RGB LED control V2

Design

Bassel Yasser Mahmoud

Mahmoud Adel

2023

Contents

Project Introduction

Read System Requirements

Hardware Requirements

1. Use the TivaC board
2. Use SW1 as an input button
3. Use the RGB LED

Software Requirements

4. The RGB LED is OFF initially
5. Pressing SW1:
    1. After the first press, the Red led is on **for 1 second only**
    2. After the second press, the Green Led is on **for 1 second only**
    3. After the third press, the Blue led is on **for 1 second only**
    4. After the fourth press, all LEDs are on **for 1 second only**
    5. After the fifth press, should disable all LEDs
    6. After the sixth press, repeat steps from 1 to 6

High Level Design

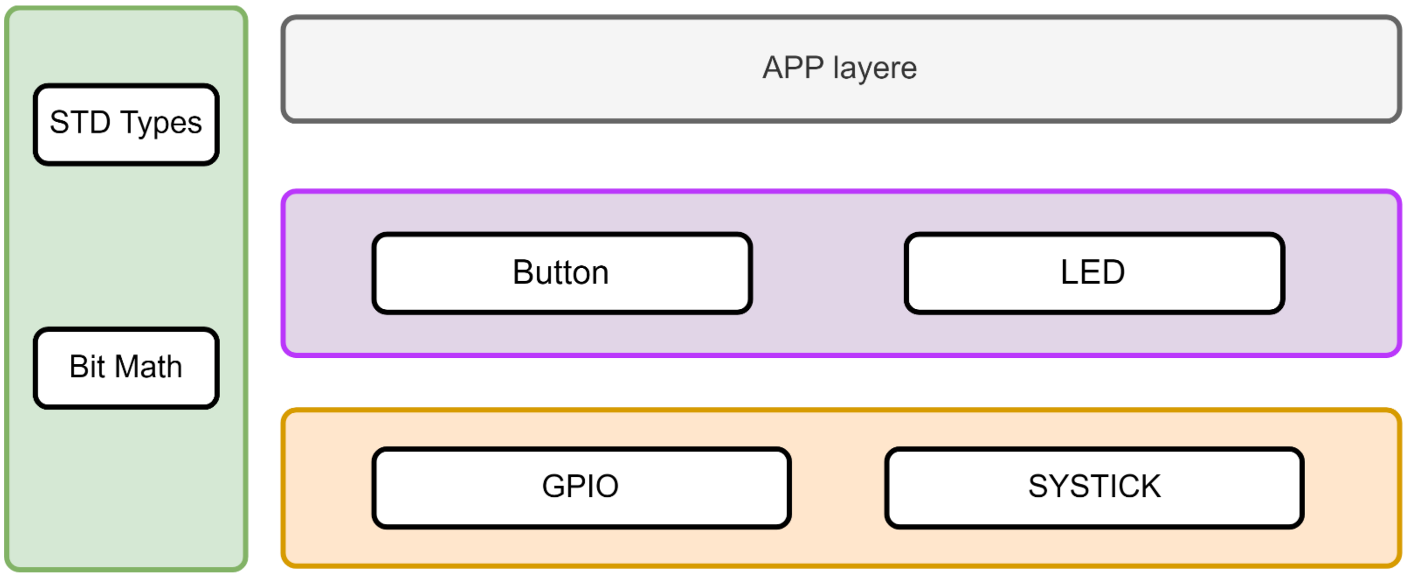
Layered Architecture

APP Layer: written in high level languages like java, C++, C# with rich GUI support. Application layer calls the middleware api in response to action by the user or an event.

HAL Layer: are a way to provide an interface between hardware and software so applications can be device independent.

MCAL Layer: is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU. Details of the MCAL software module are shown below.

Common Layer: is the layer which consists of BIT_MATH and STD types



Module Description

- **APP Layer**
  - **App:** written in high level languages like java, C++, C# with rich GUI support. Application layer calls the middleware api in response to action by the user or an event.
- **HAL Layer**
  - **button:** Initialize selected button pin as input
  - **Led:** this led module configure selected pin as output and generate volt
- **MCAL Layer**
  - **GPIO:** this module having configuration and Initialization for GPIO which communicate to hardware register directly
  - **SYSTICK:** Timer Core Peripheral from ARM design
- **COMMON Layer**
  - **std_types:** having basic standard types like (Uint32_t, Uint8_t, …..)
  - **bit_math :** Consist of bit manipulation like (SetBit, ClrBit, GetBit, ..)

Drivers' documentation

APP

APP_vidInit

Service name	APP_vidInit
Description	This Function Make Modules Initialization
Syntax	void APP_vidInit (void)
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	void
Parameters (out)	None
Return	void
Available via	app.h

APP_vidStart

Service name	APP_vidStart
Description	This Function Start the Application.
Syntax	void APP_vidStart (void)
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	void
Parameters (out)	None
Return	void
Available via	app.h

HAL

HLED module

HLed_Init

Service name	HLed_Init
Description	This Function Init LED dio pin as output
Syntax	`enu_ledError_t HLed_Init (enu_pin en_pinNum)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`en_pinNum: dio pin selection`
Parameters (out)	None
Return	`en_ledError_t`
Available via	hled.h

HLed_on

Service name	HLed_on
Description	This Function give LED pin logic 1
Syntax	`enu_ledError_t HLed_on (enu_pin en_pinx);`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`en_pinNum: dio pin selection`
Parameters (out)	None
Return	`en_ledError_t`
Available via	hled.h

HLed_off

Service name	HLed_off
Description	This Function give LED pin logic 0
Syntax	`enu_ledError_t HLed_off (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`en_pinNum: dio pin selection`
Parameters (out)	None
Return	`en_ledError_t`
Available via	hled.h

HLed_toggle

Service name	HLed_toggle
Description	This Function Change previous state of LED pin
Syntax	`enu_ledError_t HLed_toggle (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`en_pinNum: dio pin selection`
Parameters (out)	None
Return	`en_ledError_t`
Available via	hled.h

Button module

HButton_Init

Service name	HButton_Init
Description	This Function Initialize button DIO pin as input and pull up
Syntax	`enu_buttonError_t HButton_Init (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinx: DIO pin number**
Parameters (out)	None
Return	*BUTTON_OK: in case of successful operation*
	BUTTON_NOK: in case of failer operation
Available via	button.h

HButton_getPinVal

Service name	HButton_getPinVal
Description	This Function Get button state
Syntax	`enu_buttonError_t HButton_getPinVal (enu_pin en_pinx, Uint8_t* pu8_refVal)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinx:** DIO pin number
Parameters (out)	**pu8_refVal:** address of variable which button state to be stored
Return	*BUTTON_OK: in case of successful operation*
	BUTTON_NOK: in case of failer operation
Available via	button.h

MCAL

GPIO module

MGPIO_u8Init

Service name	**MGPIO_u8Init**
Description	This Function Initialize GPIO configuration
Syntax	`uint8_ MGPIO_u8Init (st_gpio_cfg_t* st_gpio_cfg)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**st_gpio_cfg:** Address of struct Instance
Parameters (out)	None
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio_Interface.h

MGPIO_u8SetPinData

Service name	**MGPIO_u8SetPinData**
Description	This Function Initialize Pin Value High or Low
Syntax	`uint8_ MGPIO_u8SetPinData (enu_pin_t Copy_enPinNum, uint8_ Copy_PinValue)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**Copy_enPinNum:** MGPIO_PINA_0 ~ MGPIO_PINF_7
	Copy_PinValue: MGPIO_PIN_LOW / MGPIO_PIN_HIGH
Parameters (out)	None
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio _Interface.h

MGPIO_u8GetPinData

Service name	**MGPIO_u8GetPinData**
Description	This Function Get value from selected pin
Syntax	`uint8_ MGPIO_u8GetPinData (enu_pin_t Copy_enPinNum, uint8_* Ref_puint8_PinVal)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**Copy_enPinNum:** MGPIO_PINA_0 ~ MGPIO_PINF_7
Parameters (out)	**Ref_puint8_PinVal:** Reference to variable where the value status store on it
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio _Interface.h

MGPIO_u8IRQEnable

Service name	MGPIO_u8IRQEnable
Description	This Function Get value from selected pin
Syntax	`uint8_ MGPIO_u8IRQEnable (enu_pin_t Copy_enPinNum, enu_int_sens_type_t enu_int_sens_type, enu_int_sens_ctrl_t enu_int_sens_ctrl)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**Copy_enPinNum:** MGPIO_PINA_0 ~ MGPIO_PINF_7
	enu_int_sens_type: MGPIO_INT_EDGE_SENSETIVE ~ MGPIO_INT_LEVEL_SENSETIVE
	enu_int_sens_ctrl: MGPIO_INT_BOTH_EDGES - MGPIO_INT_FALL_E_LOW_L - MGPIO_INT_RIS_E_HIGH_L
Parameters (out)	**NONE**
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio _Interface.h

MGPIO_u8IRQDisable

Service name	**MGPIO_u8IRQDisable**
Description	This Function Get value from selected pin
Syntax	`uint8_ MGPIO_u8IRQDisable (enu_pin_t Copy_enPinNum)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**Copy_enPinNum:** MGPIO_PINA_0 ~ MGPIO_PINF_7
Parameters (out)	**NONE**
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio _Interface.h

MGPIO_u8SetCallBack

Service name	**MGPIO_u8SetCallBack**
Description	This Function Get value from selected pin
Syntax	`uint8_ MGPIO_u8SetCallBack (enu_pin_t Copy_enPinNum, ptr_func_t ptr_func)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**Copy_enPinNum:** MGPIO_PINA_0 ~ MGPIO_PINF_7
Parameters (out)	**ptr_func:** Address of application Function
Return	MGPIO_SUCCESS: *in case of successful operation*
	MGPIO_FAILED: *in case of failer operation*
Available via	mgpio _Interface.h

SYSTICK module

SYSTICK_u8Init

Service name	**SYSTICK_u8Init**
Description	Systick Timer Intialization
Syntax	`uint8_ SYSTICK_u8Init (st_systk_cfg_t* st_systk_cfg)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`st_systk_cfg): Address of struct Instance`
Parameters (out)	None
Return	`SUCCESS:` *in case of successful operation*
	`FAILED:` *in case of failer operation*
Available via	systick_Interface.h

SYSTICK_vidStart

Service name	**SYSTICK_vidStart**
Description	Start Timer count
Syntax	`uint8_ SYSTICK_vidStart(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**`Void`**
Parameters (out)	None
Return	`SUCCESS:` *in case of successful operation*
	`FAILED:` *in case of failer operation*
Available via	systick_Interface.h

SYSTICK_vidResetTimer

Service name	SYSTICK_vidResetTimer
Description	Systick Reset Counter and start from beginning
Syntax	`uint8_ SYSTICK_vidResetTimer (void)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`void`
Parameters (out)	`NONE`
Return	`SUCCESS:` *in case of successful operation*
	`FAILED:` *in case of failer operation*
Available via	systick_Interface.h

SYSTICK_vidStop

Service name	SYSTICK_vidStop
Description	Stop Systick Timer Counter
Syntax	`void SYSTICK_vidStop (void)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`void`
Parameters (out)	`NONE`
Return	`SUCCESS:` *in case of successful operation*
	`FAILED:` *in case of failer operation*
Available via	systick_Interface.h

SYSTICK_u8GetIntStatus

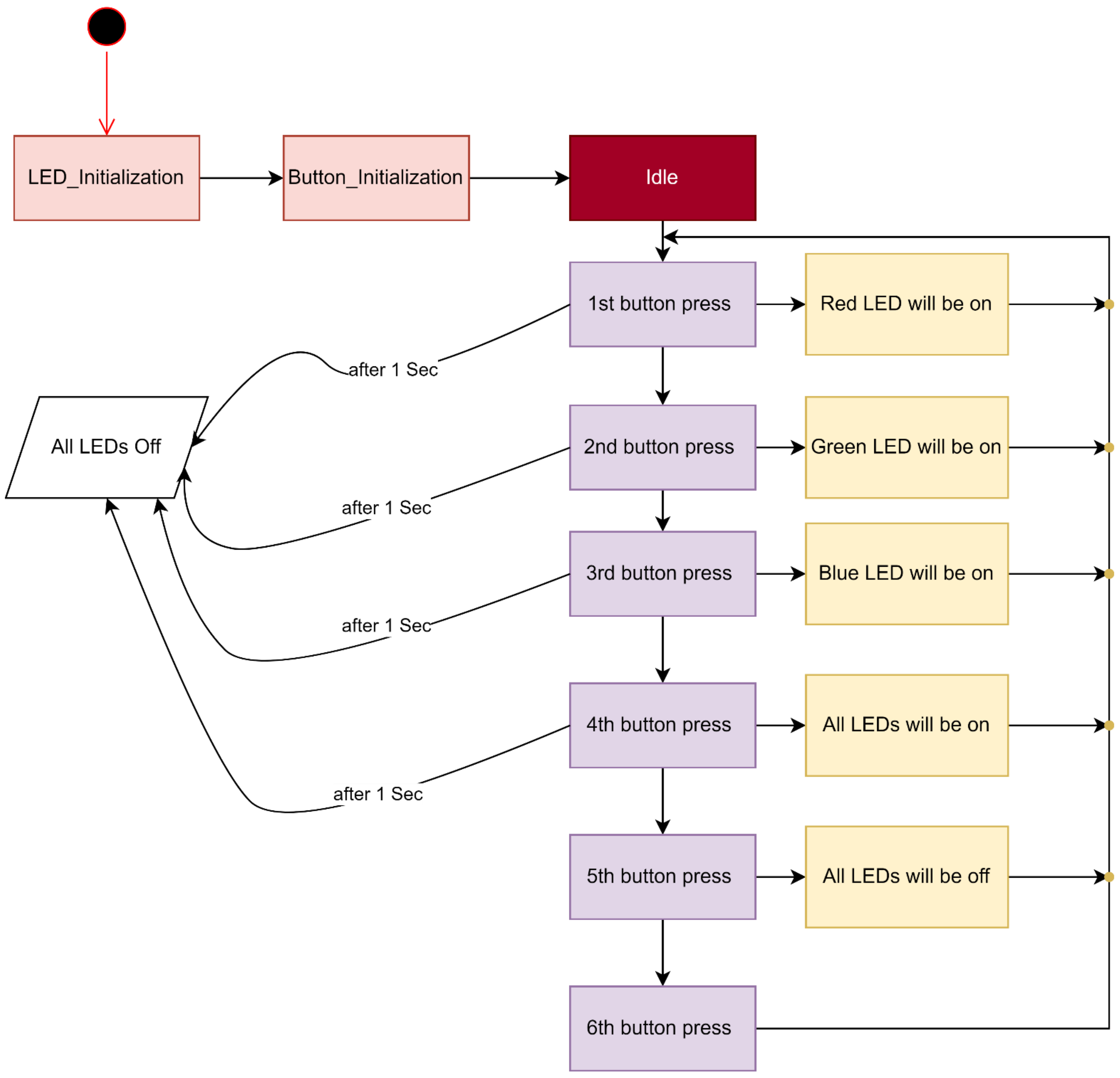
Service name	**SYSTICK_u8GetIntStatus**
Description	Get Systick Interrupt Flag to poll on it
Syntax	`uint8_ SYSTICK_u8GetIntStatus (uint8_* p_u8_int_status)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**NONE**
Parameters (out)	**p_u8_int_status:** Reference to variable where the value status store on it
Return	SUCCESS: *in case of successful operation*
	FAILED: *in case of failer operation*
Available via	systick_Interface.h

SYSTICK_u8DeInit

Service name	**SYSTICK_u8DeInit**
Description	De-Initialize Systick Timer
Syntax	`Void SYSTICK_u8DeInit (void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**void**
Parameters (out)	**p_u8_int_status:** Reference to variable where the value status store on it
Return	SUCCESS: *in case of successful operation*
	FAILED: *in case of failer operation*
Available via	systick_Interface.h

UML

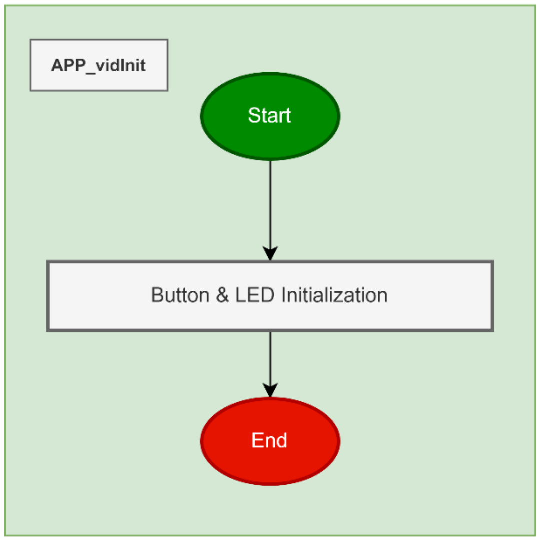
State Machine

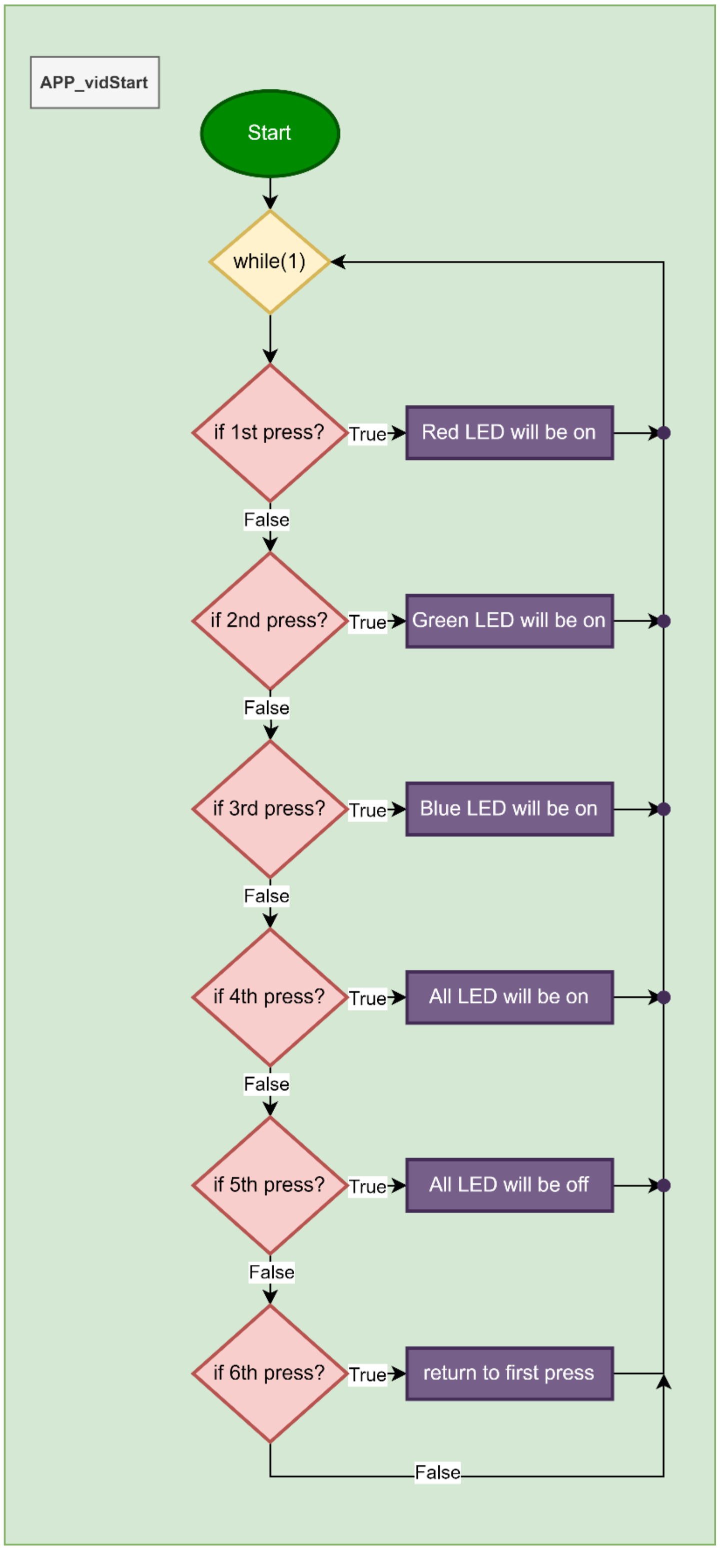


Low Level Design

Flowchart

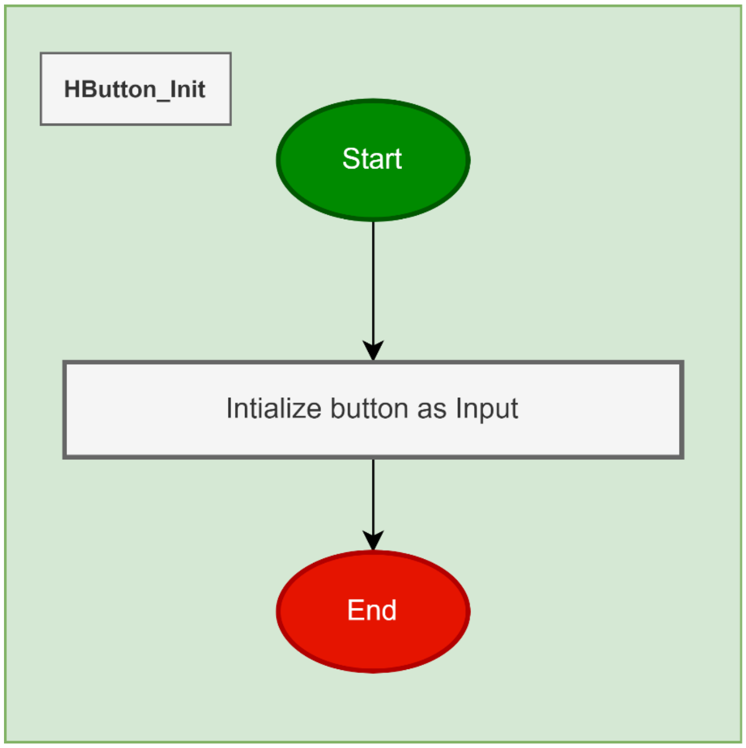
APP

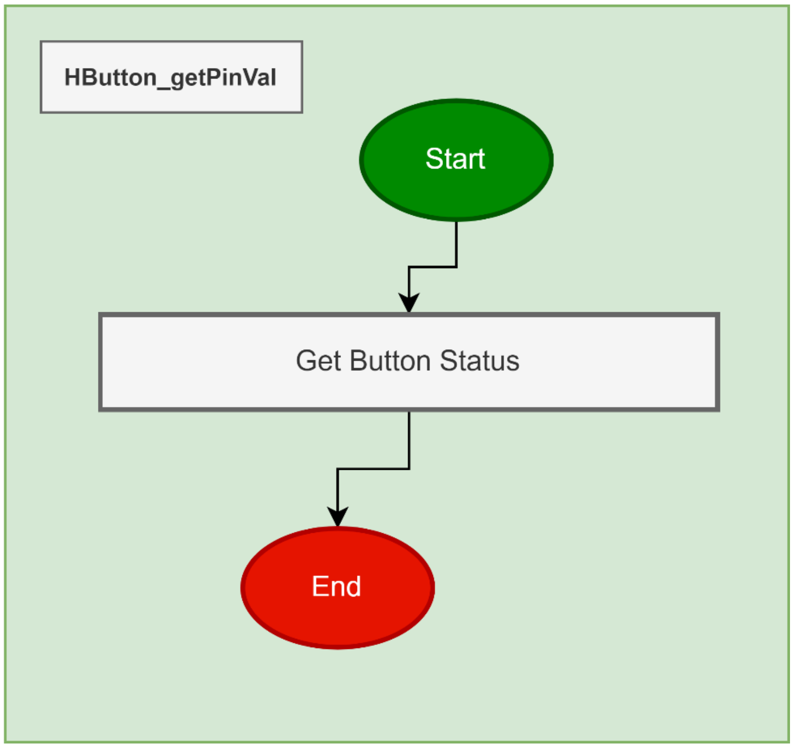




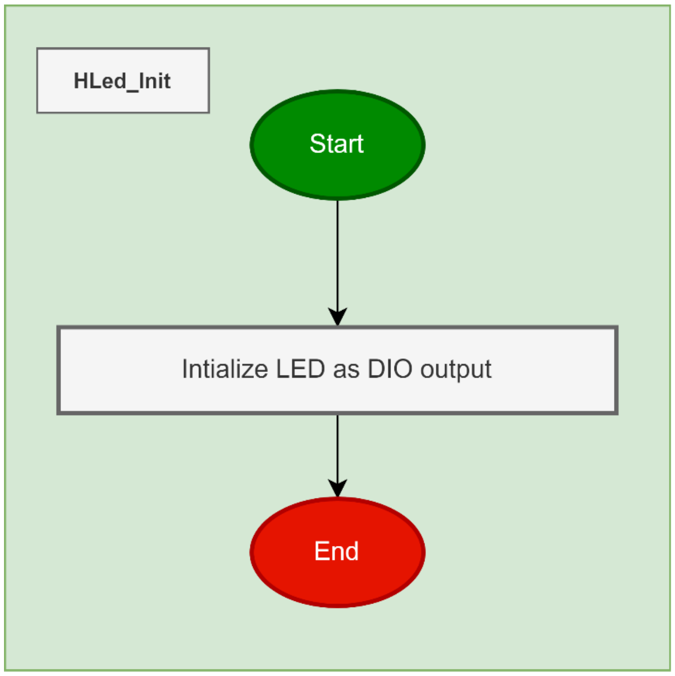
HAL

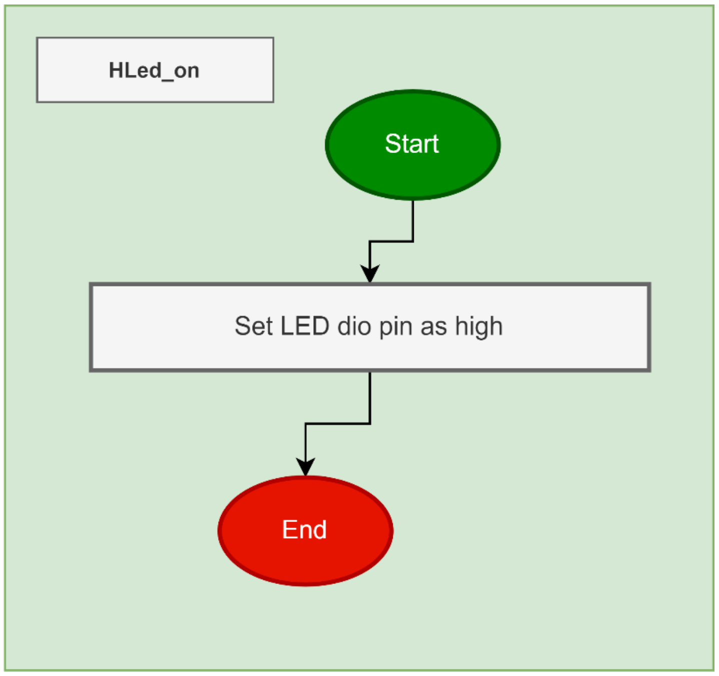
Button module

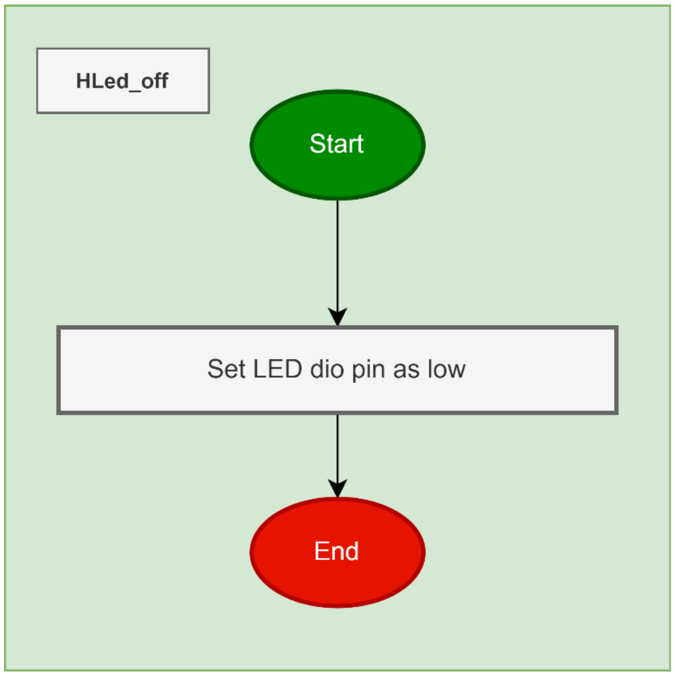


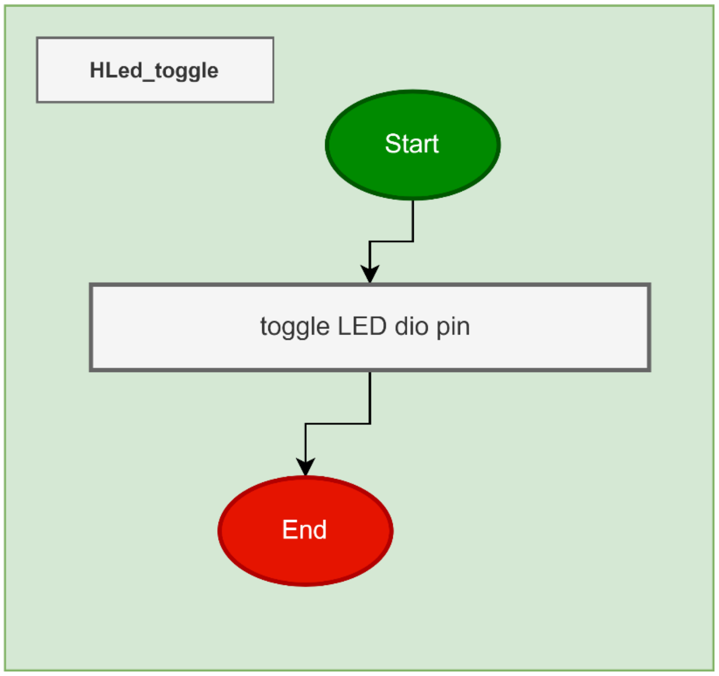


HLED module


HLed_Init
Start
Intialize LED as DIO output
End


HLed_on
Start
Set LED dio pin as high
End

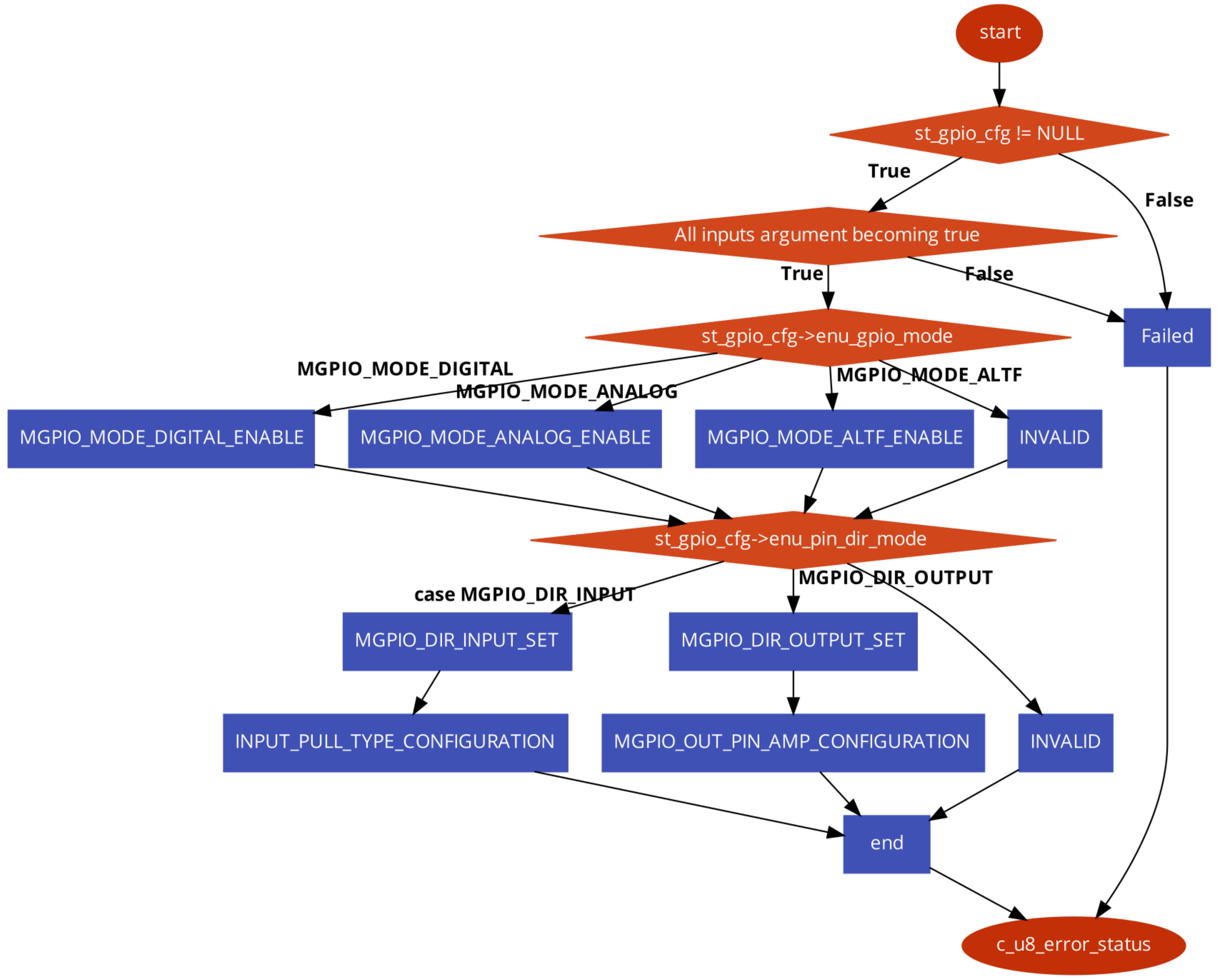

HLed_off
Start
Set LED dio pin as low
End


HLed_toggle
Start
toggle LED dio pin
End

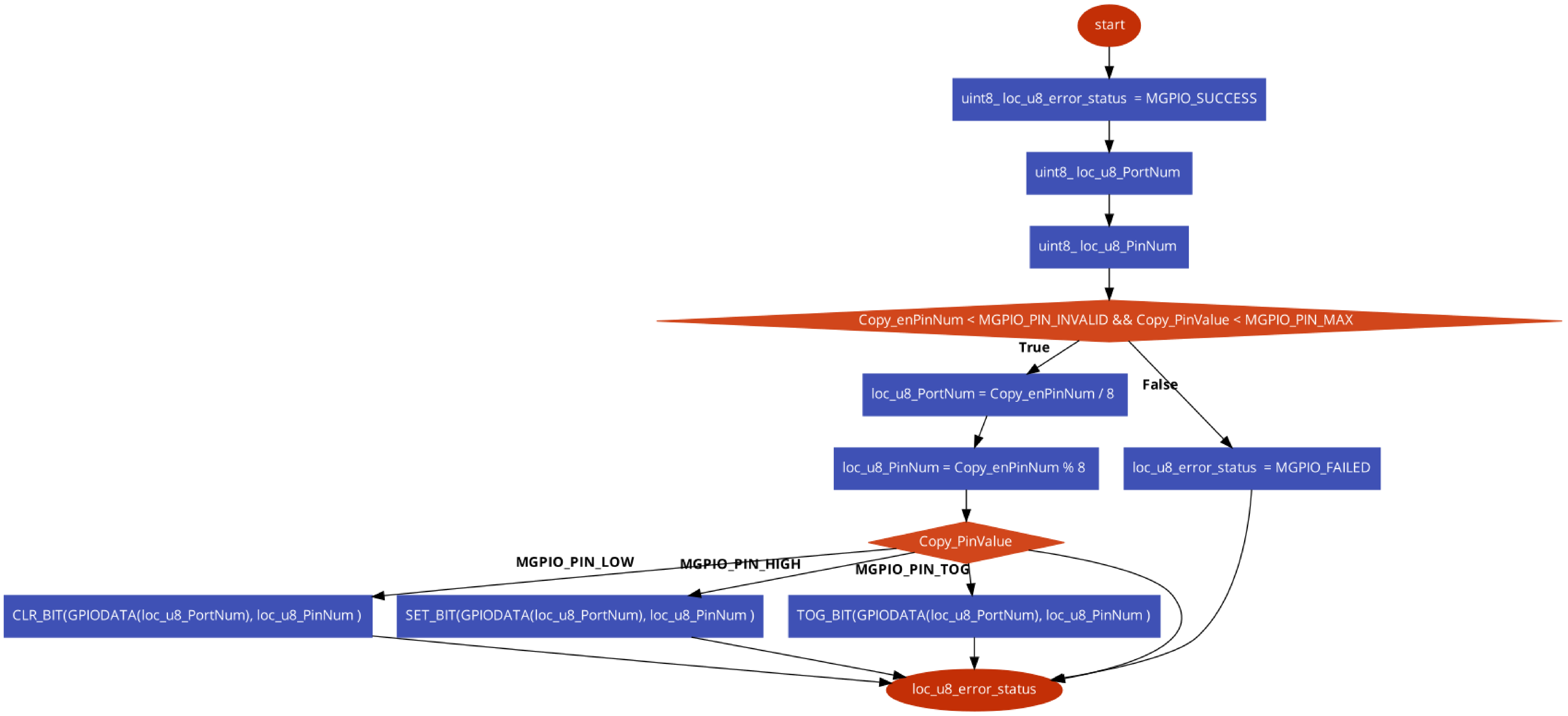
MCAL

GPIO module

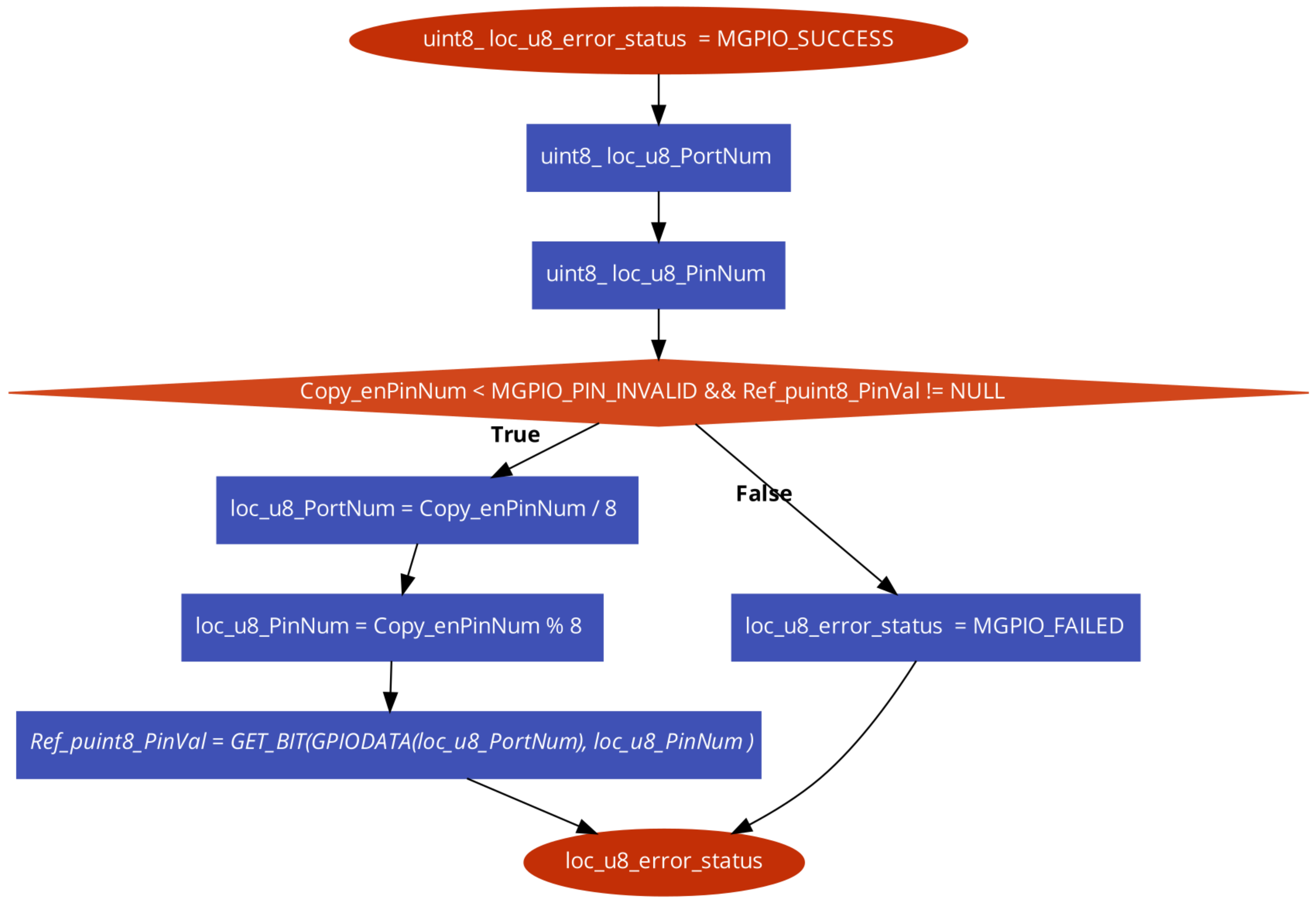
MGPIO_u8Init



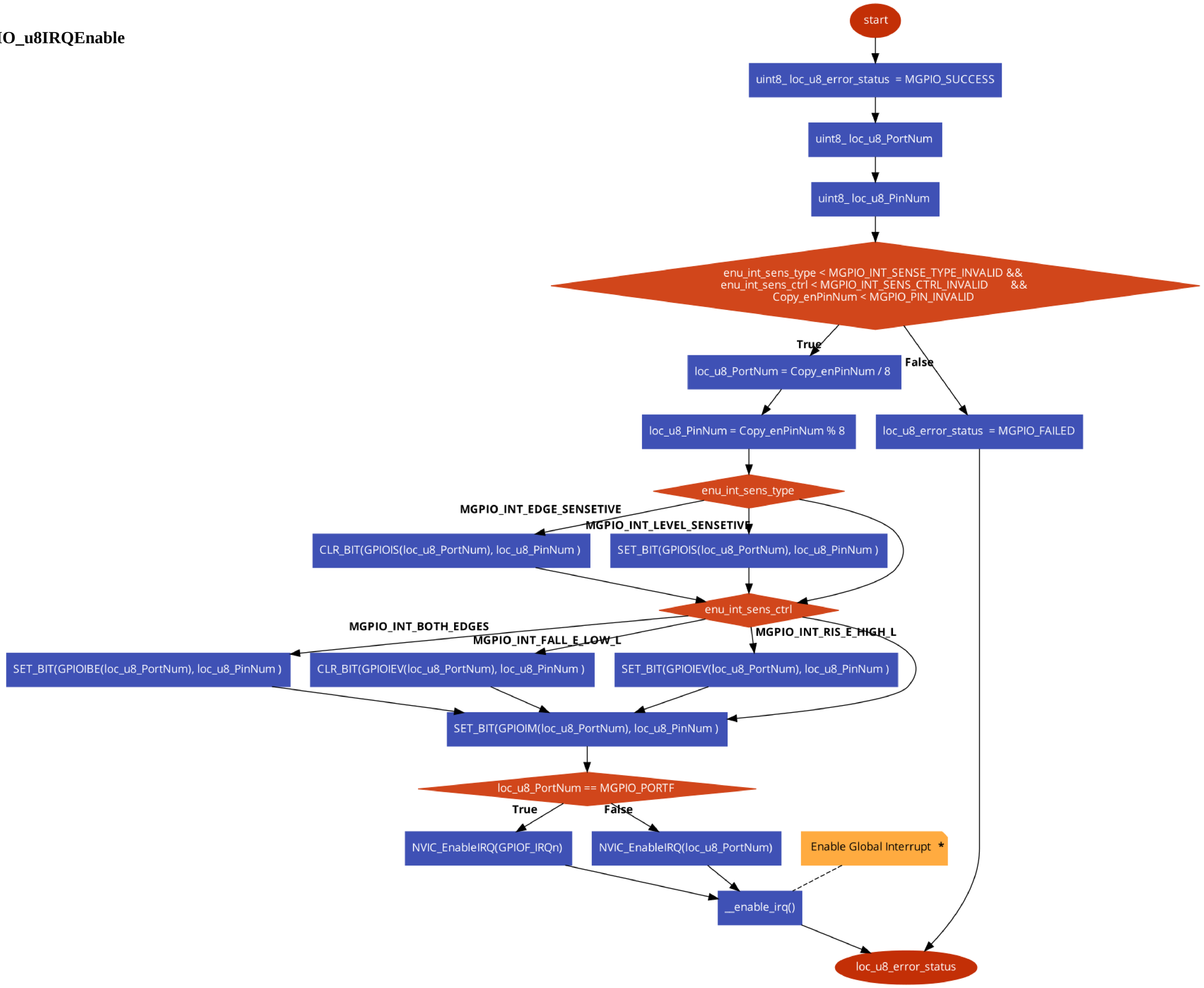
MGPIO_u8SetPinData


start
uint8_ loc_u8_error_status = MGPIO_SUCCESS
uint8_ loc_u8_PortNum
uint8_ loc_u8_PinNum
Copy_enPinNum < MGPIO_PIN_INVALID && Copy_PinValue < MGPIO_PIN_MAX
True
False
loc_u8_PortNum = Copy_enPinNum / 8
loc_u8_PinNum = Copy_enPinNum % 8
loc_u8_error_status = MGPIO_FAILED
Copy_PinValue
MGPIO_PIN_LOW
MGPIO_PIN_HIGH
MGPIO_PIN_TOG
CLR_BIT(GPIODATA(loc_u8_PortNum), loc_u8_PinNum)
SET_BIT(GPIODATA(loc_u8_PortNum), loc_u8_PinNum)
TOG_BIT(GPIODATA(loc_u8_PortNum), loc_u8_PinNum)
loc_u8_error_status

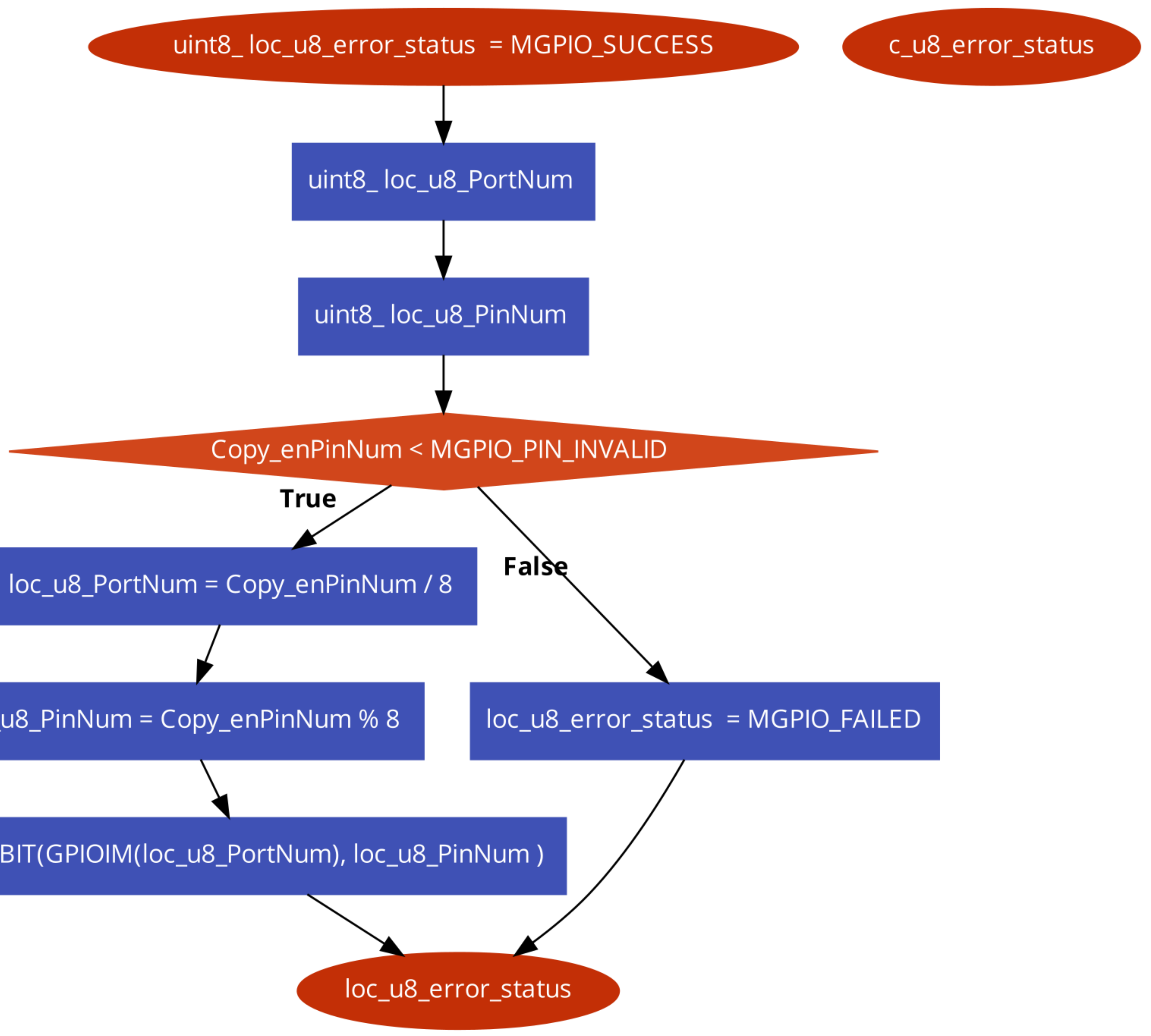
MGPIO_u8GetPinData


uint8_ loc_u8_error_status = MGPIO_SUCCESS
uint8_ loc_u8_PortNum
uint8_ loc_u8_PinNum
Copy_enPinNum < MGPIO_PIN_INVALID && Ref_puint8_PinVal != NULL
True
False
loc_u8_PortNum = Copy_enPinNum / 8
loc_u8_PinNum = Copy_enPinNum % 8
loc_u8_error_status = MGPIO_FAILED
Ref_puint8_PinVal = GET_BIT(GPIODATA(loc_u8_PortNum), loc_u8_PinNum)
loc_u8_error_status

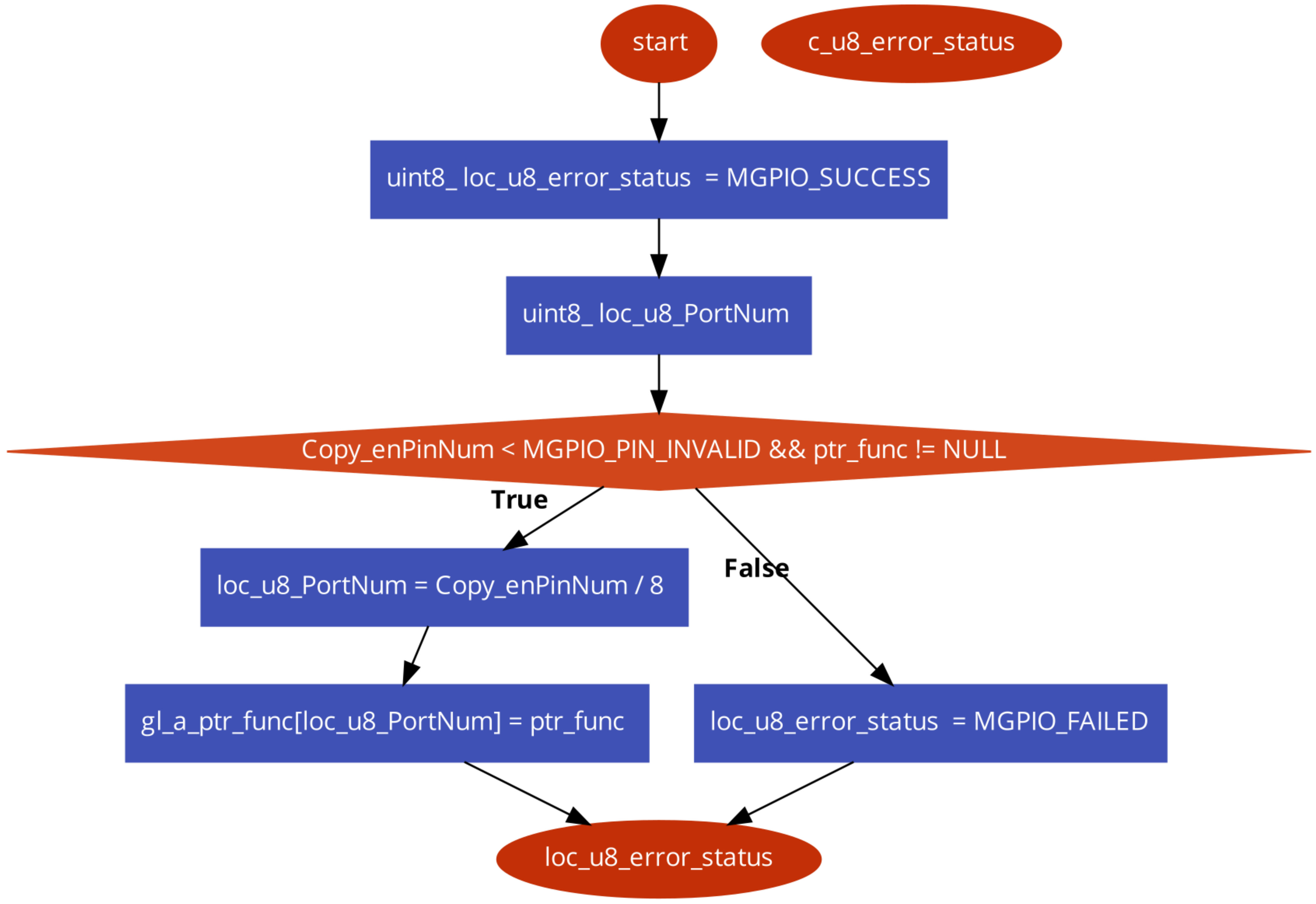
MGPIO_u8IRQEnable


start
uint8_ loc_u8_error_status = MGPIO_SUCCESS
uint8_ loc_u8_PortNum
uint8_ loc_u8_PinNum
enu_int_sens_type < MGPIO_INT_SENSE_TYPE_INVALID && enu_int_sens_ctrl < MGPIO_INT_SENS_CTRL_INVALID && Copy_enPinNum < MGPIO_PIN_INVALID
True
False
loc_u8_PortNum = Copy_enPinNum / 8
loc_u8_PinNum = Copy_enPinNum % 8
loc_u8_error_status = MGPIO_FAILED
enu_int_sens_type
MGPIO_INT_EDGE_SENSETIVE
MGPIO_INT_LEVEL_SENSETIVE
CLR_BIT(GPIOIS(loc_u8_PortNum), loc_u8_PinNum)
SET_BIT(GPIOIS(loc_u8_PortNum), loc_u8_PinNum)
enu_int_sens_ctrl
MGPIO_INT_BOTH_EDGES
MGPIO_INT_FALL_E_LOW_L
MGPIO_INT_RIS_E_HIGH_L
SET_BIT(GPIOIBE(loc_u8_PortNum), loc_u8_PinNum)
CLR_BIT(GPIOIEV(loc_u8_PortNum), loc_u8_PinNum)
SET_BIT(GPIOIEV(loc_u8_PortNum), loc_u8_PinNum)
SET_BIT(GPIOIM(loc_u8_PortNum), loc_u8_PinNum)
loc_u8_PortNum == MGPIO_PORTF
True
False
NVIC_EnableIRQ(GPIOF_IRQn)
NVIC_EnableIRQ(loc_u8_PortNum)
Enable Global Interrupt *
__enable_irq()
loc_u8_error_status

MGPIO_u8IRQDisable

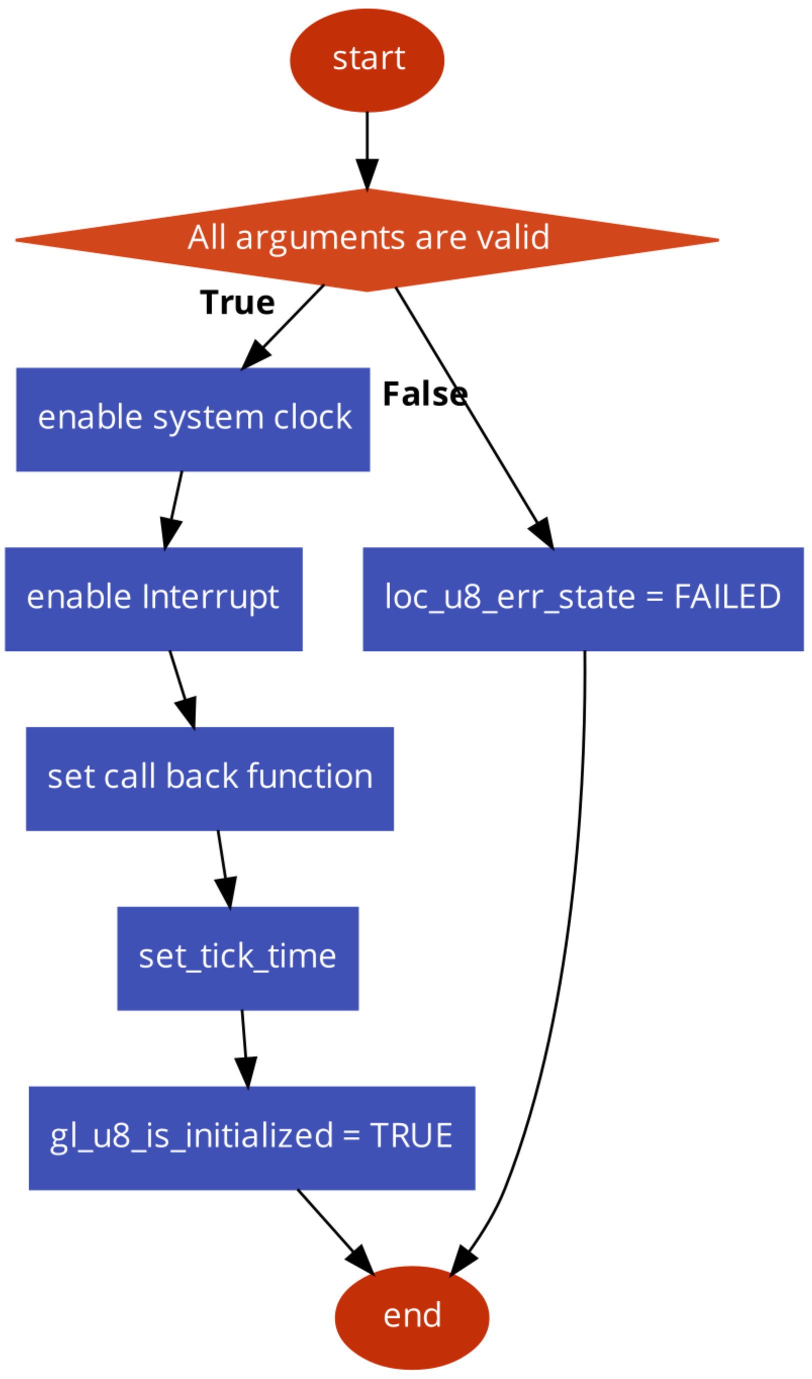

uint8_ loc_u8_error_status = MGPIO_SUCCESS
c_u8_error_status
uint8_ loc_u8_PortNum
uint8_ loc_u8_PinNum
Copy_enPinNum < MGPIO_PIN_INVALID
True
False
loc_u8_PortNum = Copy_enPinNum / 8
loc_u8_PinNum = Copy_enPinNum % 8
loc_u8_error_status = MGPIO_FAILED
CLR_BIT(GPIOIM(loc_u8_PortNum), loc_u8_PinNum)
loc_u8_error_status

MGPIO_u8SetCallBack

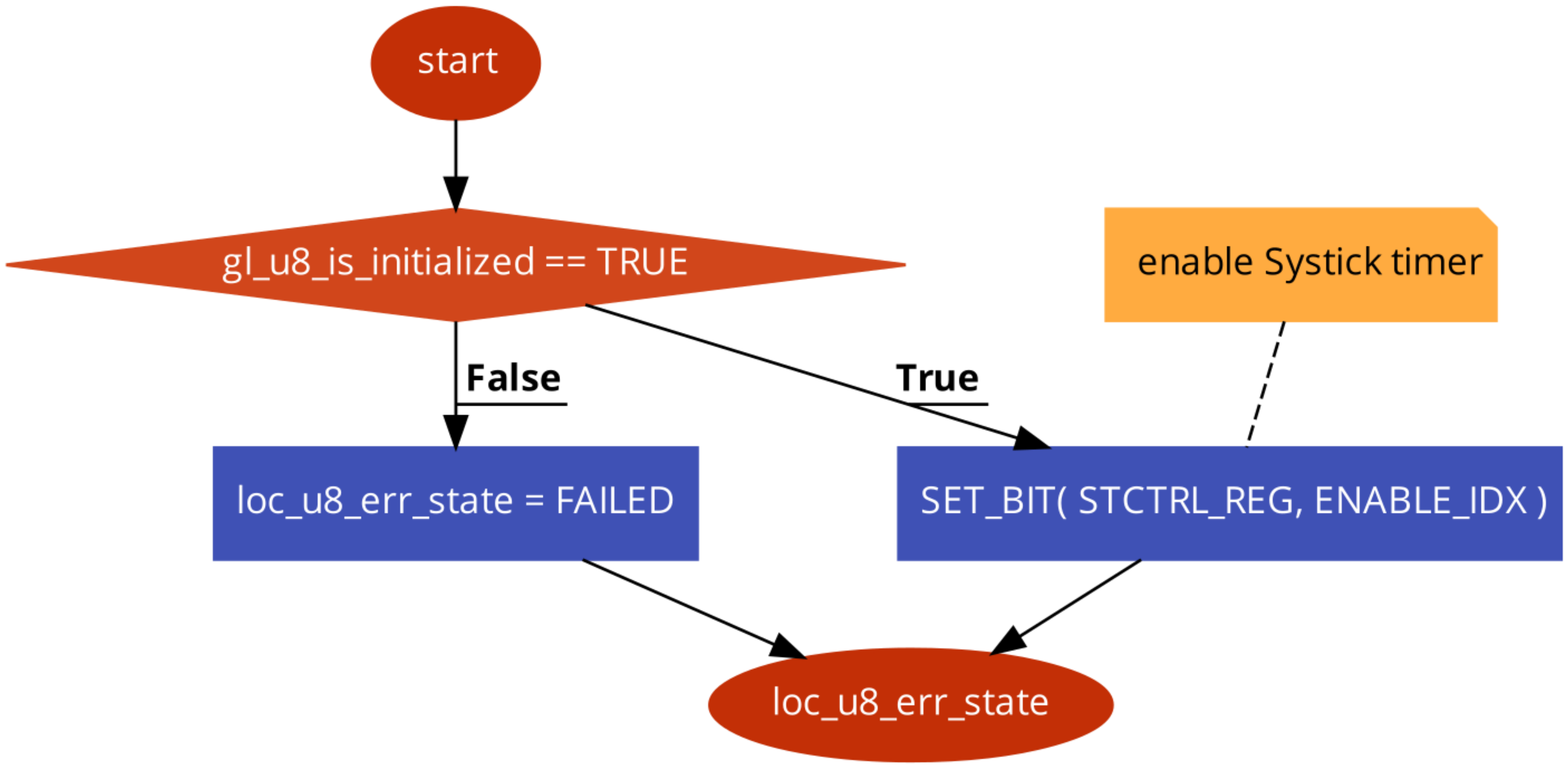

start
c_u8_error_status
uint8_ loc_u8_error_status = MGPIO_SUCCESS
uint8_ loc_u8_PortNum
Copy_enPinNum < MGPIO_PIN_INVALID && ptr_func != NULL
True
False
loc_u8_PortNum = Copy_enPinNum / 8
gl_a_ptr_func[loc_u8_PortNum] = ptr_func
loc_u8_error_status = MGPIO_FAILED
loc_u8_error_status

SYSTICK module

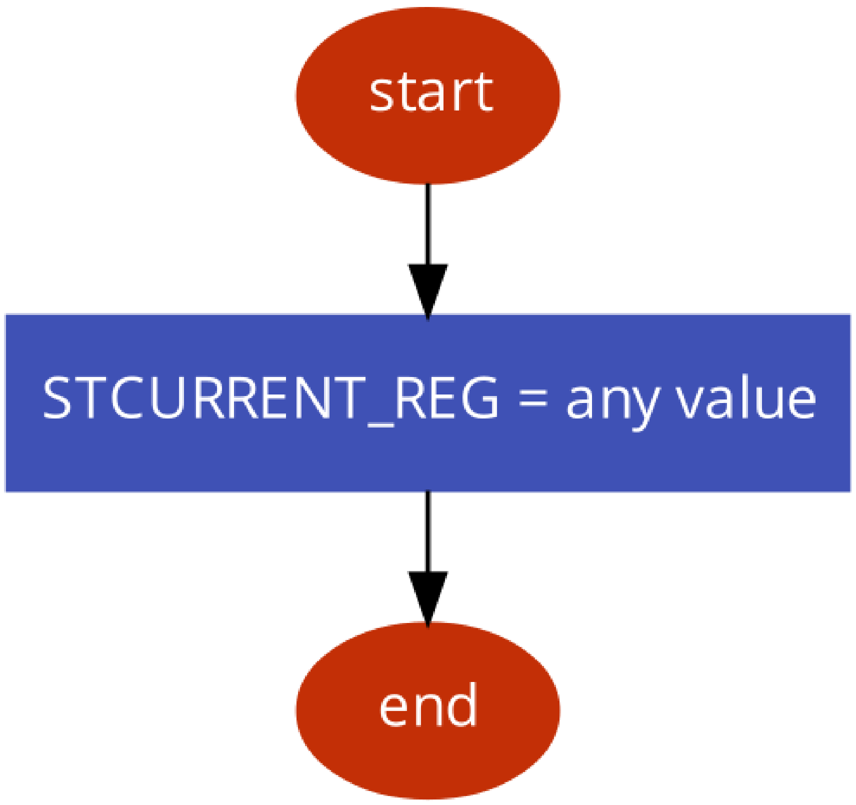
SYSTICK_u8Init



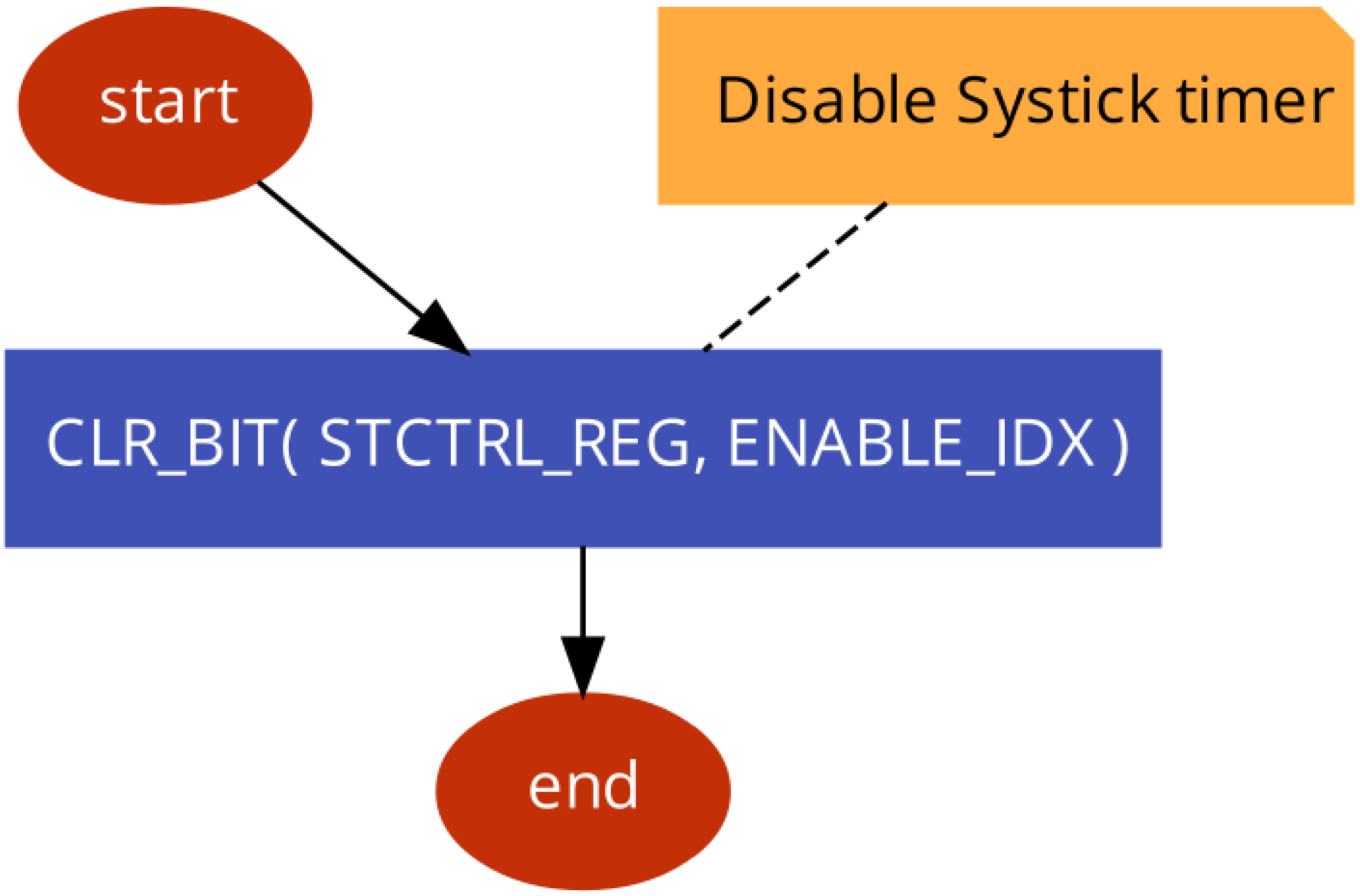
SYSTICK_vidStart


start
gl_u8_is_initialized == TRUE
False
True
enable Systick timer
loc_u8_err_state = FAILED
SET_BIT(STCTRL_REG, ENABLE_IDX)
loc_u8_err_state

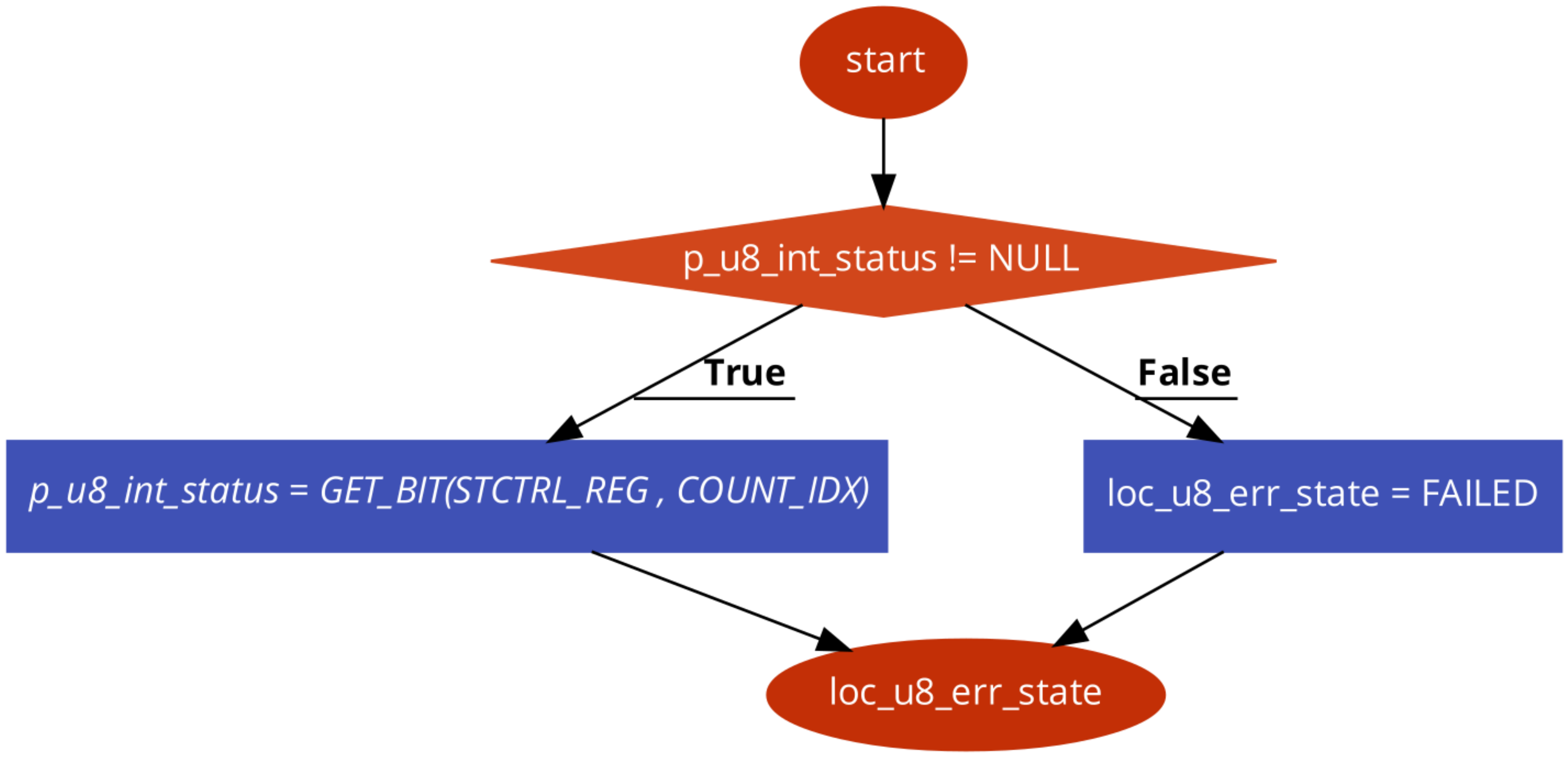
SYSTICK_vidResetTimer



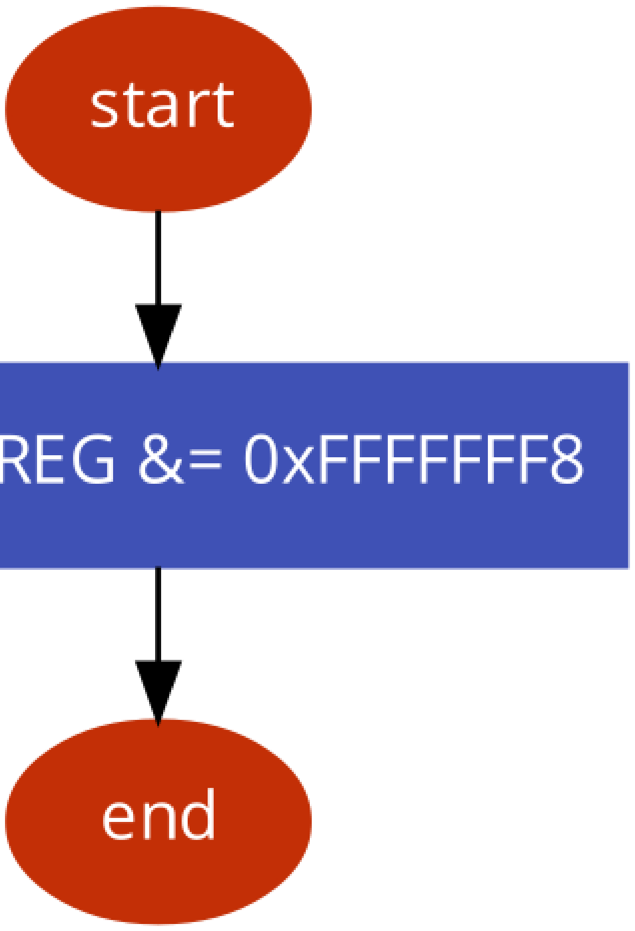
SYSTICK_vidStop



SYSTICK_u8GetIntStatus


start
p_u8_int_status != NULL
True
False
p_u8_int_status = GET_BIT(STCTRL_REG , COUNT_IDX)
loc_u8_err_state = FAILED
loc_u8_err_state

SYSTICK_u8DeInit



Pre-compiling configuration

MCAL

MGPIO module

GPIO_BUS_TYPE

Name	GPIO_BUS_TYPE
Type	MACRO
Description	Define GPIO_bus
Configuration	***GPIO_APB***
	GPIO_AHB
Found in	mgpio_private.h

Linking Configuration

MCAL

MGPIO module

st_gpio_cfg_t

Name	st_gpio_cfg_t	
Type	struct	
Description	GPIO pin configuration	
Members	*enu_pin*	
	enu_gpio_mode	
	enu_pin_dir_mode	
	un_gpio_conf	*enu_gpio_amp_mode*
		u8_input_pull_type
Found in	mgpio_Interface.h	

enu_pin_t

Name	enu_pin_t
Type	enum
Description	GPIO pin Selection
Configuration	*MGPIO_PINA_0 ~ MGPIO_PINA_7*
	MGPIO_PINB_0 ~ MGPIO_PINB_7
	MGPIO_PINC_0 ~ MGPIO_PINC_7
	MGPIO_PIND_0 ~ MGPIO_PIND_7
	MGPIO_PINE_0 ~ MGPIO_PINE_7
	MGPIO_PINF_0 ~ MGPIO_PINF_7
Found in	mgpio_Interface.h

enu_gpio_mode_t

Name	enu_gpio_mode_t
Type	enum
Description	GPIO Mode Selection
Configuration	*MGPIO_DIR_INPUT*
	MGPIO_DIR_OUTPUT
	MGPIO_DIR_INVALID
Found in	mgpio_Interface.h

enu_gpio_amp_mode_t

Name	enu_gpio_amp_mode_t
Type	enum
Description	GPIO Ampere mode Selection
Configuration	*MGPIO_OPEN_DRAIN*
	MGPIO_MAMP_2
	MGPIO_MAMP_4
	MGPIO_MAMP_8
	MGPIO_MAMP_INVALID
Found in	mgpio_Interface.h

enu_gpio_int_t

Name	enu_gpio_int_t
Type	enum
Description	GPIO Interrupt mode Selection
Configuration	*MGPIO_INT_ENABLE*
	MGPIO_INT_DISABLE
	MGPIO_INT_INVALID
Found in	mgpio_Interface.h

enu_int_sens_type_t

Name	enu_int_sens_type_t
Type	enum
Description	GPIO Ampere mode Selection
Configuration	*MGPIO_INT_EDGE_SENSETIVE*
	MGPIO_INT_LEVEL_SENSETIVE
	MGPIO_INT_SENSE_TYPE_INVALID
Found in	mgpio_Interface.h

enu_int_sens_ctrl_t

Name	enu_int_sens_ctrl_t
Type	enum
Description	GPIO Ampere mode Selection
Configuration	*MGPIO_INT_BOTH_EDGES*
	MGPIO_INT_FALL_E_LOW_L
	MGPIO_INT_RIS_E_HIGH_L
	MGPIO_INT_SENS_CTRL_INVALID
Found in	mgpio_Interface.h

SYSTICK module

st_systk_cfg_t

Name	st_systk_cfg_t
Type	struct
Description	SYSTICK configuration
Members	*en_systck_clk_src*
	en_systck_int
	ptr_func
	u32_time_ms
Found in	systick_Interface.h

en_systck_int_t

Name	en_systck_int_t
Type	enum
Description	IRQ Enable / Disable
Configuration	*SYSTK_IRQ_DISABLE*
	SYSTK_IRQ_ENABLE
Found in	systick_Interface.h

en_systck_clk_src_t

Name	en_systck_clk_src_t
Type	enum
Description	SYSTICK clock Source Selection
Configuration	*SYSTK_PIOSC*
	SYSTK_SYSTEM_CLK
Found in	systick_Interface.h