Air Conditioning Project

Team Members:

Alaa Ibrahim
Bassel Yasser
Sharpel Malek
Sherif Khadr

2023

Contents

INTRODUCTION

High Level Design

01) Layered Architecture

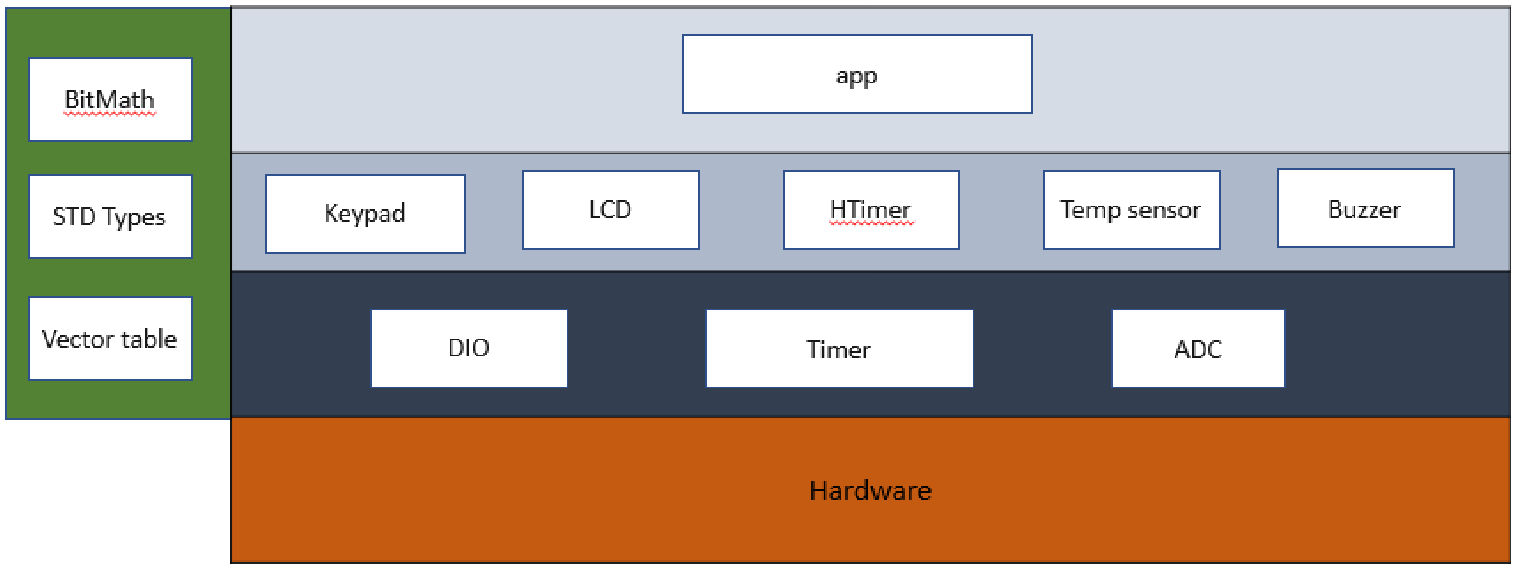


Figure 1: Project Layered Architecture

02) Modules Description

MCAL Layer:

- **DIO:** For controlling GPIO pins
- **Timer:** Provides an interface with timer 0 low-level capabilities.
- **ADC:** Provides interface to control and read from ADC peripheral.

HAL Layer:

- **Keypad:** Deal with a set of buttons arranged in a block. The 3 x 3 matrix keypad usually is used as input in a project
- **LCD:** Use for display data
- **Temp Sensor:** Provides functions to get readings from temperature sensor.
- **Buzzer:** Simple module to control a buzzer.
- **HTimer:** Provides high-level functions using the lower level timer 0 module capabilities.

Service Layer:

- **STD_Types:** Contains all the standard types used by all the layers.
- **BIT_Math:** Provides bit-wise operations.
- **Vect_table:** Contains all interrupt vectors and provides macros for dealing with general interrupt.

Application Layer:

Contains the main logic of the project.

03)   Drivers' Documentation

MCAL Layer

- **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)
```

- **Timer 0**

```
/**
 * \brief Initialize the timer with given mode
 * \param u8_a_Mode
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_voidInit(en_TIMMode_t u8_a_Mode);

/**
 * \brief Start the timer clock after prescaling it with given value
 * \param u8_a_prescaler
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_Start(en_TIM_CLK_SELECT_t u8_a_prescaler);

/**
 * \brief Function to stop timer 0
 * \return void
 */
void TIM0_Stop();

/**
 * \brief Set the timer to start from a certain value
 * \param u8_a_FlagValue The value to start the timer from
 * \return void
 */
void TIM0_SetValue(Uchar8_t u8_a_startValue);

/**
 * \brief Function to get the value of the overflow flag of timer 0
 * \param u8_a_FlagValue reference to a variable to store flag value *
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_GetOVF(Uchar8_t* u8_a_FlagValue);

/**
 * \brief Function to clear timer 0 overflow flag
 * \return void
 */
void TIM0_ClearOVF(void);

/**
 * \brief Function to get the timer state (running/stopped)
 * \param u8_a_State reference to a variable to store timer state
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_GetState(en_TIMState_t* u8_a_State);

/**
 * \brief Function to set a function to call when the timer0
 *       Overflow Interrupt is triggered
 * \param pv_a_CallbackFn reference to the function to call
 * \return en_TIMErrorState_t
 */
```

- **ADC**

```
/* Struct Contain all adc information to config it */
typedef struct
{
    void(*interruptHandler)(void);
    EN_ADC_REFERENCE_SELECTION_BITS_t referenceSource;
    EN_ADC_ADJUST_RESULT_t resultAdjust;
    EN_ADC_PRESCALER_SELECTION_t prescalerDivision;
    EN_ADC_EVENT_TRIGGER_SOUREC_t triggerSource;
}ST_ADC_CFG_t;
```

```
/**
 * \brief  : This Function Use To Init The Adc It Set Bits For Prescaler , Refrence
Source , event trigger resource and Adjust Resualt
 *
 * \param  : const ST_ADC_CFG_t *_adc
 *
 * \return : Std_ReturnType
 */
   Std_ReturnType ADC_Init(const ST_ADC_CFG_t *_adc)

/**
 * \brief  : This Function Use To Disable The ADC
 *
 * \param  : const ST_ADC_CFG_t *_adc
 *
 * \return : Std_ReturnType
 */
   Std_ReturnType ADC_Deinit(const ST_ADC_CFG_t *_adc)


/**
 * \brief  : This Function Is Used To Select ADC Channel
 *
 * \param  : const ST_ADC_CFG_t *_adc
 * \param  : EN_ADC_CHANNEL_SELECTION_t _channel
 *
 * \return : Std_ReturnType
 */
   Std_ReturnType ADC_SetChannel(const ST_ADC_CFG_t *_adc , EN_ADC_CHANNEL_SELECTION_t
   _channel)


/**
 * \brief  : This Function Use To Start Conversion
 *
 * \param  : const ST_ADC_CFG_t *_adc
 *
 * \return : Std_ReturnType
 */
   Std_ReturnType ADC_StartConversion(const ST_ADC_CFG_t *_adc)



/**
```

```
 * \brief  : This Function Use To Polling On The ADC Flag To Return The
Conversion Resualt
 *
 * \param  : const ST_ADC_CFG_t *_adc
 * \param  : Uint16_t *_ConversionResult
 *
 * \return : Std_ReturnType
 */
   Std_ReturnType ADC_GetConversionResult(const ST_ADC_CFG_t *_adc , Uint16_t
   *_ConversionResult)

/**
 * \brief : This Function Use To Make All Operation Of The Adc
 *
 * \param : const ST_ADC_CFG_t *_adc
 * \param : EN_ADC_CHANNEL_SELECTION_t _channel
 * \param : Uint16_t *_ConversionResult
 *
 * \return Std_ReturnType
 */
   Std_ReturnType ADC_Conversion(const ST_ADC_CFG_t *_adc , Uint16_t *_ConversionResult ,
   EN_ADC_CHANNEL_SELECTION_t _channel)
```

HAL Layer:

- **Keypad**

```
// Macros

#define R1     DIO_PINC_2
#define R2     DIO_PINC_3
#define R3     DIO_PINC_4
#define C1     DIO_PINC_5
#define C2     DIO_PINC_6
#define C3     DIO_PINC_7

// user defined datatypes

typedef enum EN_KEYPAD_BTNS
{
    KEY_INCREAMENT=0,
    KEY_DECREAMENT,
    KEY_SET,
    KEY_ADJUST,
    KEY_RESET,
    KEY_6,
    KEY_7,
    KEY_8,
    KEY_9,
    KEY_NOTHING

}EN_KEYPAD_BTNS;

// functions prototypes

/*************************************************************************************
************************
 *
 Name : KEYPAD_init()
 Description : This Function Initializes keypad pins (Rows are outputs & Columns are
inputs).
 ARGS : void
 return : void

**************************************************************************************
**********************/

void KEYPAD_init(void);


/*************************************************************************************
************************
 *
 Name : KEYPAD_GetButton
 Description : This Function loops over other three functions (Checks (R1,R2,R3)).
 ARGS : void
 return : the pressed key or Nothing pressed

**************************************************************************************
**********************/

EN_KEYPAD_BTNS KEYPAD_GetButton(void);
```

```
/*******************************************************************************************
************************
 *
 Name : KEYPAD_checkR1 ,  KEYPAD_checkR2, KEYPAD_checkR3
 Description : functions are checking the entire row if it pressed or not.
 ARGS : void
 return : the pressed key or Nothing pressed

*******************************************************************************************
**********************/

EN_KEYPAD_BTNS KEYPAD_checkR1(void);
EN_KEYPAD_BTNS KEYPAD_checkR2(void);
EN_KEYPAD_BTNS KEYPAD_checkR3(void);
```

- **HTimer:**

```
/**
 * \brief Generate Synchronous delay (busy waiting)*
 * \param Copy_delayTime Desired delay
 * \param Copy_timeUnit Time units (Seconds, mSeconds, uSeconds)
 *
 * \return en_HTIMErrorState_t
 */
en_HTIMErrorState_t TIM0_SyncDelay(Uint32_t u32_a_delay, en_timeUnits_t
u8_a_timeUnit);


/**
 * \brief Generates delay asynchronously
* \param u32_a_delay desired delay
 * \param u8_a_timeUnit delay time units
 * \param Copy_pvCallbackFn function to call when delay is complete
 *
 * \return en_TIMErrorState_t
 */
en_HTIMErrorState_t TIM0_AsyncDelay(Uint32_t u32_a_delay, en_timeUnits_t
u8_a_timeUnit, void (*Copy_pvCallbackFn)(void));

/**
 * \brief Function to end a delay asynchronously
 * To Stop Async Delay: No Restrictions
 * To Stop Sync Delay: should only be called in an ISR/Callback function
 *
 * \return void
 */
void TIM0_AsyncEndDelay();
```

- **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_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)
```

- **Buzzer**

```
/**
 * \brief Initialize buzzer pin as output
 * \param pst_a_buzzer reference to buzzer
 * \return void
 */
void BUZ_Init(st_Buzzer_t* pst_a_buzzer);

/**
 * \brief Turn the buzzer on/off
 * \param pst_a_buzzer reference to buzzer
 * \param u16_a_state BUZ_ON (or) BUZ_OFF
 * \return en_BuzzerErrorState_t
 */
en_BuzzerErrorState_t BUZ_SetState(st_Buzzer_t* pst_a_buzzer, en_BuzzerState_t
en_a_state);
```

- **Temperature Sensor**

```
/**
 * \brief Function to initialize the sensor port/pin
 * \param pst_a_sensor reference to sensor info
 * \return void
 */
void TSENSOR_Init(st_TempSensor_t* pst_a_sensor);

/**
 * \brief Function to get the current sensor reading
 * \param pst_a_Sensor reference to sensor info
 * \param f32_a_Value reference to variable to store Analog value
 *
 * \return en_SensorError_t
 */
en_SensorError_t TSENSOR_ReadValue(st_TempSensor_t *pst_a_Sensor, float32_t
*f32_a_Value);
```

Application Layer:

```c
/**
 * \brief Initialize all modules and execute welcome routine
 *
 * \param
 *
 * \return void
 */
void APP_Init(void);


/**
 * \brief Application main logic
 *
 * \param
 *
 * \return void
 */
void APP_Start(void);

/**
 * \brief Initialize temperature adjustment process
 *
 * \param
 *
 * \return void
 */
static void APP_adjustInit(void);


/**
 * \brief timeout callback function
 *
 * \return void
 */
void timeout(void);
```

Low Level Design

MCAL Layer:

- **DIO**

```
Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)
```

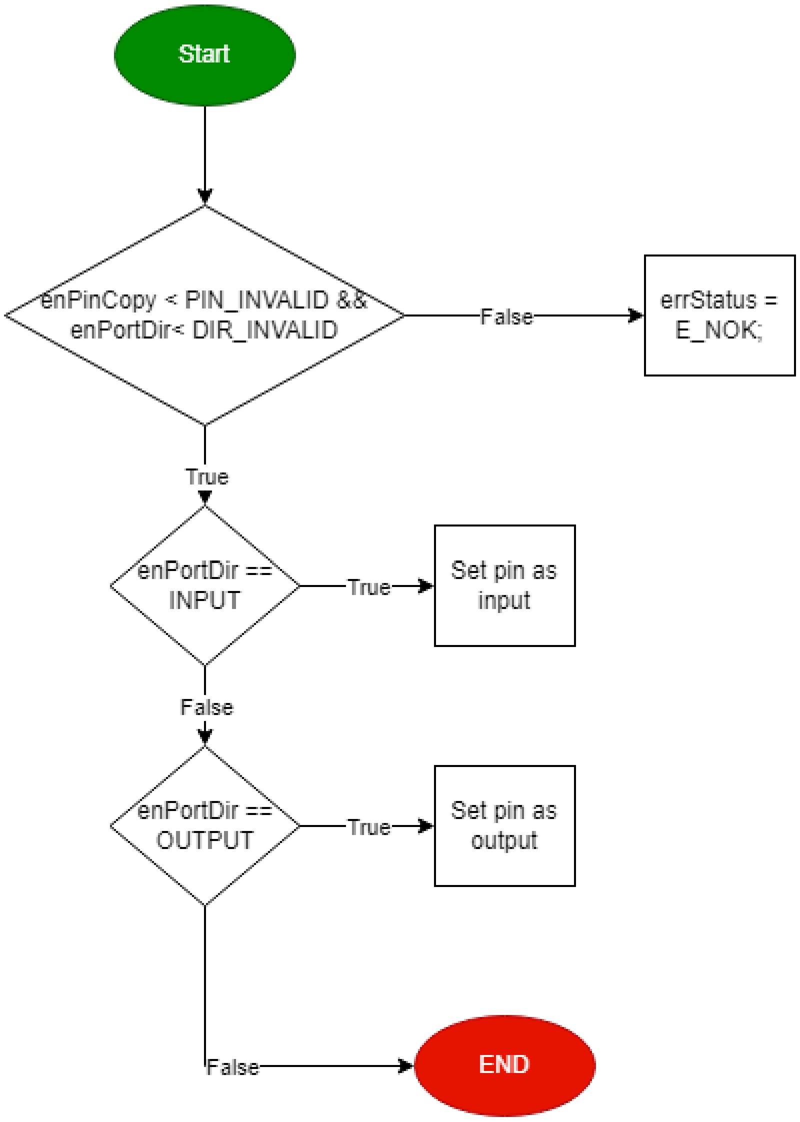


Figure 2 DIO_s8SETPinDir Flow Chart

```
Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_val enPortVal)
```

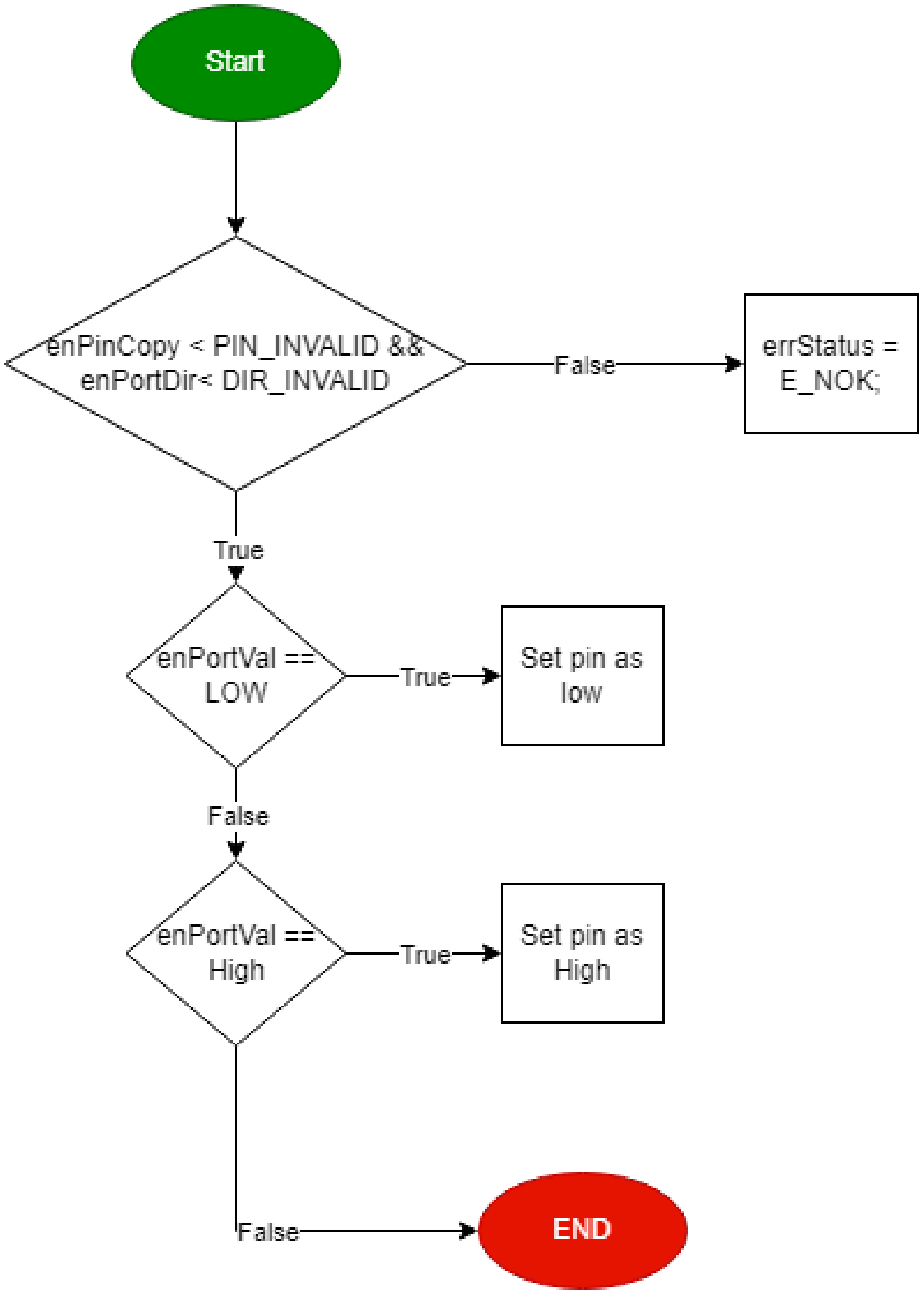


Figure 3 DIO_s8SETPinVal Flow chart

```
Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)
```

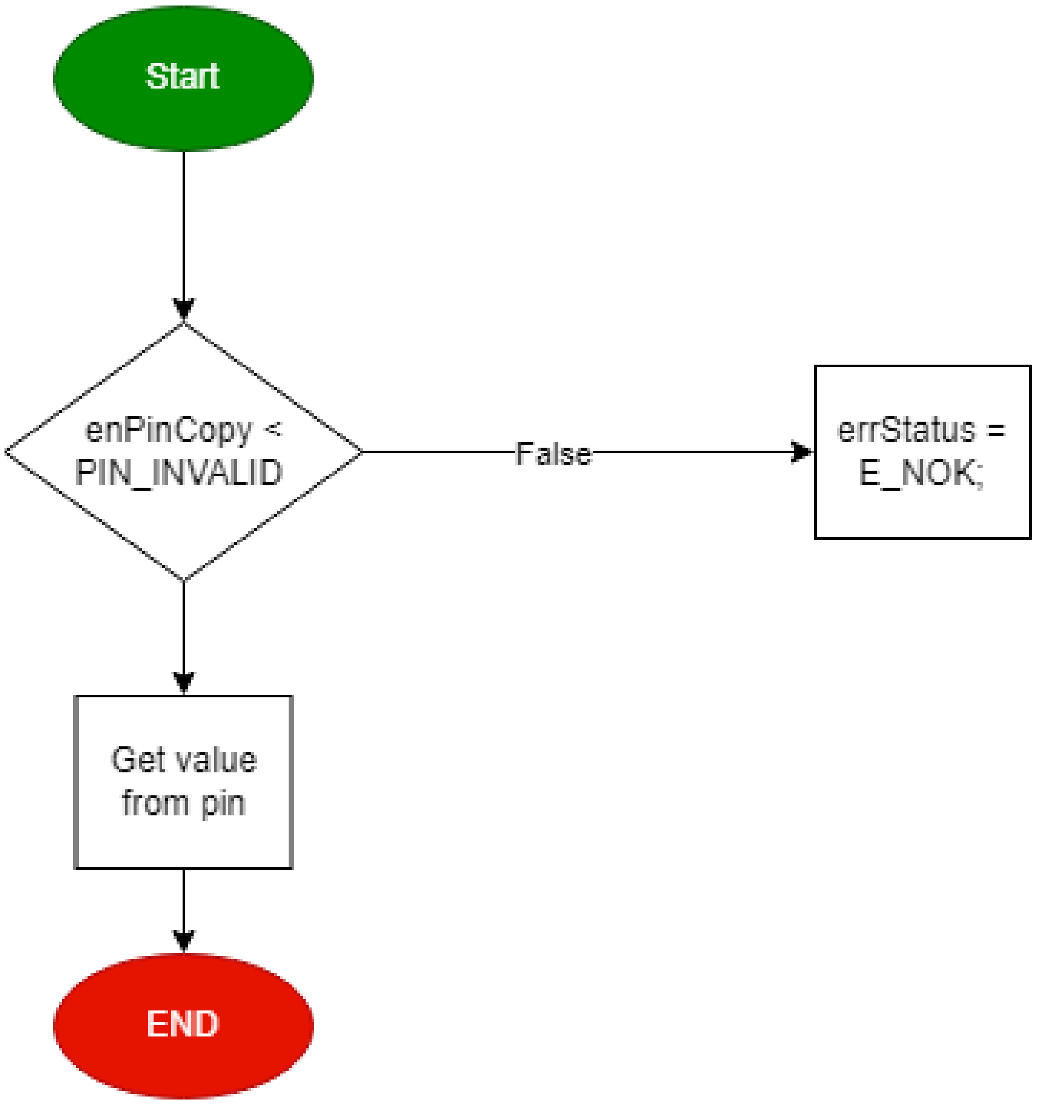


Figure 4 DIO_s8GETPinVal Flow Chart

- **Timer:**

TIM0_Init

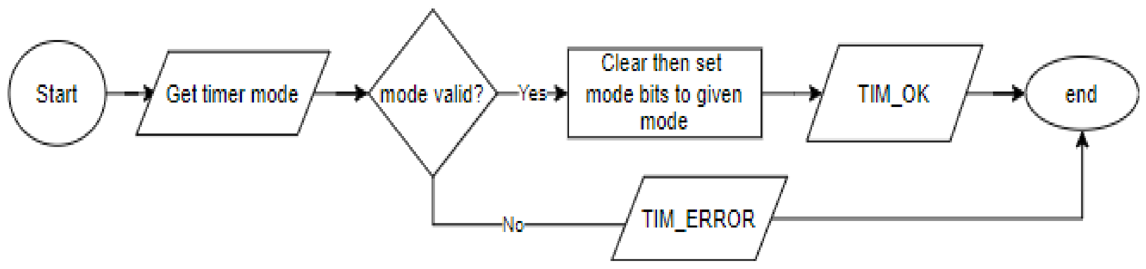


Figure 5 TIM0_Init Flow Chart

TIM0_Start

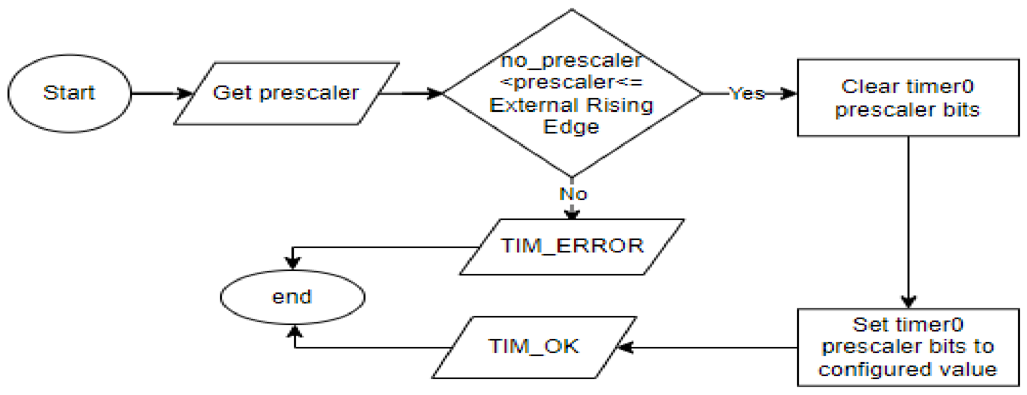

Start
Get prescaler
no_prescaler <prescaler<= External Rising Edge
Yes
Clear timer0 prescaler bits
No
TIM_ERROR
end
TIM_OK
Set timer0 prescaler bits to configured value

Figure 6 TIM0_Start Flow Chart

TIM0_Stop

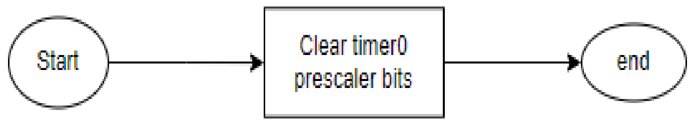
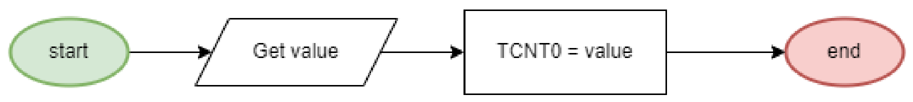
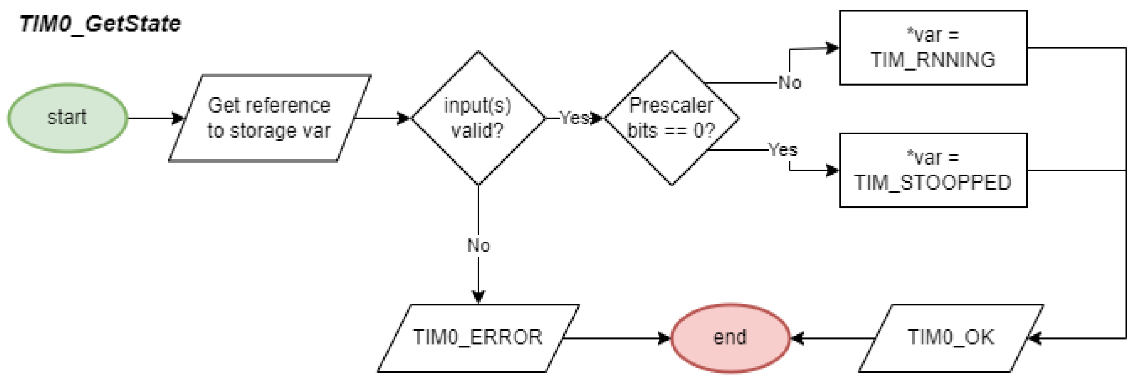


Figure 7 TIM0_Stop Flow Chart

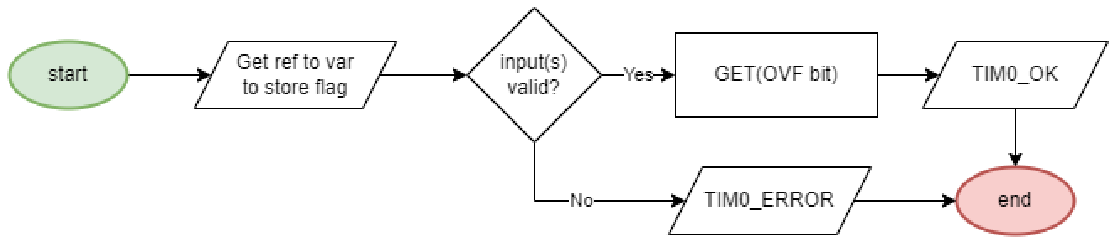
TIM0_SetValue



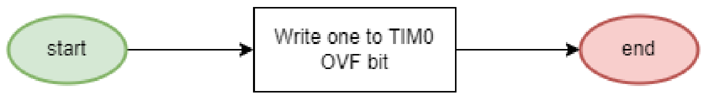
TIM0_GetState



TIM0_GetOVF



TIM0_ClearOVF



TIM0_EnableOVFInt

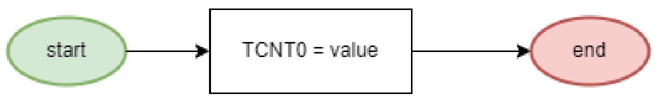


Figure 8 TIM0 remaining Flow Charts

- **ADC**

  - ADC_Init

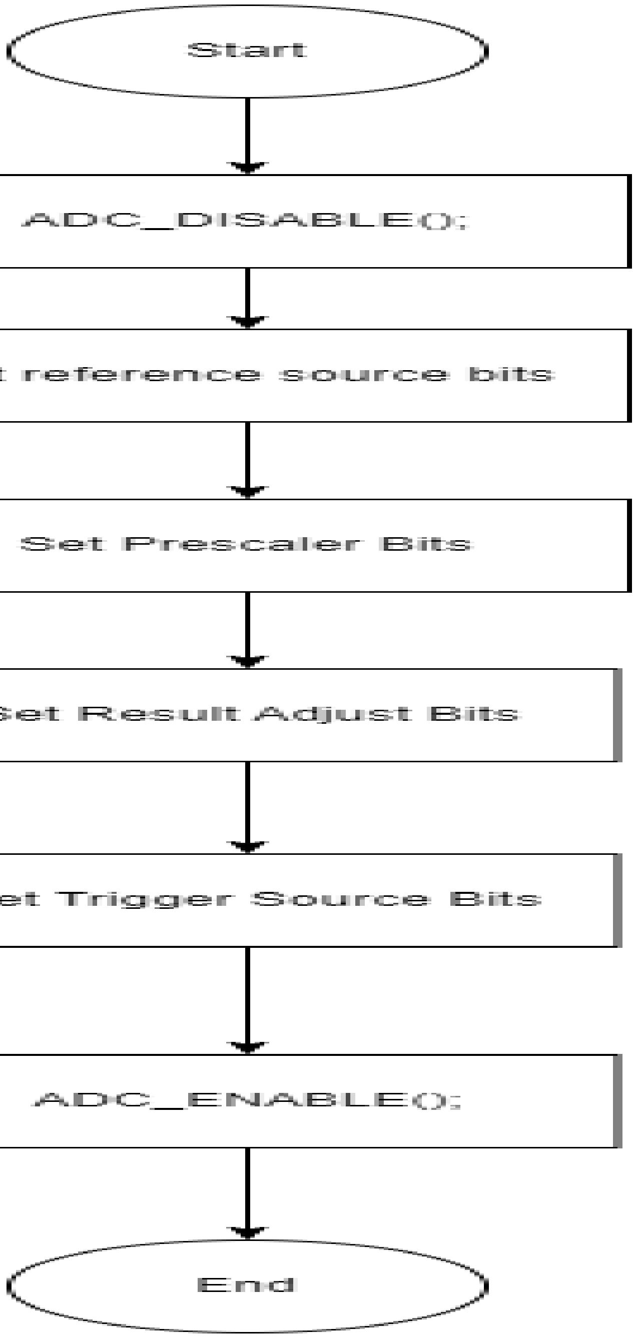
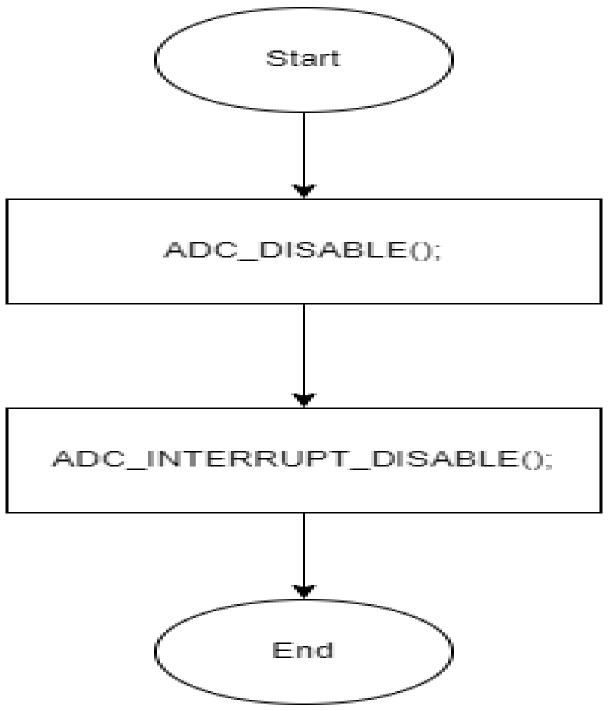


Figure 9 ADC_Init Flow Chart

ADC_Deinit



-

Figure 11 ADC_Deinit Flow Chart

ADC_SetChannel

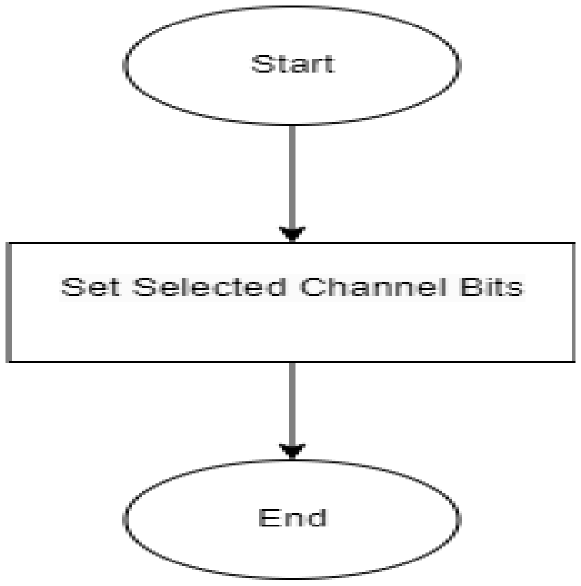


Figure 10 ADC_SetChannel Flow Chart

- **ADC_StartConversion**

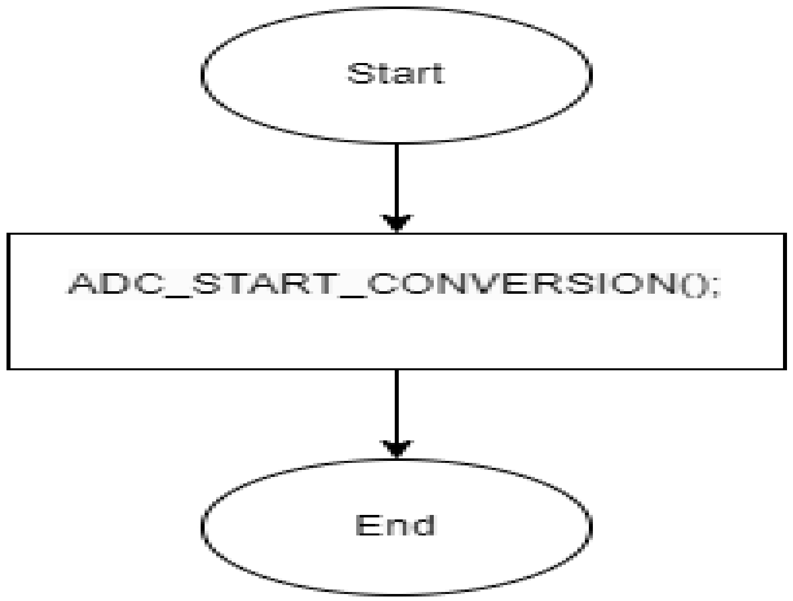


Figure 12 ADC_StartConversion Flow Chart

- **ADC_GetConversionResult**

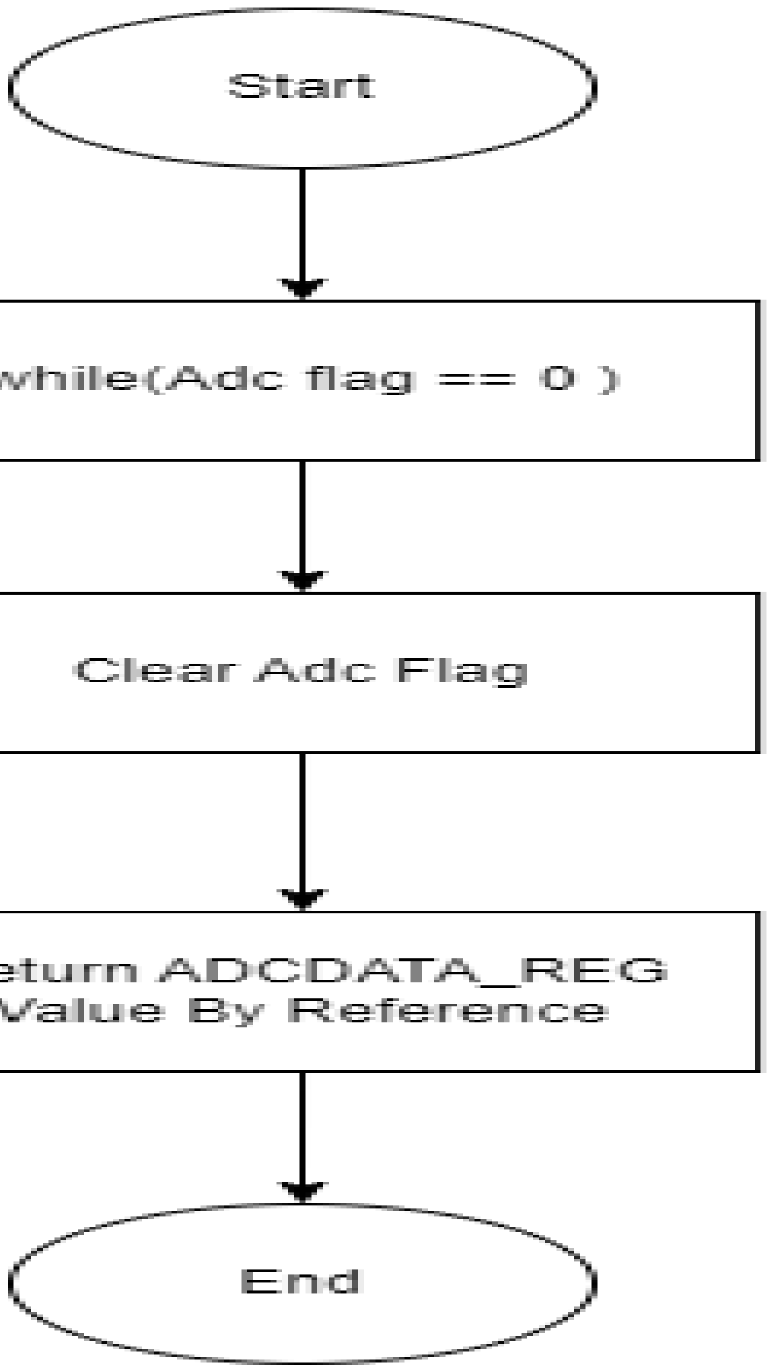


Figure 13 ADC_GetConversionResult Flow Chart

- **ADC_Conversion**

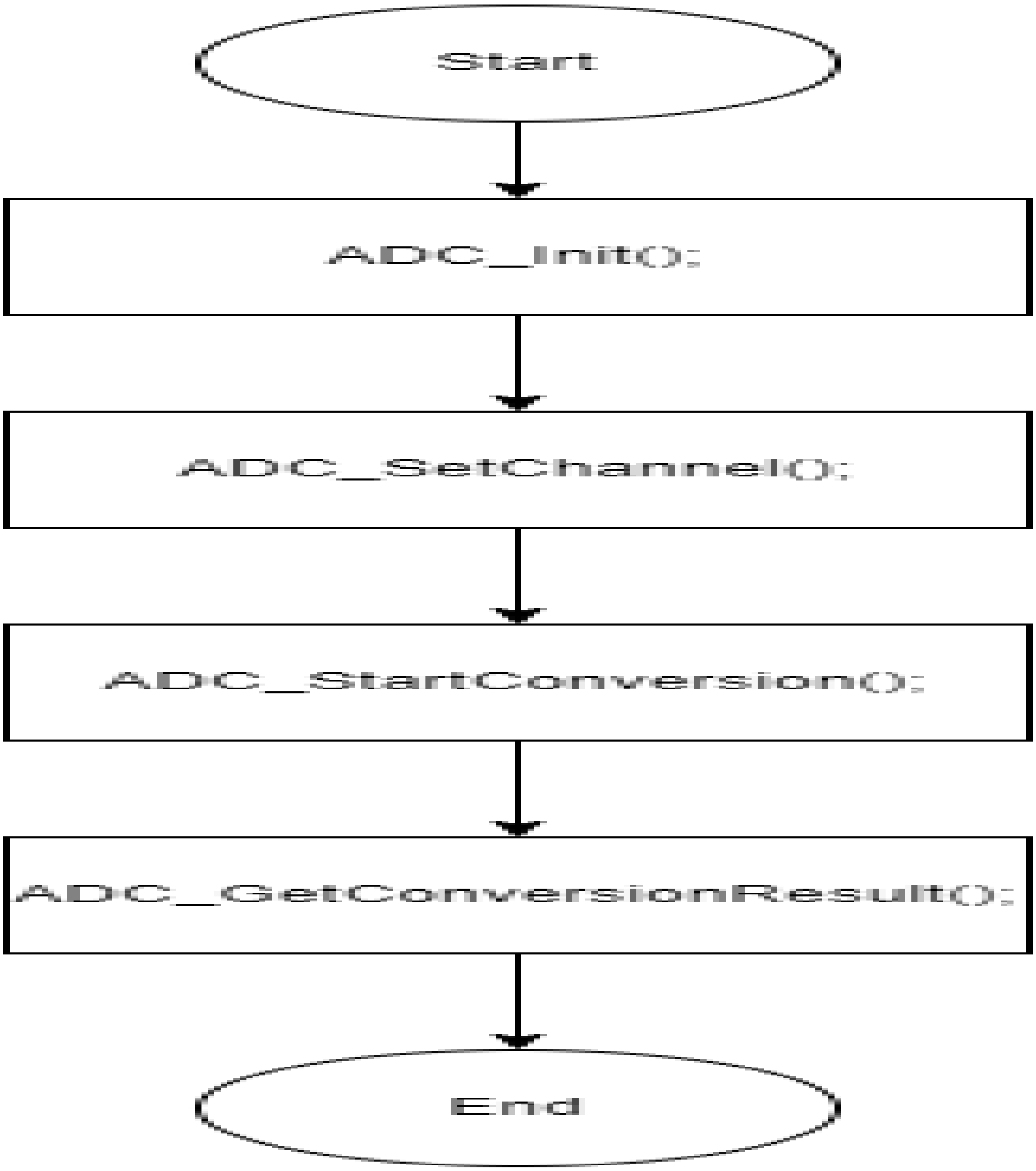
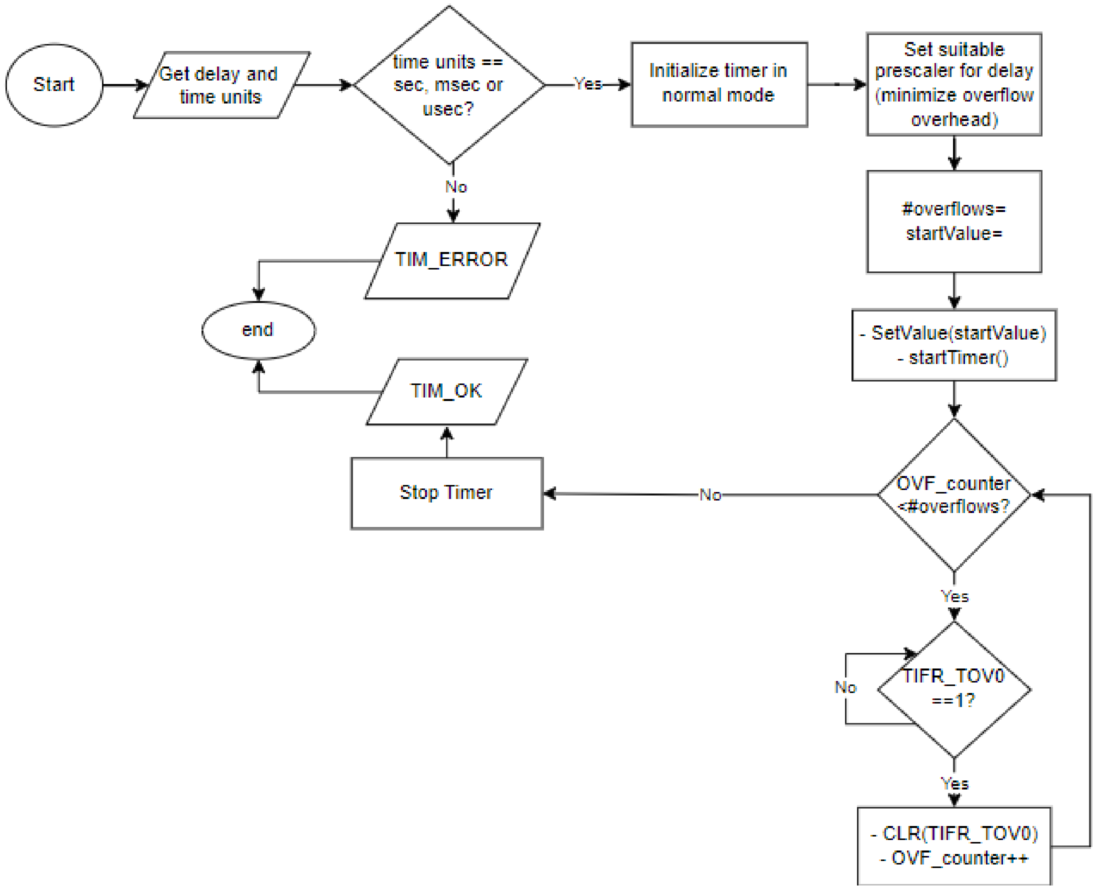


Figure 14 ADC_Conversion Flow Chart

HAL Layer

- **HTimer0**

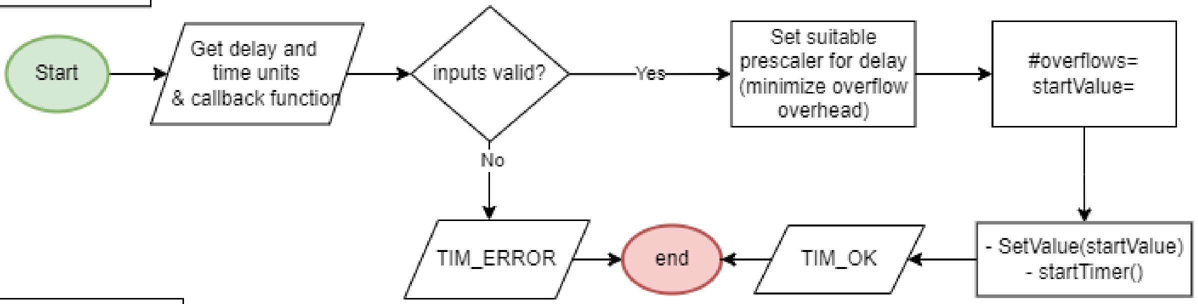
HTIM0_SyncDelay



Start
Get delay and time units
time units == sec, msec or usec?
Yes
No
Initialize timer in normal mode
Set suitable prescaler for delay (minimize overflow overhead)
#overflows= startValue=
- SetValue(startValue) - startTimer()
TIM_ERROR
end
TIM_OK
Stop Timer
No
OVF_counter <#overflows?
Yes
TIFR_TOV0 ==1?
No
Yes
- CLR(TIFR_TOV0) - OVF_counter++

Figure 15 HTIM0_SyncDelay Flow Chart

HTIM0_AsyncDelay



Start
Get delay and time units & callback function
inputs valid?
Yes
No
Set suitable prescaler for delay (minimize overflow overhead)
#overflows= startValue=
- SetValue(startValue) - startTimer()
TIM_OK
end
TIM_ERROR

HTIM0_AsyncEndDelay

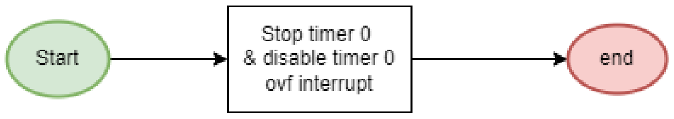


Figure 16 HTIM0_AsyncDelay and EndDelay

- **LCD**

```
void HLCD_vidInit(void)
```

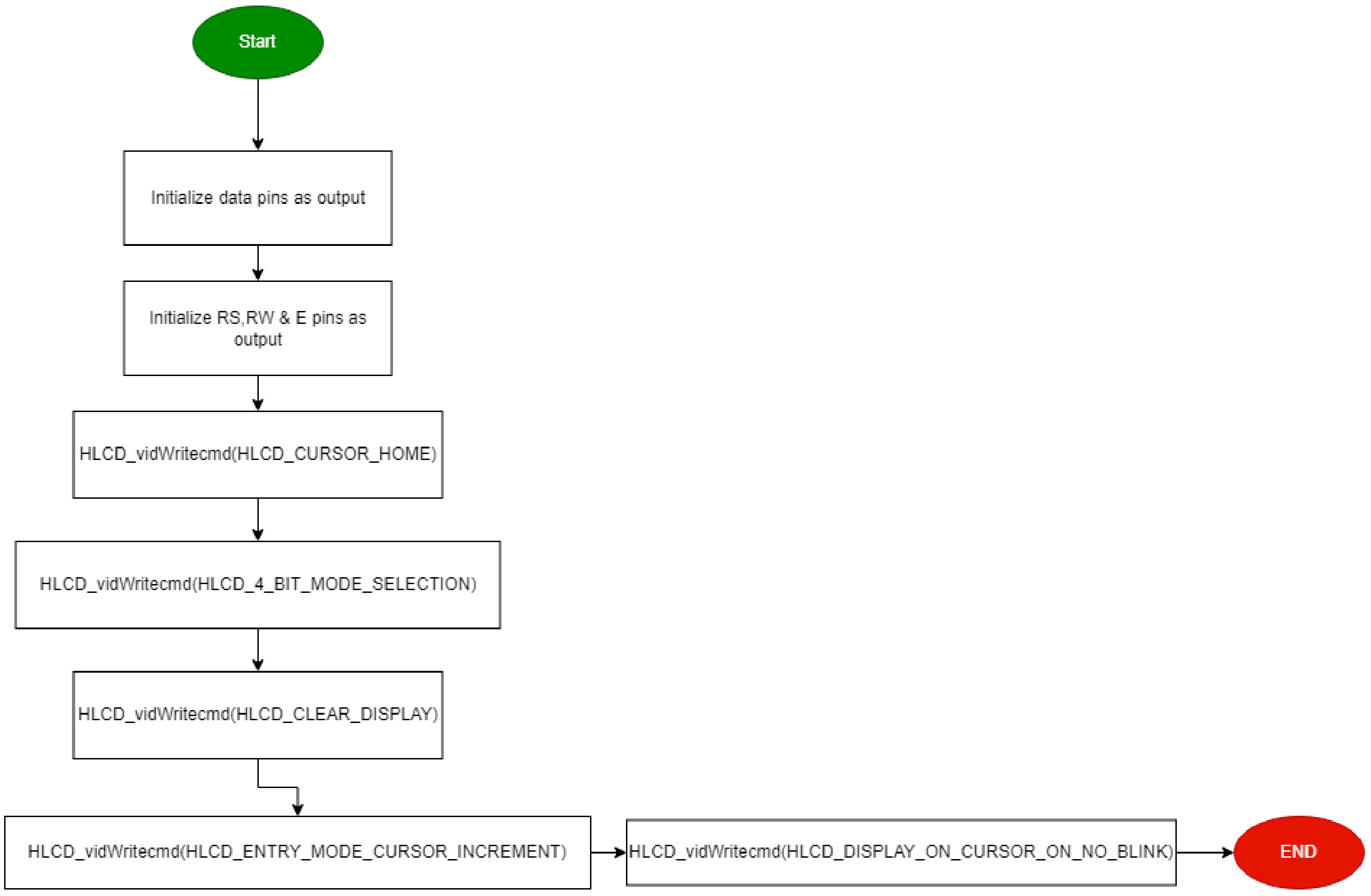

Start
Initialize data pins as output
Initialize RS,RW & E pins as output
HLCD_vidWritecmd(HLCD_CURSOR_HOME)
HLCD_vidWritecmd(HLCD_4_BIT_MODE_SELECTION)
HLCD_vidWritecmd(HLCD_CLEAR_DISPLAY)
HLCD_vidWritecmd(HLCD_ENTRY_MODE_CURSOR_INCREMENT)
HLCD_vidWritecmd(HLCD_DISPLAY_ON_CURSOR_ON_NO_BLINK)
END

Figure 17 HLCD_vidInit Flow Chart

```
void HLCD_vidWritecmd(Uint8_t u8commandCopy)
```

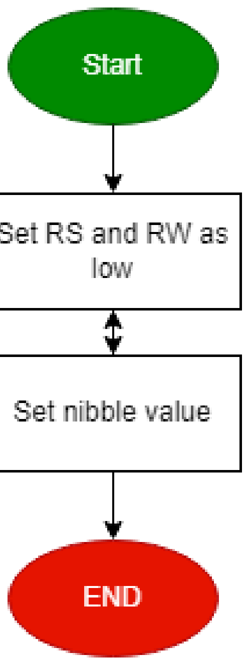


Figure 18 HLCD_vidWritecmd Flow Chart

```
void HLCD_vidWriteChar(Uint8_t u8CharCopy)
```

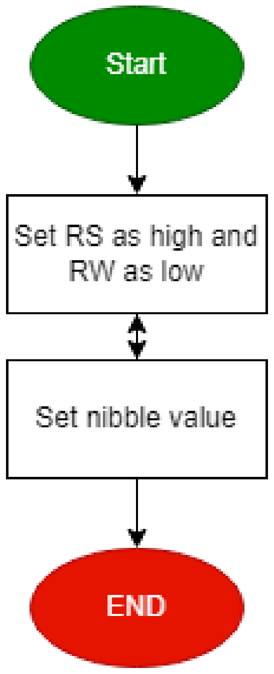


Figure 19 HLCD_vidWriteChar Flow Chart

```
void HLCD_ClrDisplay(void)
```

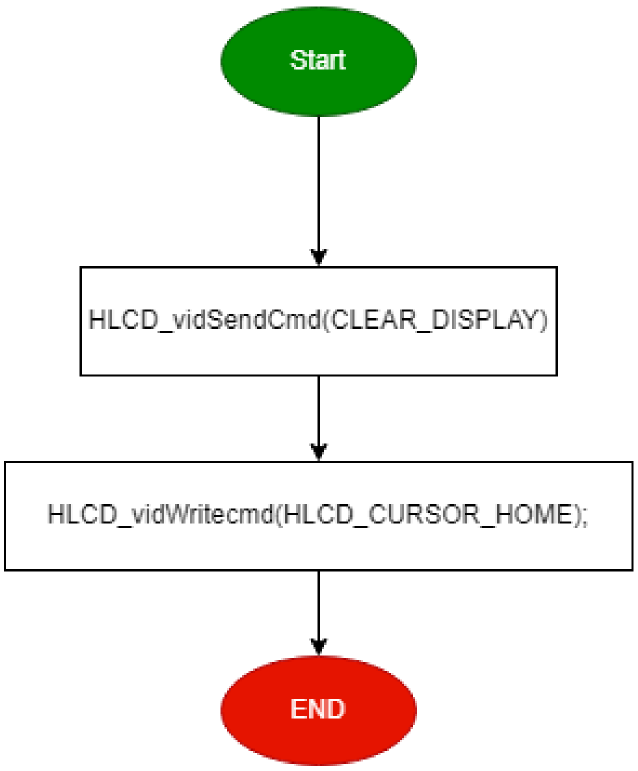

Start
HLCD_vidSendCmd(CLEAR_DISPLAY)
HLCD_vidWritecmd(HLCD_CURSOR_HOME);
END

Figure 20 HLCD_ClrDisplay Flow Chart

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

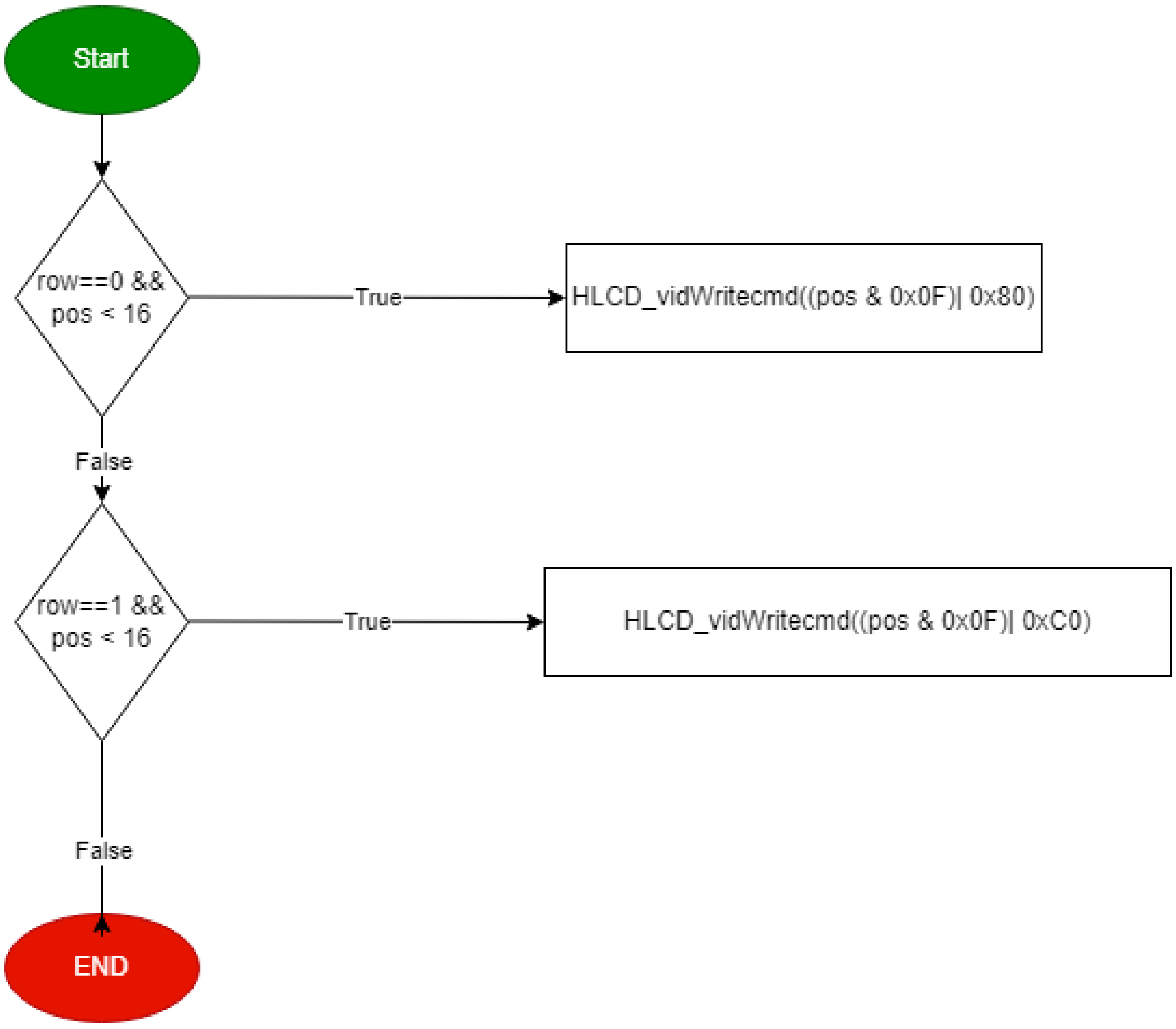

Start
row==0 && pos < 16
True
HLCD_vidWritecmd((pos & 0x0F)| 0x80)
False
row==1 && pos < 16
True
HLCD_vidWritecmd((pos & 0x0F)| 0xC0)
False
END

Figure 21 HLCD_gotoXY Flow Chart

```
void HLCD_WriteString(Uint8_t* str)
```

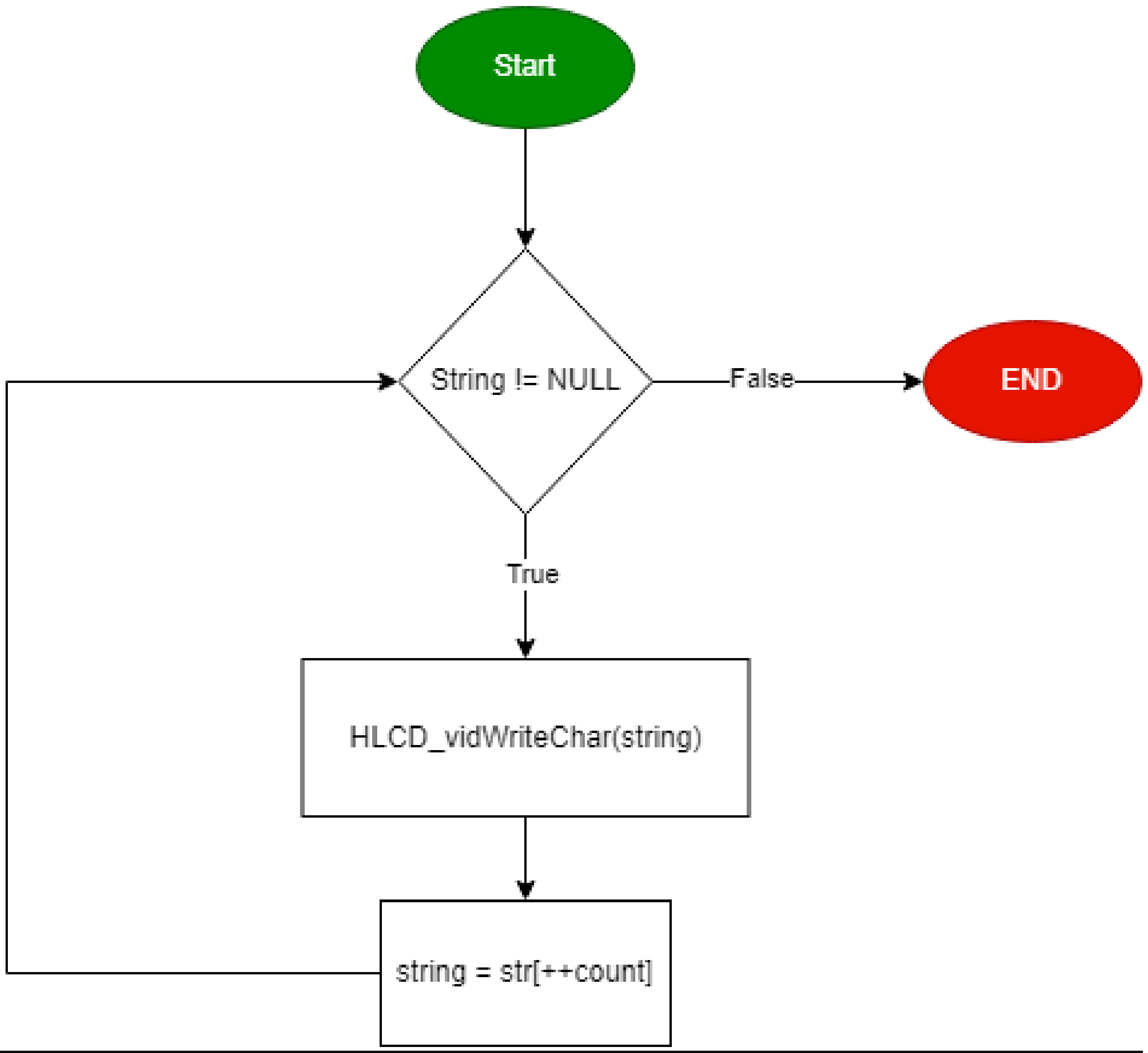


Figure 22 HLCD_WriteString Flow Chart

```
void HLCD_WriteInt(Uint32_t number)
```

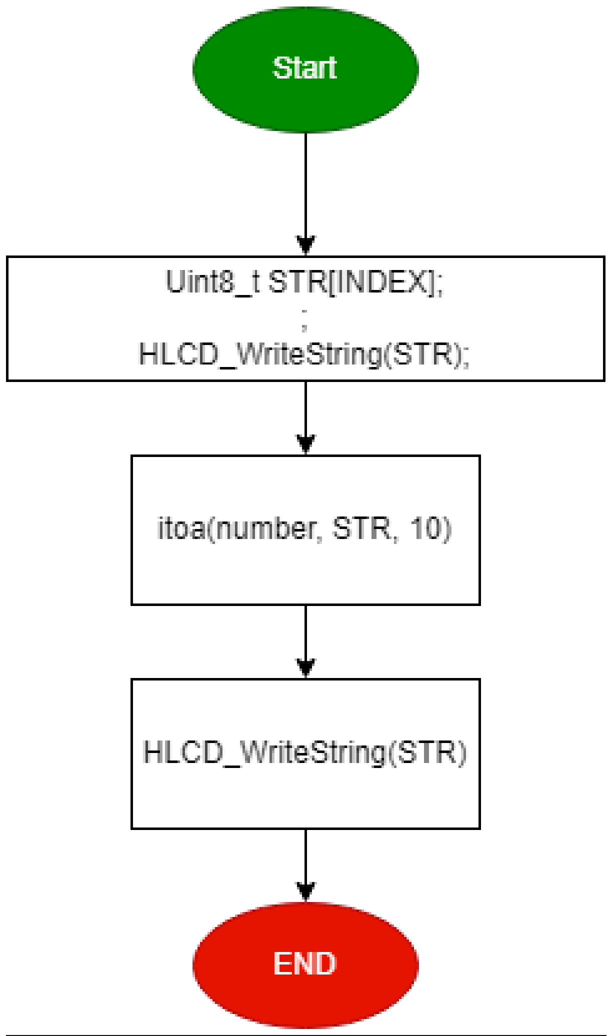


Figure 23 HLCD_WriteInt Flow Chart

```
void HLCD_vidCreatCustomChar(Uint8_t* pu8custom, Uint8_t u8Location)
```

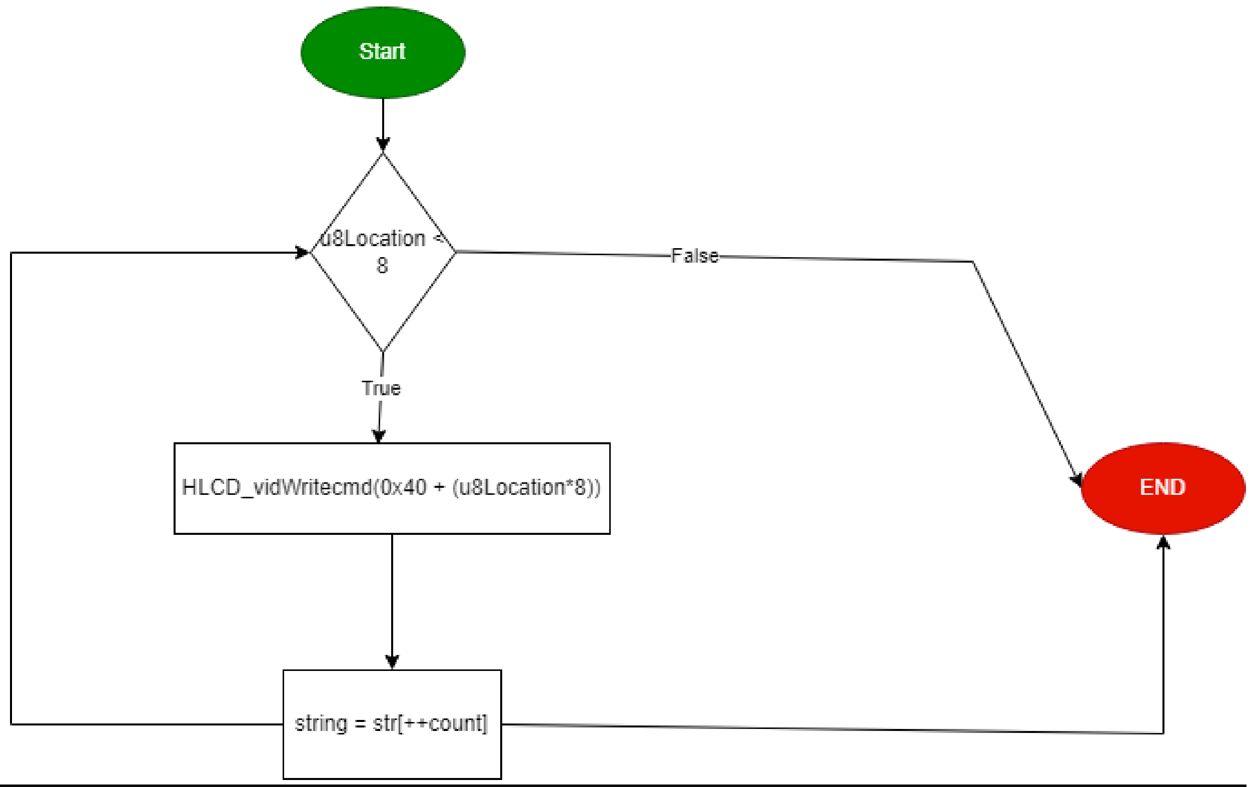


Figure 24 HLCD_vidCreatCustomChar Flow Chart

- **Keypad**

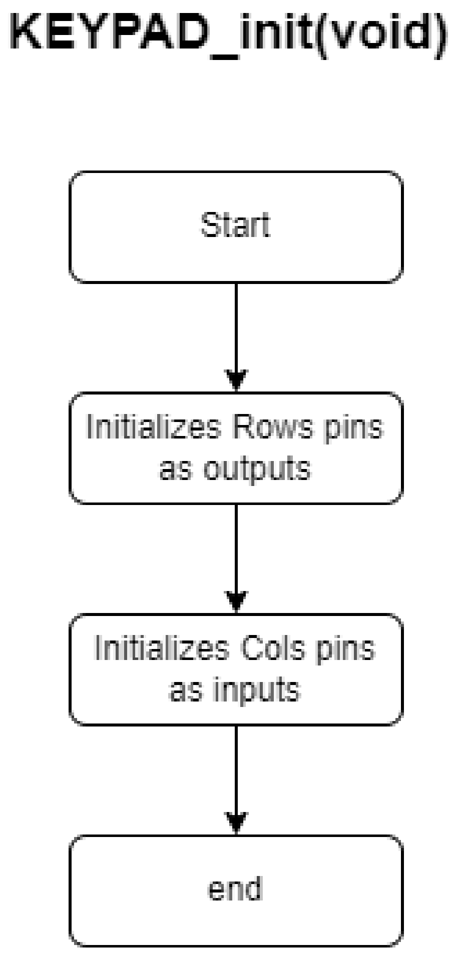

KEYPAD_init(void)
Start
Initializes Rows pins as outputs
Initializes Cols pins as inputs
end

Figure 25 KEYPAD_Init Flow Chart

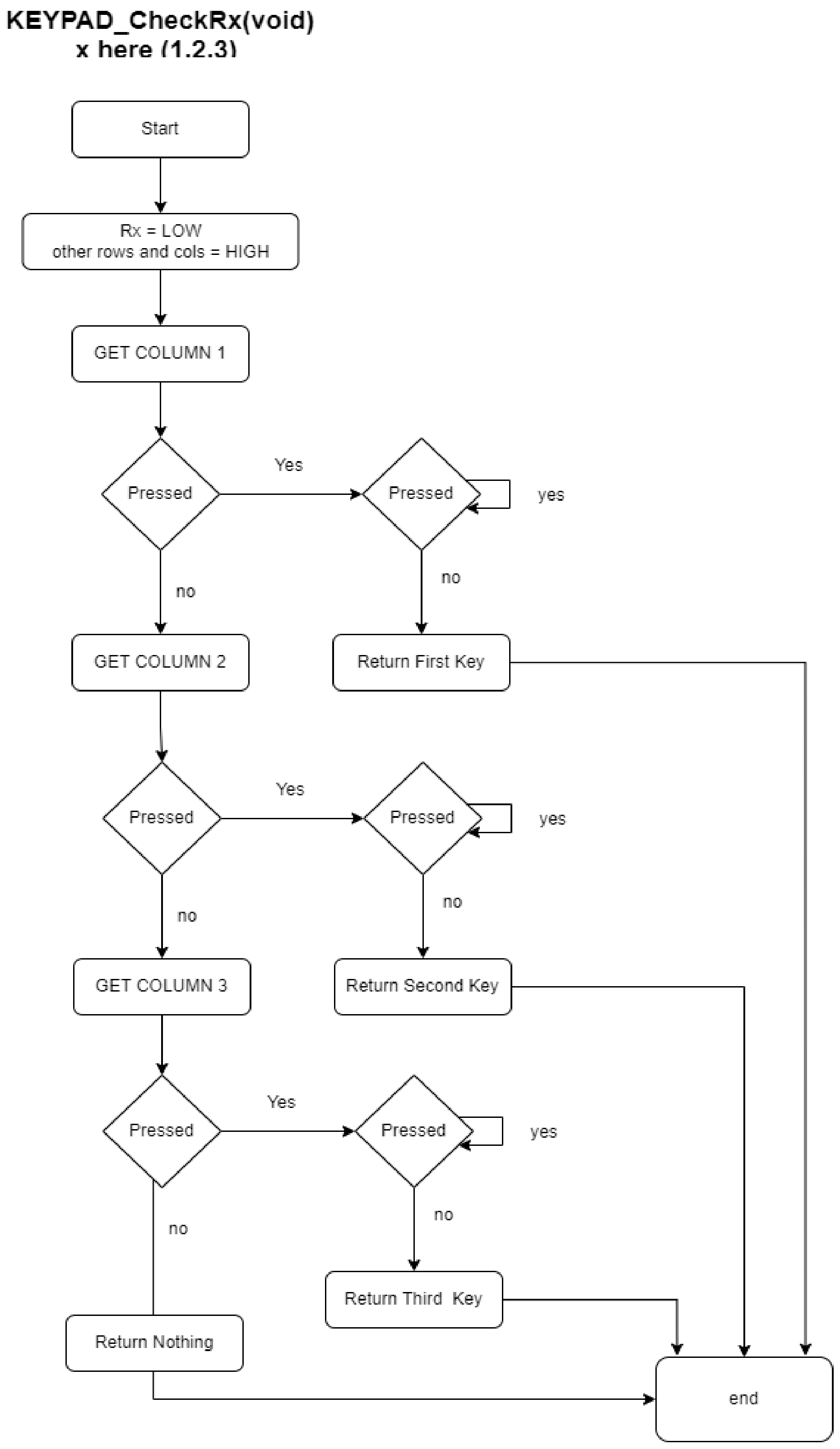


Figure 26 KEYPAD_CheckRx Flow Chart

GetButton(void)

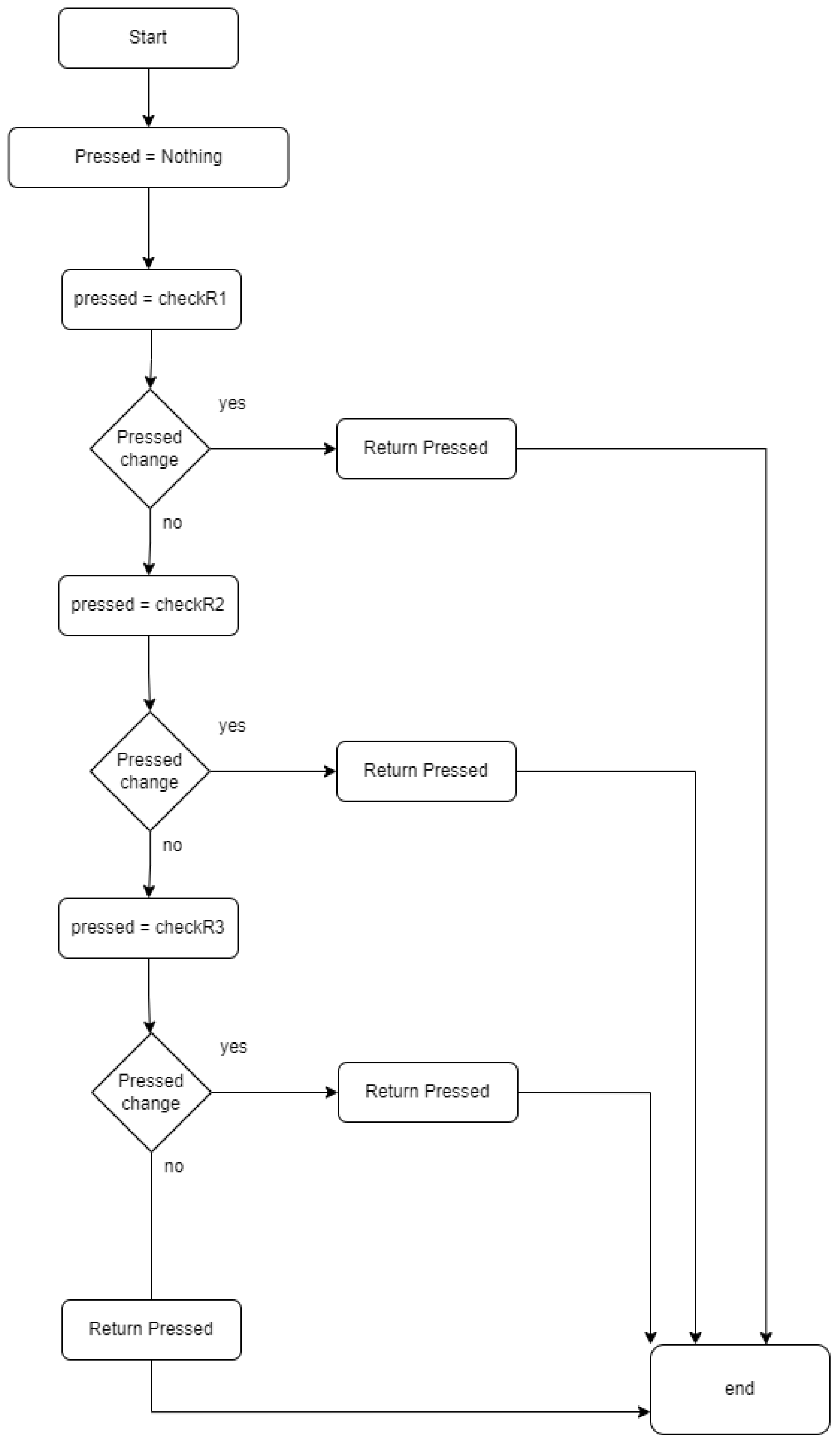
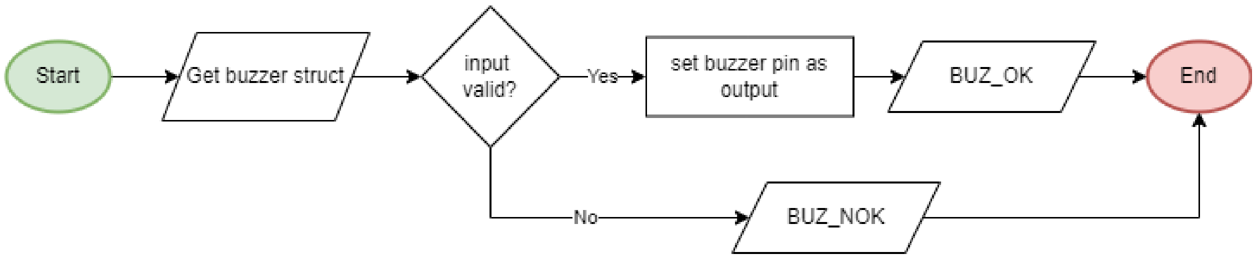


Figure 27 GetButton Flow Chart

- **Buzzer**

BUZ_Init



BUZ_SetState

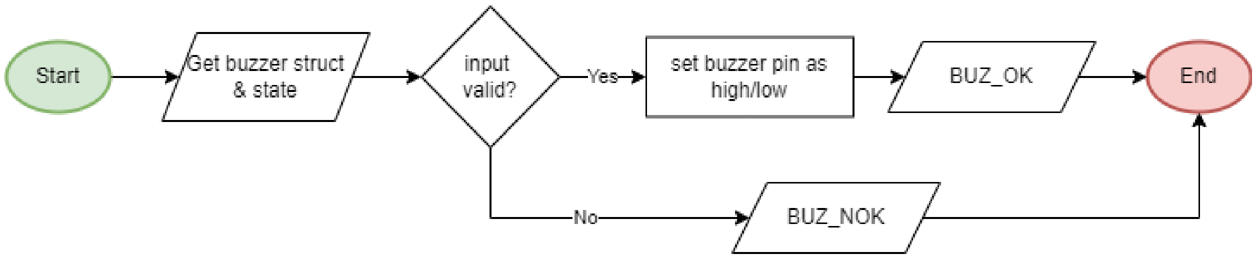
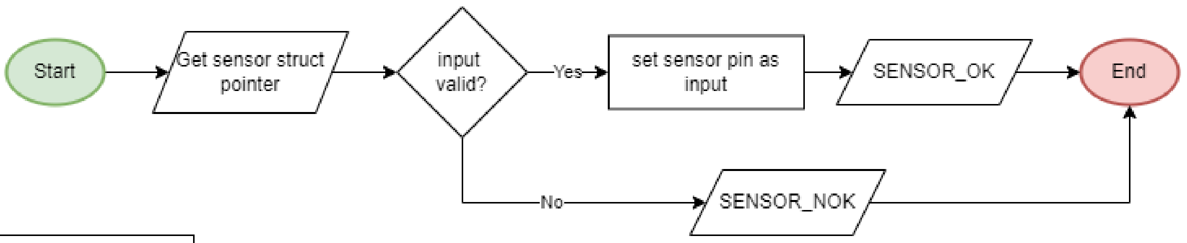


Figure 28 Buzzer Init & SetState Flow Charts

- **Temperature Sensor**

TSENSOR_Init



TSENSOR_ReadValue

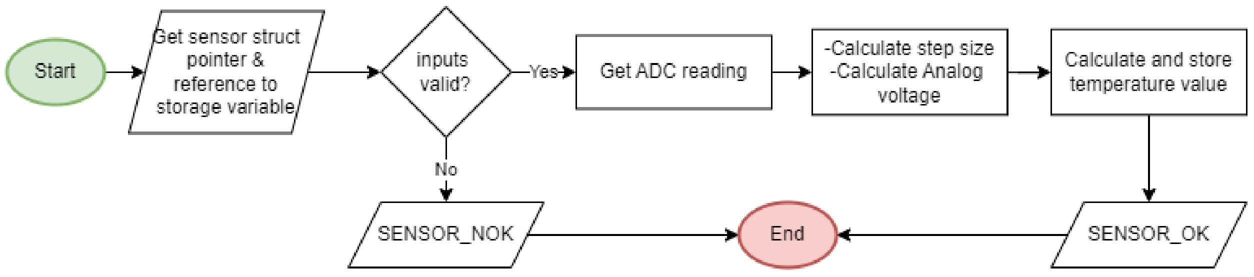

Start
Get sensor struct pointer & reference to storage variable
inputs valid?
Yes
Get ADC reading
-Calculate step size
-Calculate Analog voltage
Calculate and store temperature value
SENSOR_OK
End
No
SENSOR_NOK

Figure 29 Temp Sensor Init & ReadValue Flow Charts

Application Layer:

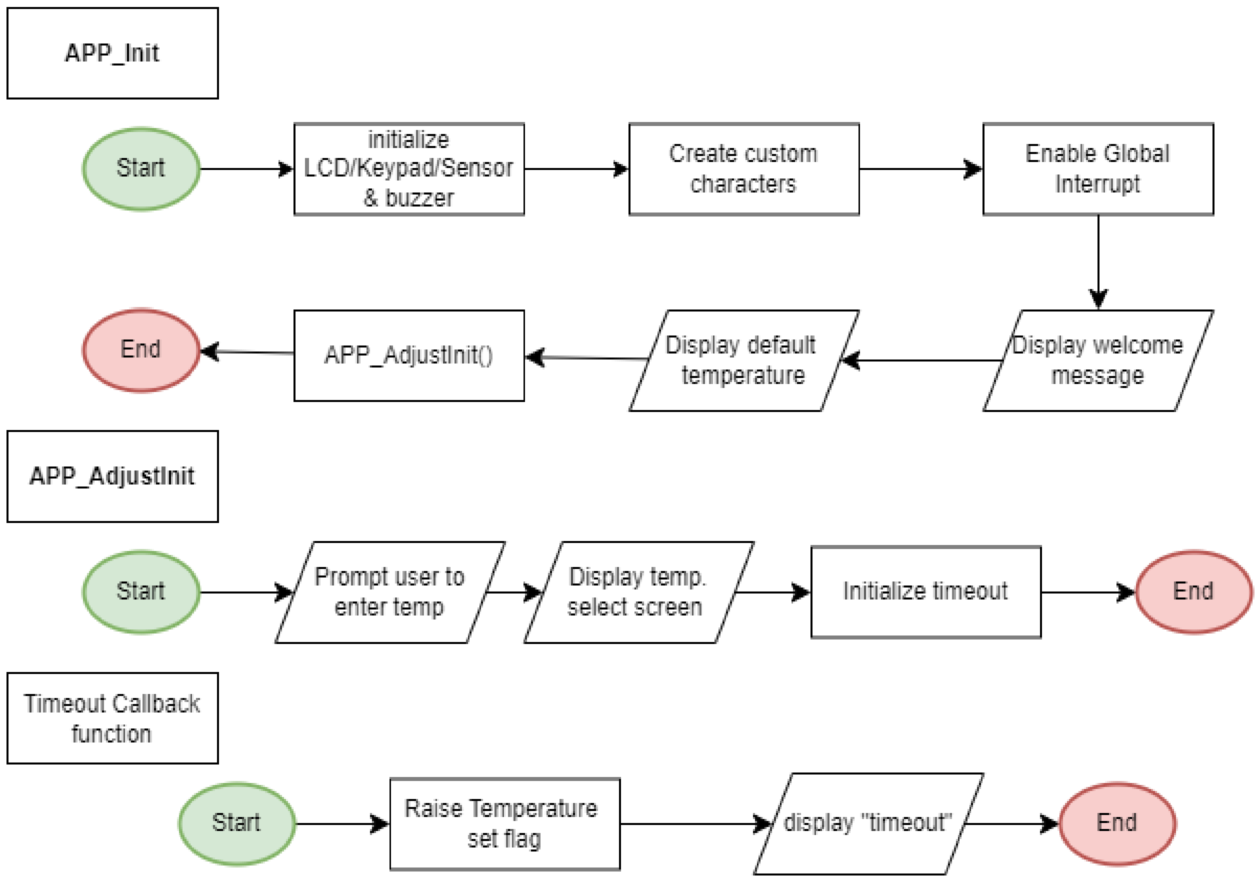

APP_Init
Start
initialize LCD/Keypad/Sensor & buzzer
Create custom characters
Enable Global Interrupt
Display welcome message
Display default temperature
APP_AdjustInit()
End
APP_AdjustInit
Start
Prompt user to enter temp
Display temp. select screen
Initialize timeout
End
Timeout Callback function
Start
Raise Temperature set flag
display "timeout"
End

Figure 30 App. APIs Flow Charts

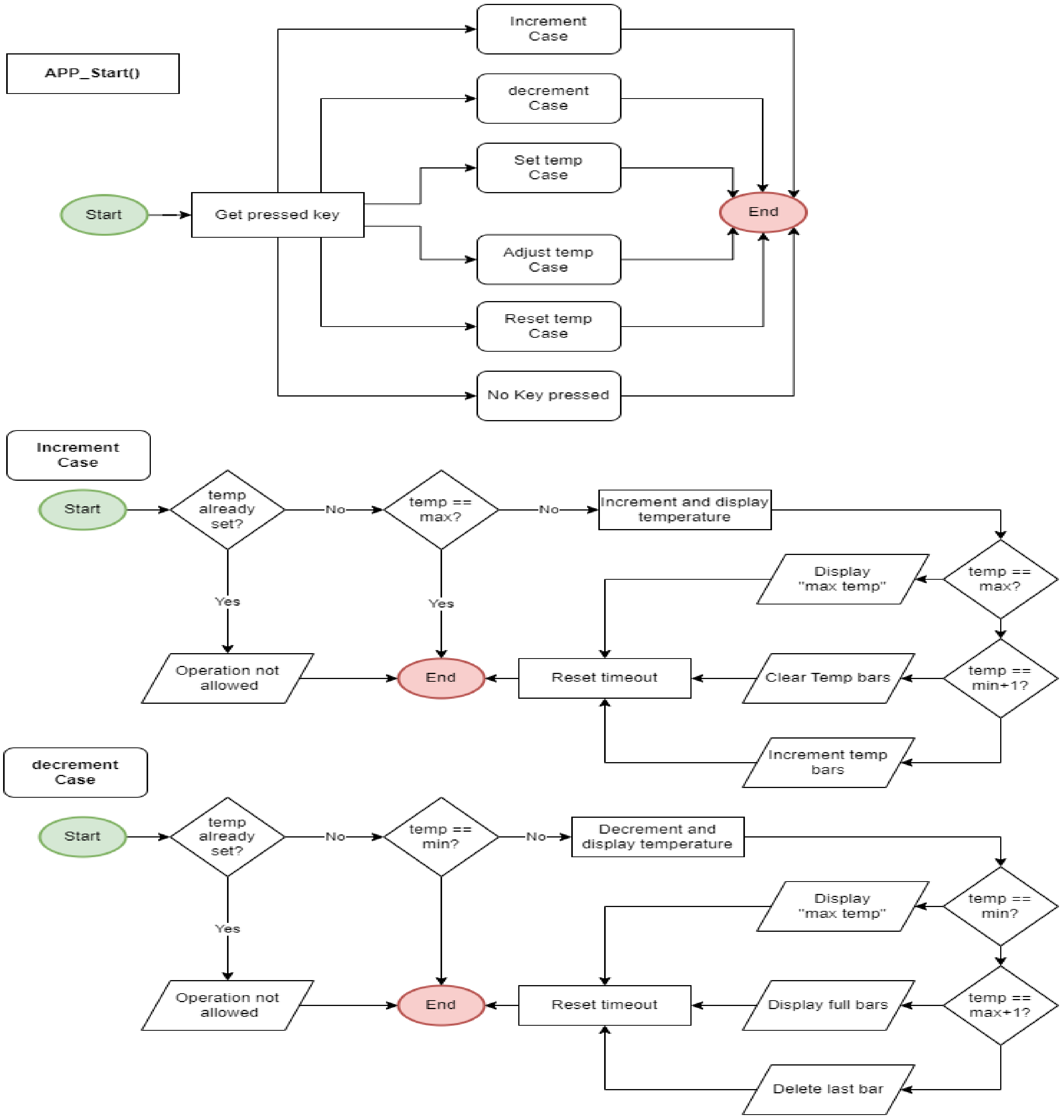


Figure 31 APP_Start & App States Flow Charts

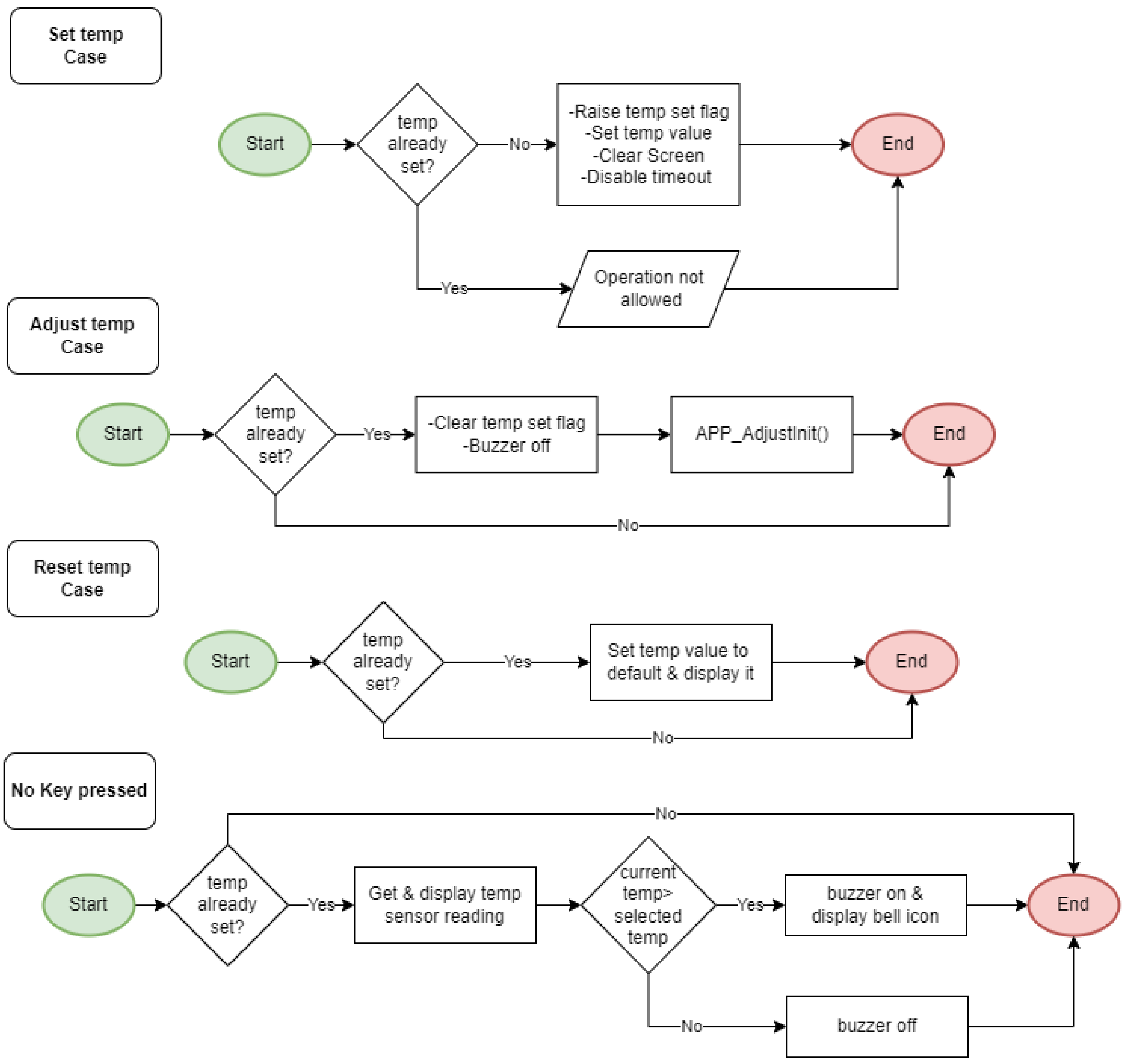


Figure 32 Cont. App States Flow Charts