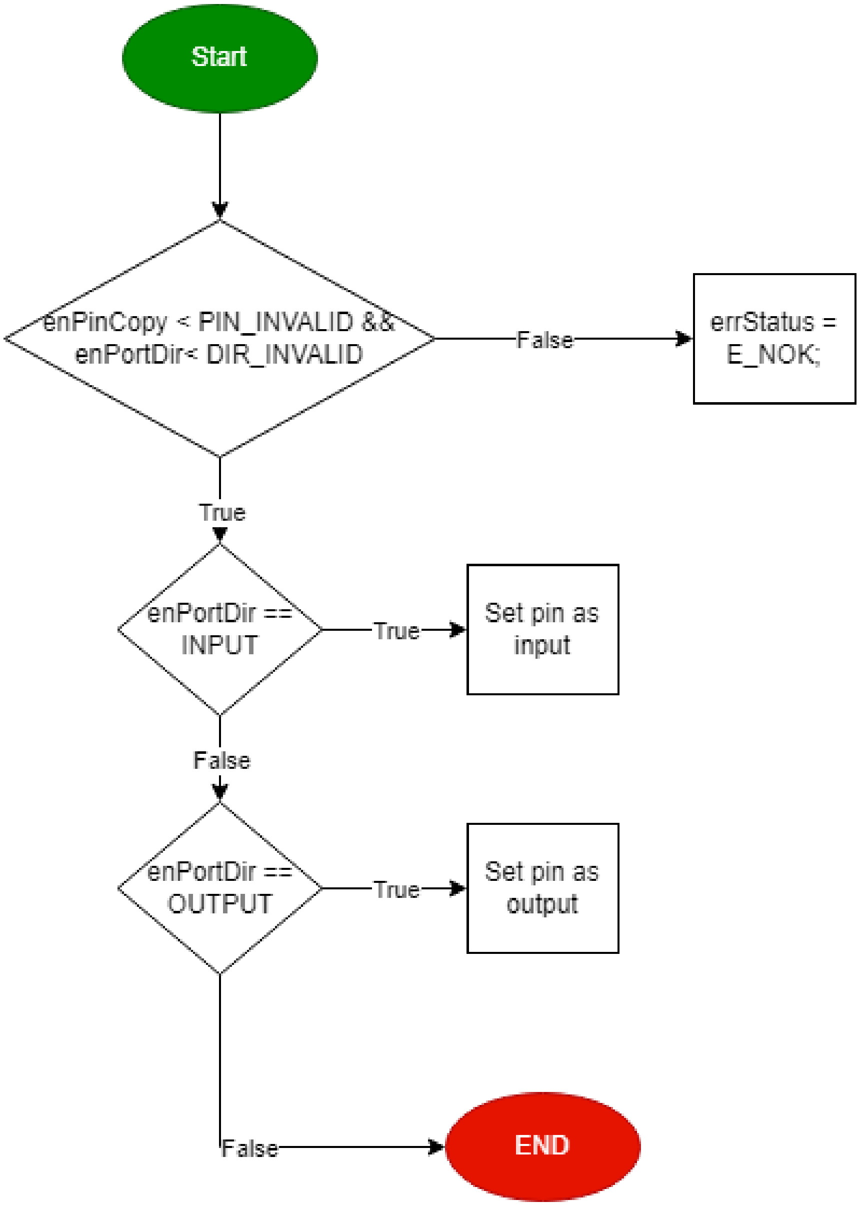
```
/*
 * Author          : Bassel Yasser Mahmoud
 * function        : vidTimer2_OvfIrqDisable
 * description     : Timer2 Interrupt Disable
 * input param     : void
 * return          : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t vidTimer2_OvfIrqDisable(void);
```

```
/*
 * Author             : Bassel Yasser Mahmoud
 * function           : vidTimer2_start
 * description        : Timer2 Start Counting
 * input param        : void
 * return             : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t vidTimer2_start(void);
/*
 * Author             : Bassel Yasser Mahmoud
 * function           : vidTimer2_stop
 * description        : Timer2 Stop
 * input param        : void
 * return             : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t vidTimer2_stop(void);

/*
 * Author             : Bassel Yasser Mahmoud
 * function           : u8Timer2_setTime_ms
 * description        : Set time in ms
 * input param        : u32_time_ms
 * return             : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t u8Timer2_setTime_ms(Uint32_t u32_time_ms);

/*
 * Author                          : Bassel Yasser Mahmoud
 * Function Name             : Timer2_enuFastPWM0Init
 * Function Description : Set PWM Mode
 * Arguments                 : copy_enPWMMode
{TIMER2_PWM_NORMAL,TIMER2_PWM_CLR_ON_CMP,TIMER2_PWM_SET_ON_CMP,..... }
 * Return                          : enu_timer2Status_t {TIMER2_OK or TIMER2_NOK}
 */enu_timer2Status_t Timer2_enuFastPWMInit (enu_pwmMode_t copy_enPWMMode)

/*
 * Author             : Bassel Yasser Mahmoud
 * function           : vidPWM2_Generate
 * description        : PWM generation
 * input param        : Copy_u8DutyCycle : Take duty cycle {0 ~ 100}
 * return             : enu_timerStatus_t {TIMER_OK, TIMER_NOK}
 * */
enu_timerStatus_t vidPWM2_Generate (Uint8_t Copy_u8DutyCycle);

/*
 * Author             : Bassel Yasser Mahmoud
 * function           : vidTimer2_setcbf_OVF
 * description        : Take pointer to function to be executed in ISR when it fires
 * input param        : cbf  : call back function
 * return             : void
 * */
void vidTimer2_setcbf_OVF (cbf_t cbf);
```

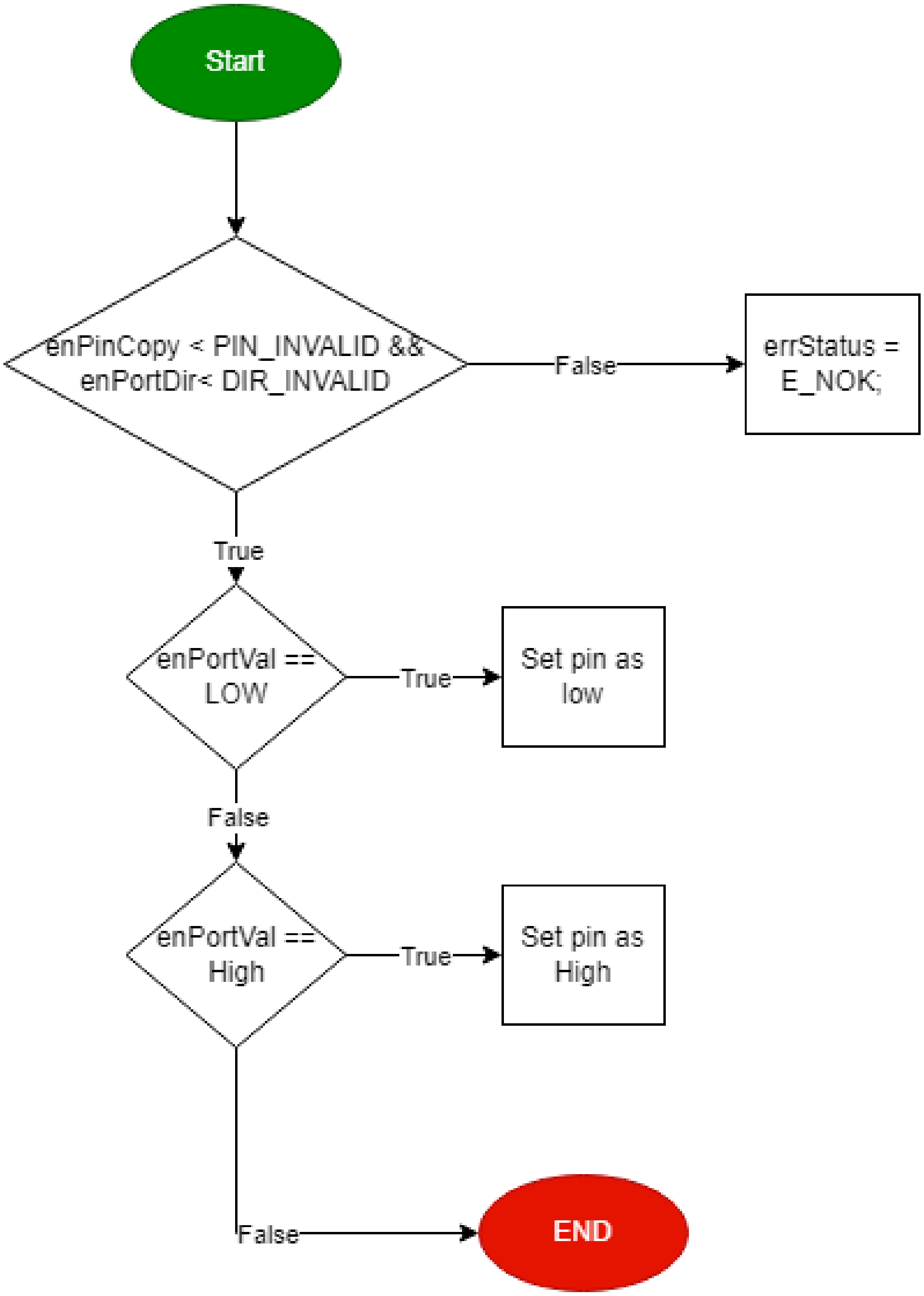
Flowchart

DIO

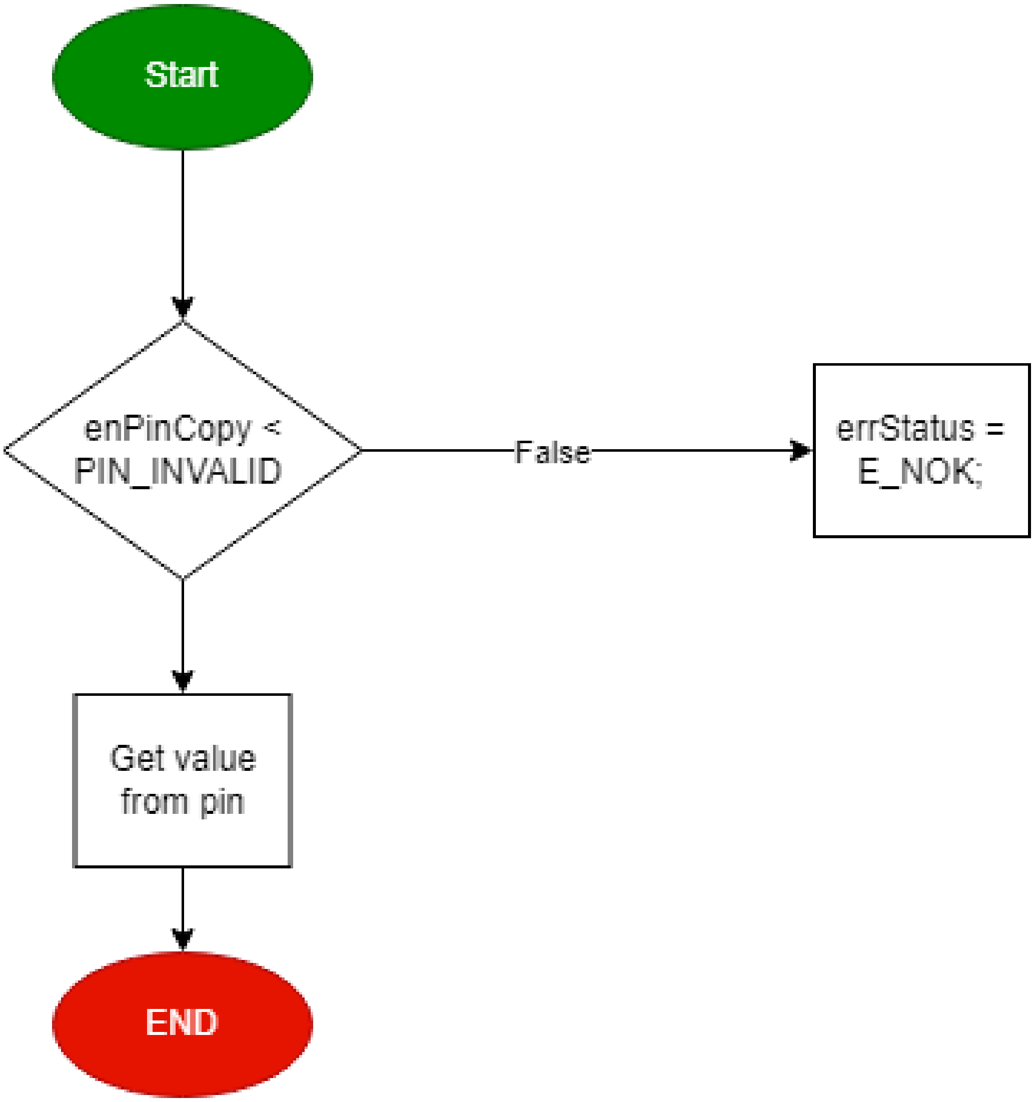
Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)



Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_val enPortVal)

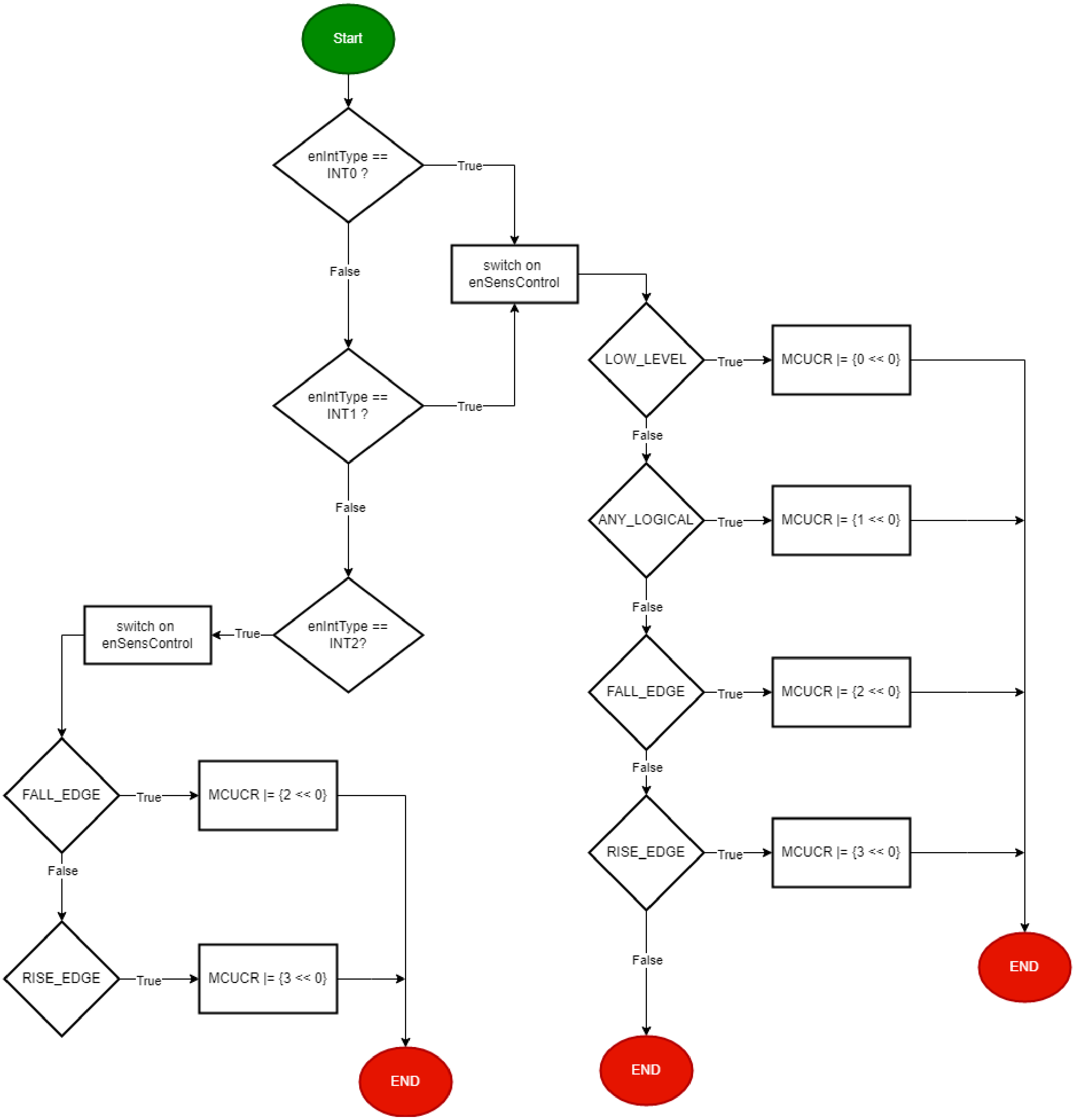


Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)

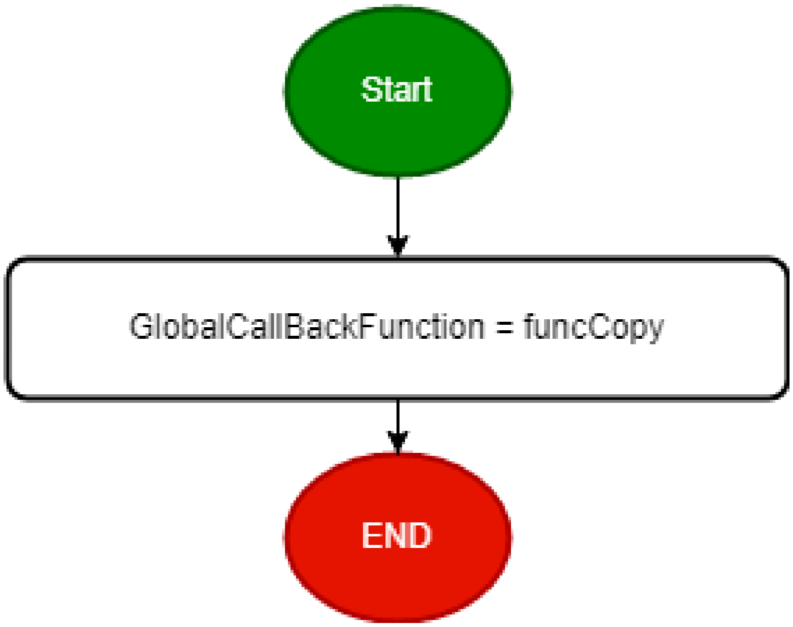


EXTINT

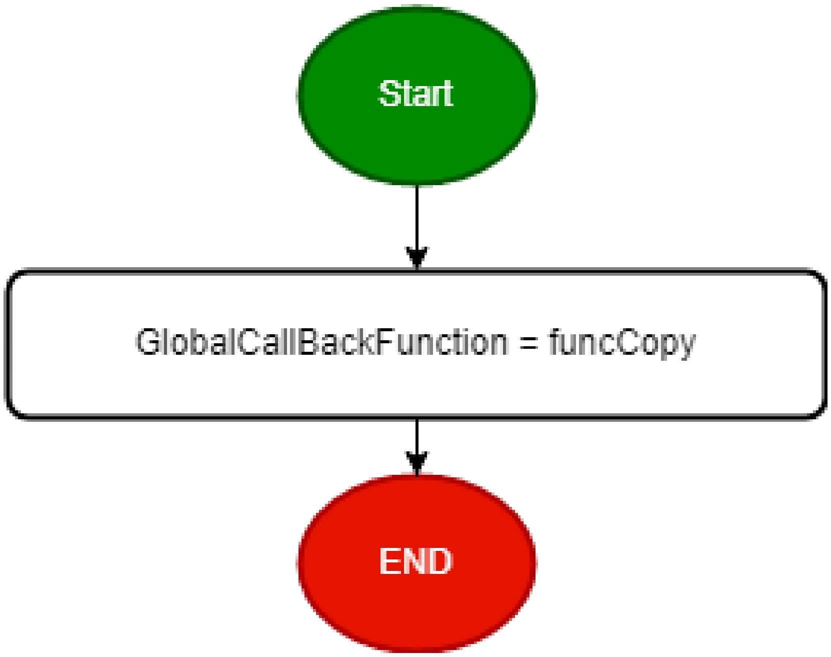
Uint8_t vidExtInt_init (enu_int_type_t, enu_sns_ctrl_t)



void vidCallBackFunc (ptr_func funcCopy)

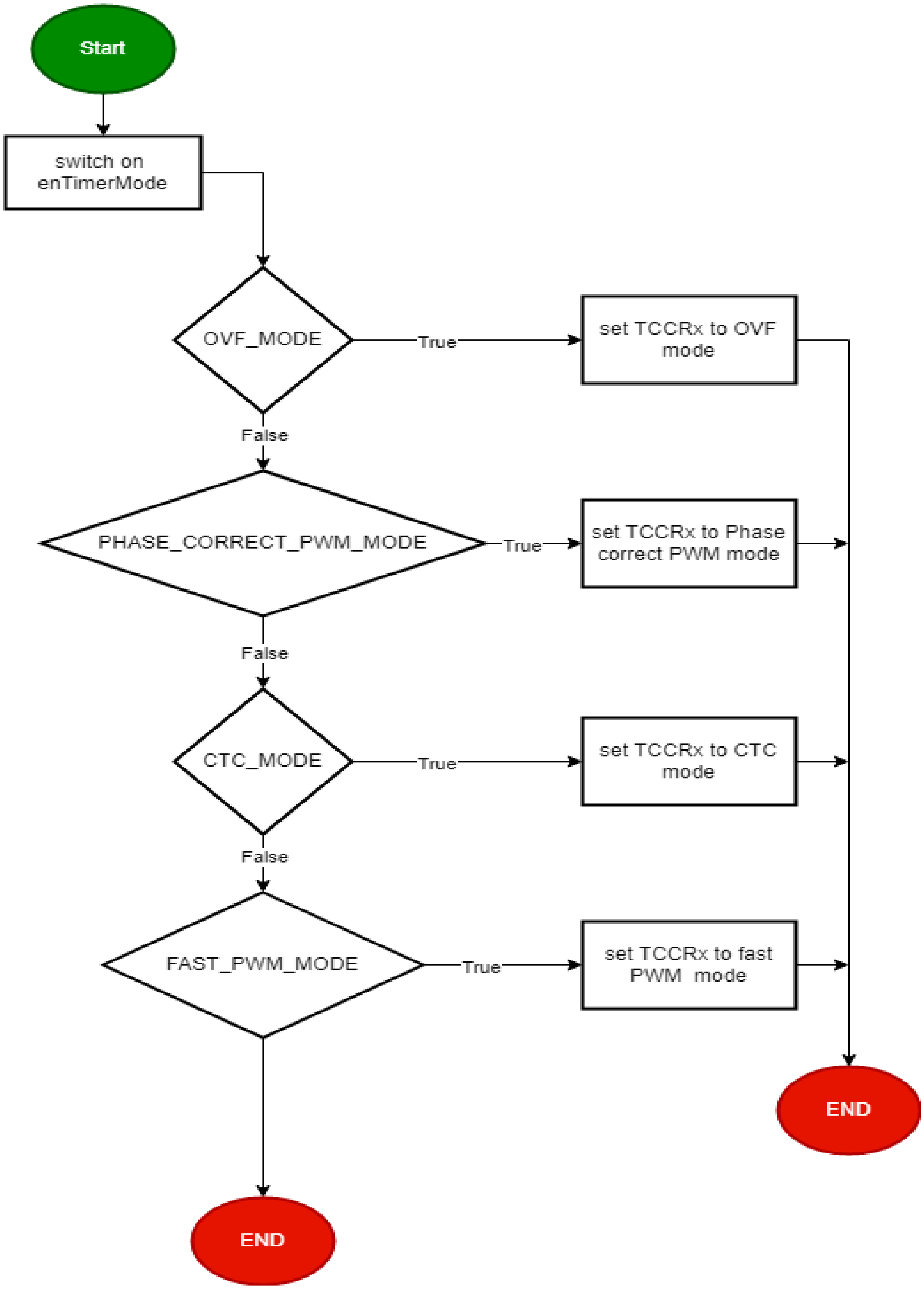


```
void vidCallBackFuncInt1(ptr_func funcCopy);
```

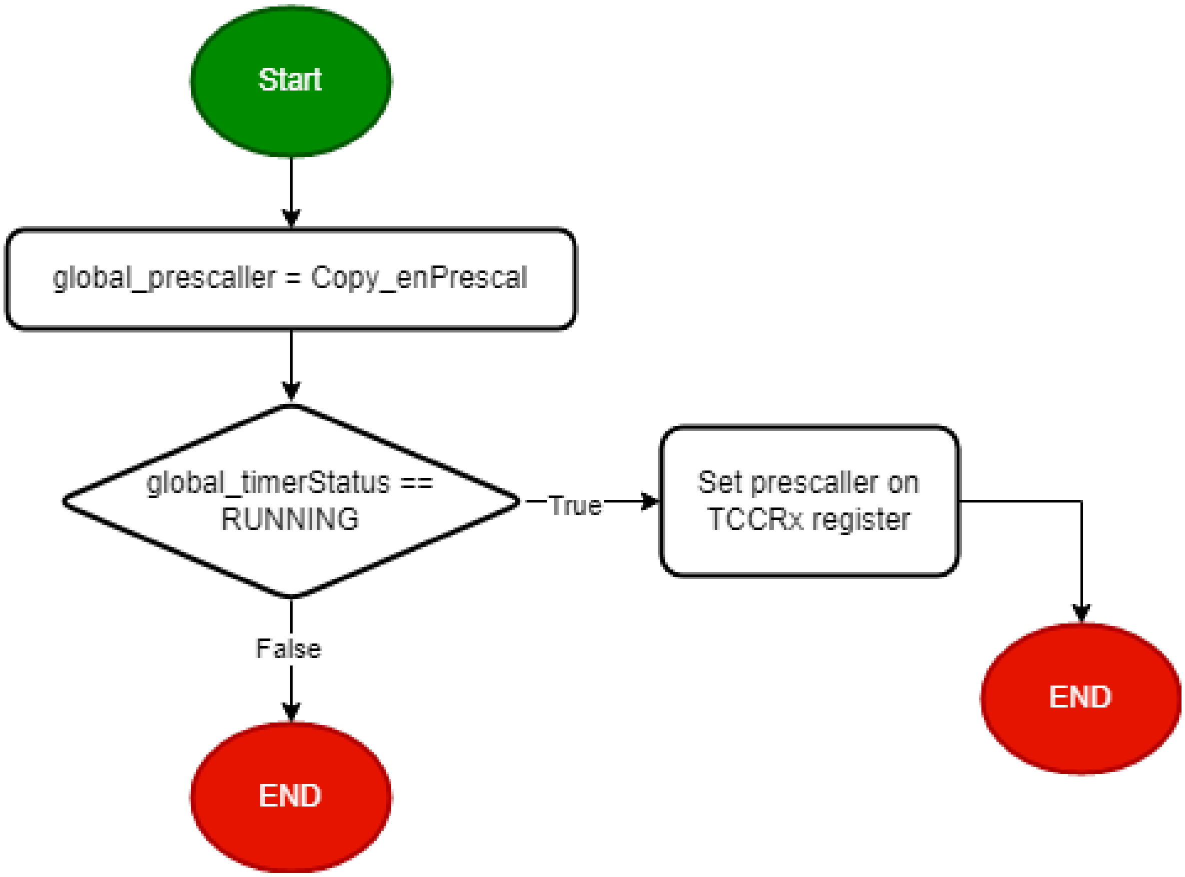


Timer

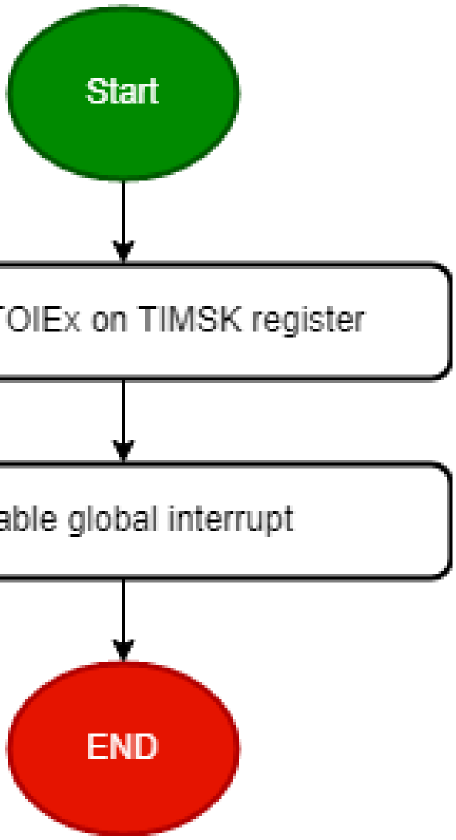
enu_timerStatus_t enuTimer2_init (enu_timerMode_t enTimerMode)



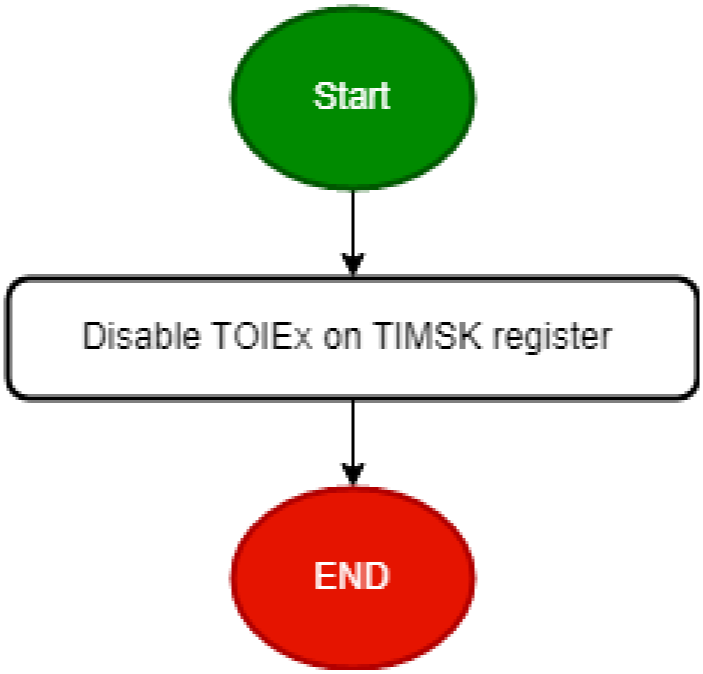
enu_timerStatus_t u8Timer2_setPrescallar (enu_timerPrescalar_t Copy_enPrescal)


Start
global_prescaller = Copy_enPrescal
global_timerStatus == RUNNING
True
Set prescaller on TCCRx register
False
END
END

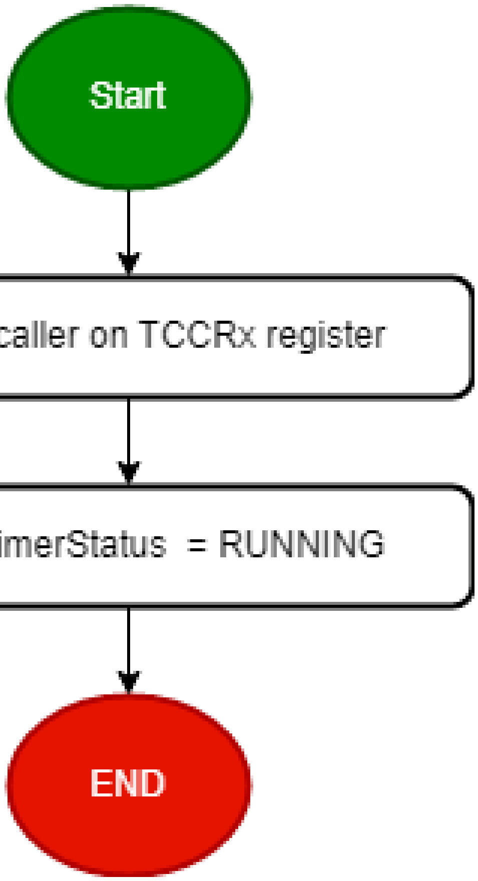
enu_timerStatus_t vidTimer2_OvfIrqEnable(void)



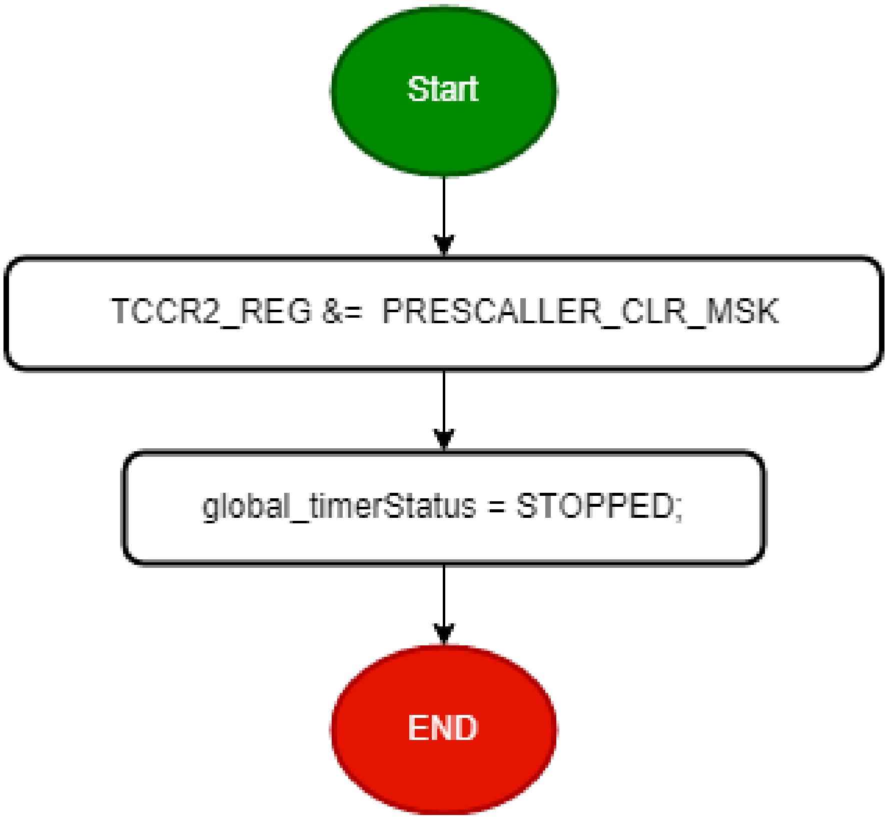
enu_timerStatus_t vidTimer2_OvfIrqDisable(void)



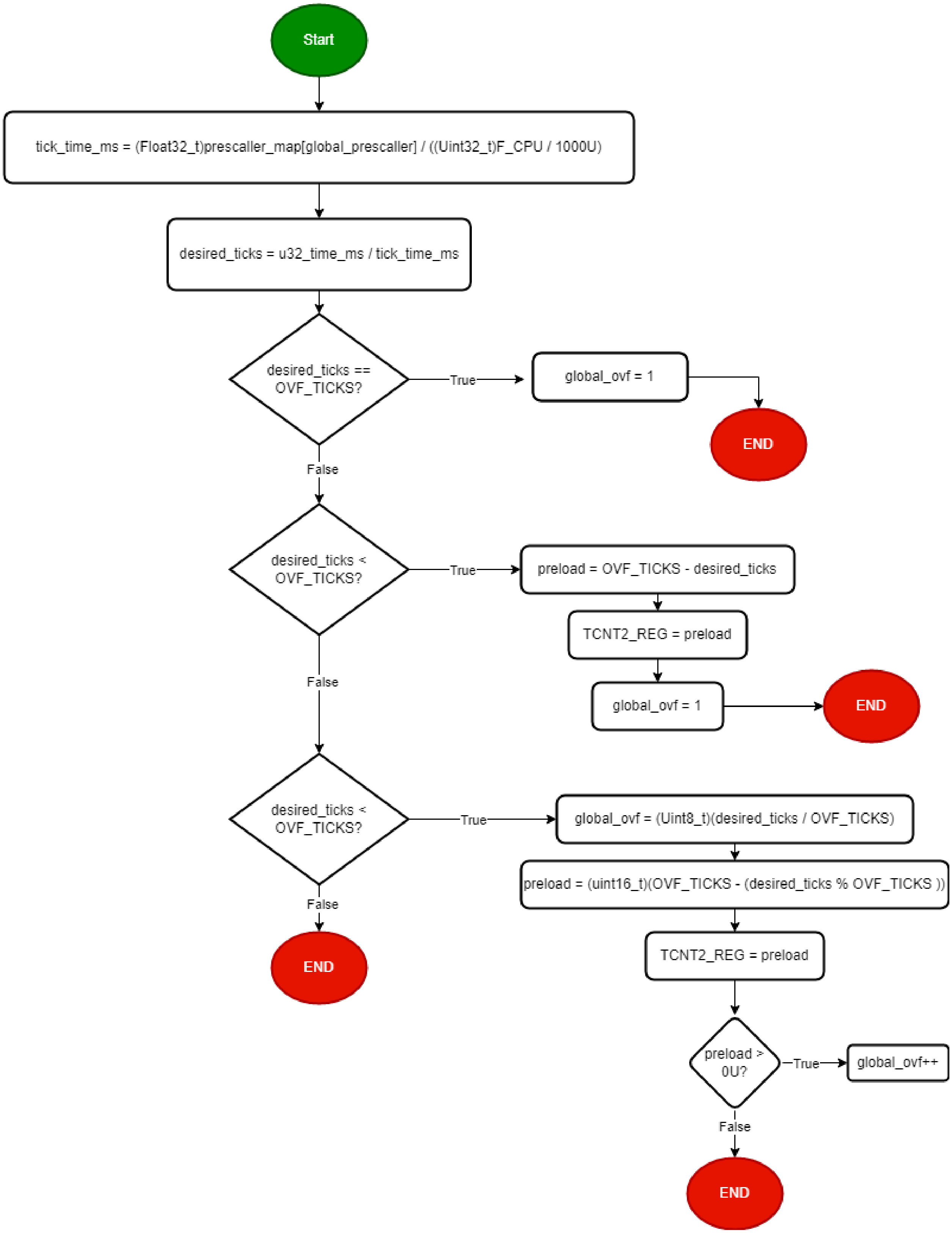
enu_timerStatus_t vidTimer2_start(void)


Start
Set prescaller on TCCRx register
global_timerStatus = RUNNING
END

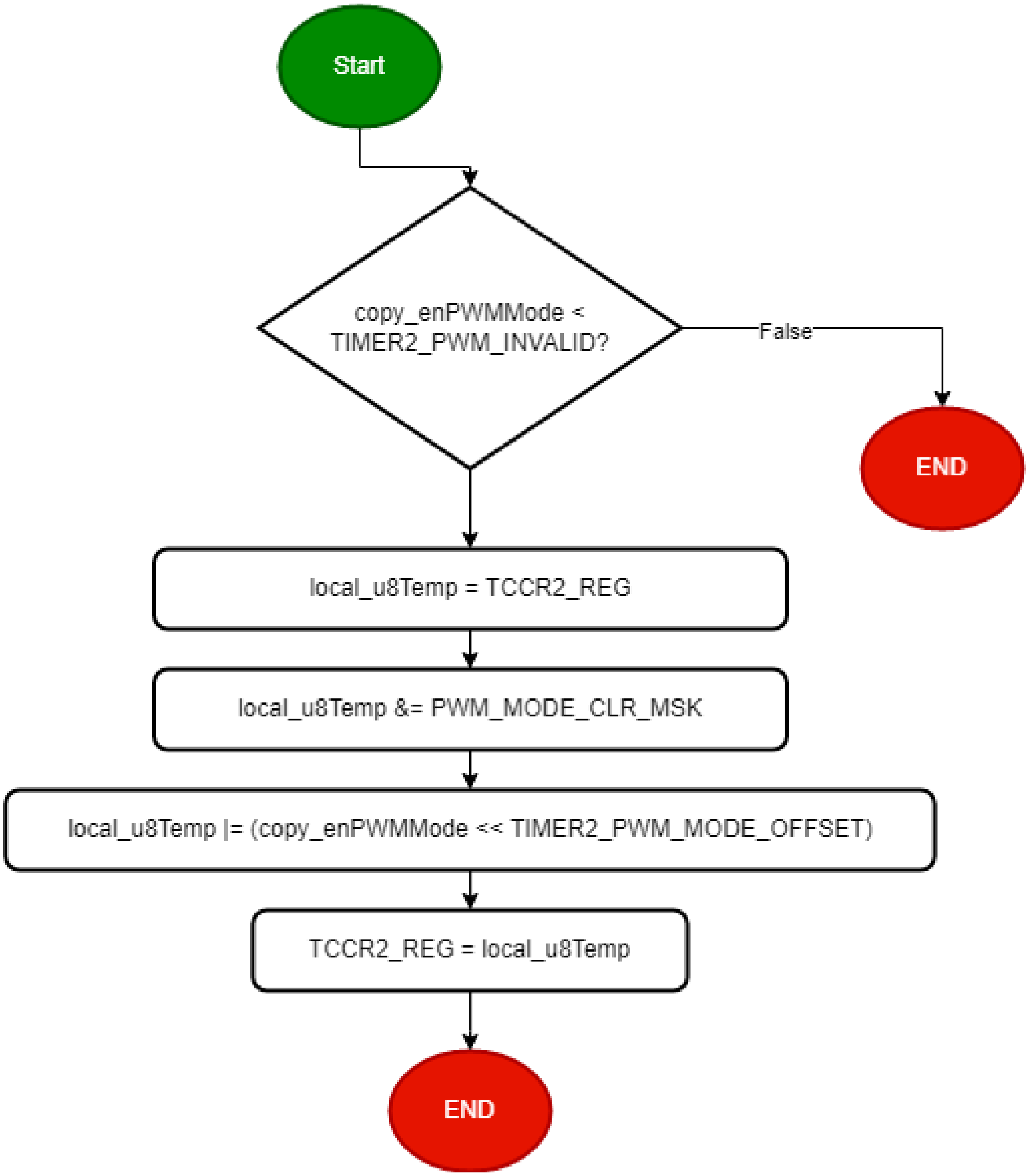
enu_timerStatus_t vidTimer2_stop(void)


Start
TCCR2_REG &= PRESCALLER_CLR_MSK
global_timerStatus = STOPPED;
END

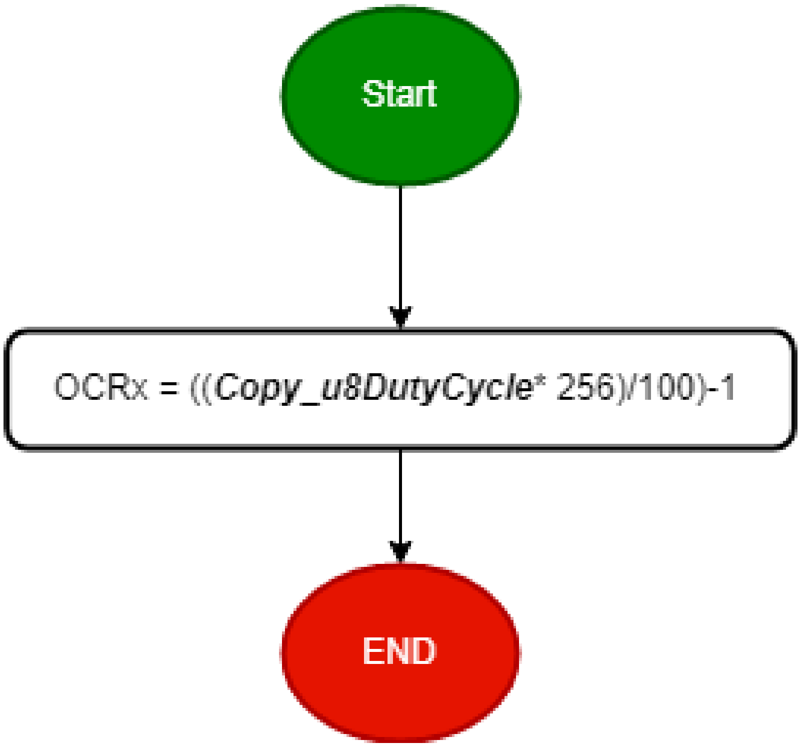
enu_timerStatus_t u8Timer2_setTime_ms (Uint32_t u32_time_ms)


Start
tick_time_ms = (Float32_t)prescaller_map[global_prescaller] / ((Uint32_t)F_CPU / 1000U)
desired_ticks = u32_time_ms / tick_time_ms
desired_ticks == OVF_TICKS?
True
global_ovf = 1
END
False
desired_ticks < OVF_TICKS?
True
preload = OVF_TICKS - desired_ticks
TCNT2_REG = preload
global_ovf = 1
END
False
desired_ticks < OVF_TICKS?
True
global_ovf = (Uint8_t)(desired_ticks / OVF_TICKS)
preload = (uint16_t)(OVF_TICKS - (desired_ticks % OVF_TICKS))
TCNT2_REG = preload
preload > 0U?
True
global_ovf++
False
END
False
END

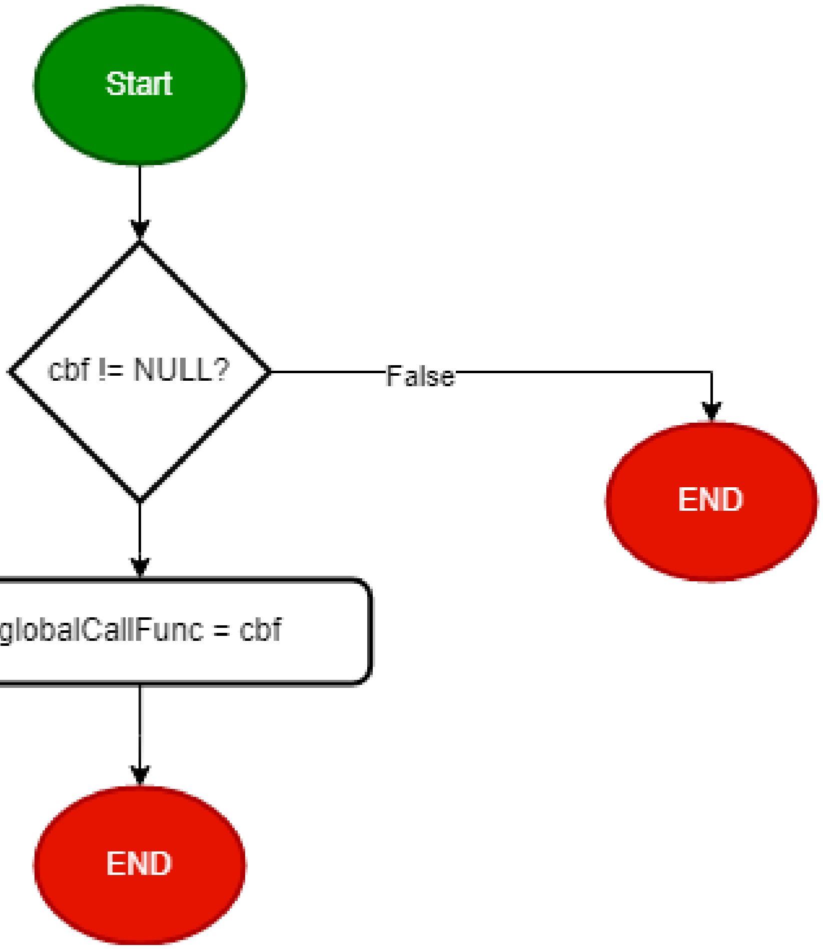
enu_timer2Status_t Timer2_enuFastPWMInit(enu_pwmMode_t copy_enPWMMode)


Start
copy_enPWMMode < TIMER2_PWM_INVALID?
False
END
local_u8Temp = TCCR2_REG
local_u8Temp &= PWM_MODE_CLR_MSK
local_u8Temp |= (copy_enPWMMode << TIMER2_PWM_MODE_OFFSET)
TCCR2_REG = local_u8Temp
END

enu_timerStatus_t vidPWM2_Generate (Uint8_t Copy_u8DutyCycle)



void vidTimer2_setcbf_OVF (cbf_t cbf)


Start
cbf != NULL?
False
END
globalCallFunc = cbf
END