LED V3 System Design

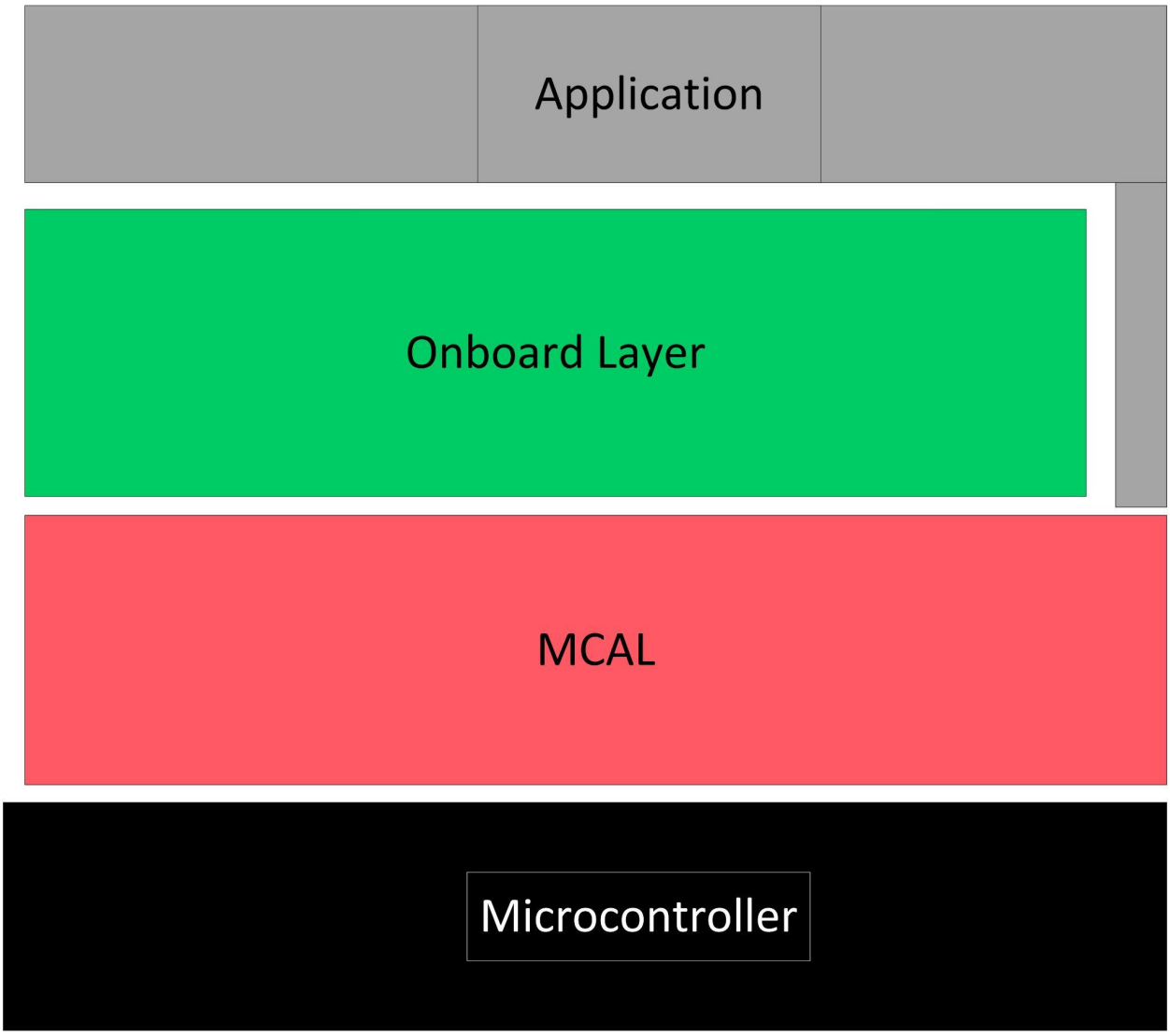
By : Mahmoud Adel Matarawy
Email : mahmoudsarhan02672@gmail.com

Detailed Requirements

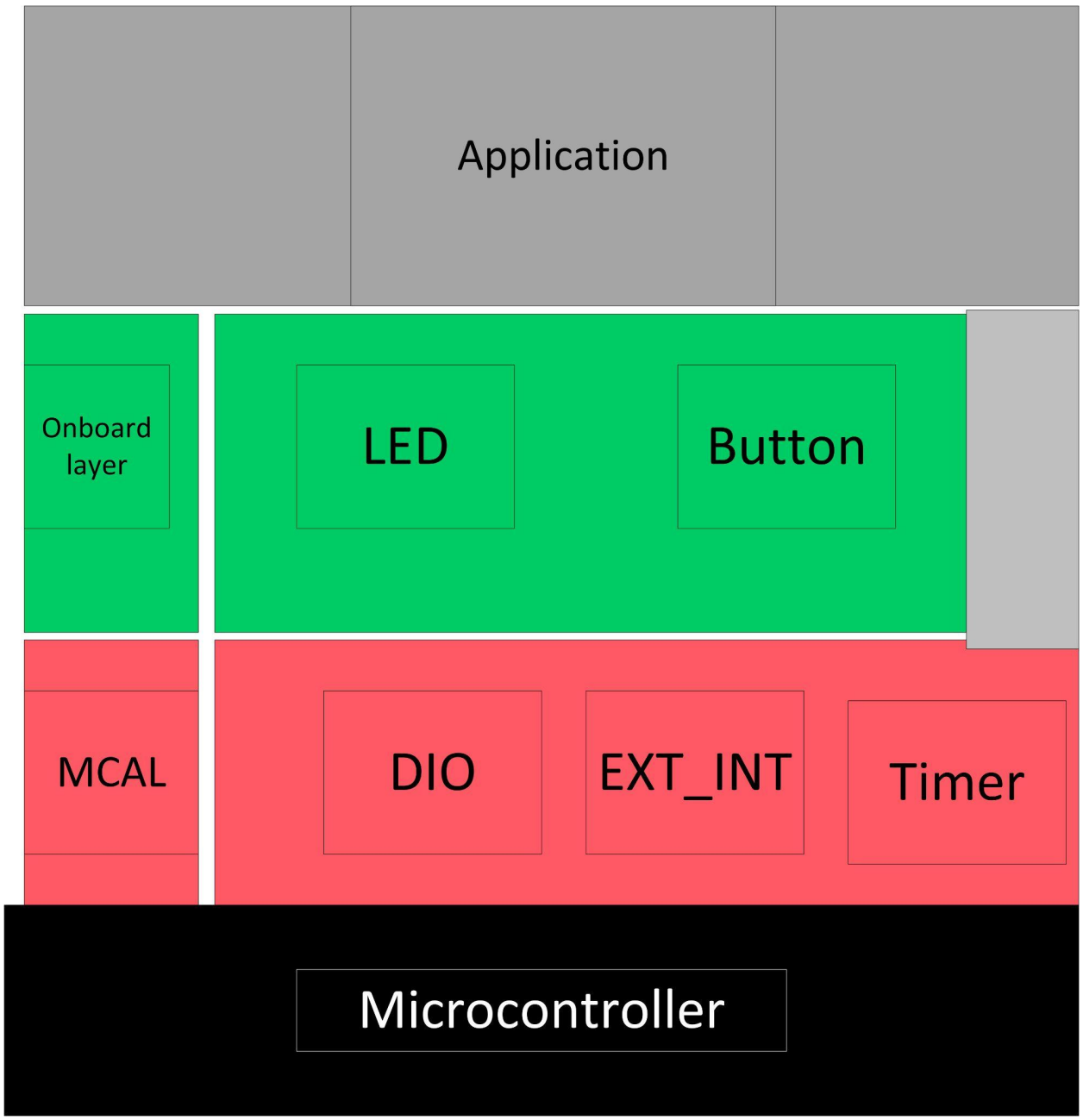
Read System Requirements Specifications

1. *Description*
    1. *Hardware Requirements*
        1. *Four LEDs (LED0, LED1, LED2, LED3)*
        2. *Two buttons (BUTTON0 and BUTTON1)*
    2. *Software Requirements*
        1. *Initially, all LEDs are OFF*
        2. *Once BUTTON0 is pressed, LED0 will blink with BLINK_1 mode*
        3. *Each press further will make another LED blinks BLINK_1 mode*
        4. *At the fifth press, LED0 will changed to be OFF*
        5. *Each press further will make only one LED is OFF*
        6. *This will be repeated forever*
        7. *The sequence is described below*
            1. *Initially (OFF, OFF, OFF, OFF)*
            2. *Press 1 (BLINK_1, OFF, OFF, OFF)*
            3. *Press 2 (BLINK_1, BLINK_1, OFF, OFF)*
            4. *Press 3 (BLINK_1, BLINK_1, BLINK_1, OFF)*
            5. *Press 4 (BLINK_1, BLINK_1, BLINK_1, BLINK_1)*
            6. *Press 5 (OFF, BLINK_1, BLINK_1, BLINK_1)*
            7. *Press 6 (OFF, OFF, BLINK_1, BLINK_1)*
            8. *Press 7 (OFF, OFF, OFF, BLINK_1)*
            9. *Press 8 (OFF, OFF, OFF, OFF)*
            10. *Press 9 (BLINK_1, OFF, OFF, OFF)*
        8. *When BUTTON1 has pressed the blinking on and off durations will be changed*
            1. *No press → BLINK_1 mode (ON: 100ms, OFF: 900ms)*
            2. *First press → BLINK_2 mode (ON: 200ms, OFF: 800ms)*
            3. *Second press → BLINK_3 mode (ON: 300ms, OFF: 700ms)*
            4. *Third press → BLINK_4 mode (ON: 500ms, OFF: 500ms)*
            5. *Fourth press → BLINK_5 mode (ON: 800ms, OFF: 200ms)*
            6. *Fifth press → BLINK_1 mode*
        9. *USE EXTERNAL INTERRUPTS*

Layered architecture



System modules



APIs

MCAL APIs

Timer API :

Type definitions:

- Timer_Errors

Name	Timer_Errors	
Type	Enumeration	
Range	Timer_E_OK	0x00
	Timer_E_TRANSITION	0x01
	Timer_E_PARAM_POINTER	0x02
	Timer_E_INIT_FAILED	0x03
	Timer_E_InvalidValue	0x04
Description	Timer_Errors	
Available via	timer_shared.h	

- Timer_Number

Name	Timer_Number	
Type	Enumeration	
Range	Timer_0	0x00

	Timer_1	0x01
	Timer_2	0x02
Description	Timer_Number ID	
Available via	timer_shared.h	

- Timer_Status

Name	Timer_Status	
Type	Enumeration	
Range	Timer_S_Ready	0x01
	Timer_S_UnInit	0x02
Description	Timer_Status ID	
Available via	timer_shared.h	

Services affecting the hardware unit:

- timer_delay_50ms

Service name	timer_delay_50ms	
Syntax	Timer_Errors timer_delay_50ms(Timer_Number num);	
Parameters (in)	num	Timer ID
Return	Timer_Errors	
Description	This Function enable interrupt every 50ms	
Available via	timer_app.h	

  - This function shall return Timer_E_InvalidValue if num is invalid

- Timer_Init

Service name	Timer_Init	
Syntax	Timer_Errors Timer_Init(Timer_Number Timer_Num);	
Parameters (in)	Timer_Num	Timer ID
Return	Timer_Errors	
Description	This Function initialize and start the timer	
Available via	timer.h	

  - This function shall return Timer_E_InvalidValue if Timer_Num is invalid
  - This function shall return Timer_E_TRANSITION if timer state is Ready

- Timer_Set

Service name	Timer_Set	
Syntax	Timer_Errors Timer_Set(Timer_Number Timer_Num);	
Parameters (in)	Timer_Num	Timer ID
Return	Timer_Errors	
Description	This Function sets the timer counter value	
Available via	timer.h	

  - This function shall return Timer_E_InvalidValue if Timer_Num is invalid
  - This function shall return Timer_E_TRANSITION if timer state is UnInit

External Interrupt API :

Type definitions:

- EXT_INT_ID_TYPE

Name	EXT_INT_ID_TYPE	
Type	Enumeration	
Range	INT_0_ID	0x00
	INT_1_ID	0x01
	INT_2_ID	0x02
Description	EXT_INT_ID_TYPE	
Available via	EXT_INT.h	

- EXT_INT_MODE_TYPE

Name	EXT_INT_MODE_TYPE	
Type	Enumeration	
Range	EXT_INT_FALLING_EDGE	0x00
	EXT_INT_RISING_EDGE	0x01
Description	EXT_INT_MODE_TYPE	
Available via	EXT_INT.h	

- EXT_INT_ERR_TYPE

Name	EXT_INT_ERR_TYPE	
Type	Enumeration	
Range	EXT_INT_ERR_OK	0x00
	EXT_INT_ERR_OutOfRange	0x01
Description	EXT_INT_ERR_TYPE	
Available via	EXT_INT.h	

Services affecting the hardware unit:

- eXT_INT_Enable

Service name	eXT_INT_Enable	
Syntax	EXT_INT_ERR_TYPE eXT_INT_Enable(EXT_INT_ID_TYPE id, EXT_INT_MODE_TYPE mode);	
Parameters (in)	id	Interrupt ID
	mode	Rising edge or falling edge
Return	EXT_INT_ERR_TYPE	EXT_INT_ERR_OK EXT_INT_ERR_OutOfRange
Description	This Function enable an external interrupt	

  - This function shall return EXT_INT_ERR_OutOfRange if id or mode is invalid.

DIO API :

Type definitions:

- Dio_ChannelType

Name	Dio_ChannelType
Type	Enumeration
Range	Shall contain all pins ID
Description	Dio_ChannelType
Available via	DIO_Config.h

- Dio_PortType

Name	Dio_PortType
Type	Enumeration

Range	Shall contain all ports ID
Description	Dio_PortType
Available via	DIO_Config.h

- DIO_Errors

Name	DIO_Errors		
Type	Enumeration		
Range	DIO_E_OK	0x00	DIO error OK
	DIO_InvalidPin	0x01	DIO error, invalid pin number.
Description	DIO Errors		
Available via	DIO.h		

- Dio_LevelType

Name	Dio_LevelType		
Type	Enumeration		
Range	STD_LOW	0x00	Physical state 0V
	STD_HIGH	0x01	Physical state 5V or 3.3V.
Description	Dio_LevelType		
Available via	DIO.h		

- Dio_DIRType

Name	Dio_DIRType

Type	Enumeration		
Range	STD_INPUT	0x00	Set pin as input pin
	STD_OUTPUT	0x01	Set pin as output pin
Description	Dio_DIRType		
Available via	DIO.h		

Services affecting the hardware unit:

- Dio_ReadChannel

Service name	Dio_ReadChannel		
Syntax	DIO_Errors Dio_ReadChannel(Dio_ChannelType ChannelId, Dio_LevelType* level);		
Parameters (in)	ChannelId	Channel ID	
	level	Pointer to store the level	STD_HIGH
			STD_LOW
Return	DIO_Errors	DIO_E_OK	
		DIO_InvalidPin	
Description	This Function gets the level of the pin		

  - This function shall return DIO_InvalidPin if pin number is invalid.

- Dio_WriteChannel

Service name	Dio_WriteChannel	
Syntax	DIO_Errors Dio_WriteChannel(Dio_ChannelType ChannelId, Dio_LevelType level);	
Parameters (in)	ChannelId	Channel ID

	level	Value to be set	STD_HIGH
			STD_LOW
Return	DIO_Errors		DIO_E_OK
			DIO_InvalidPin
Description	This Function gets the level of the pin		

- This function shall return DIO_InvalidPin if pin number is invalid.

- Dio_ChannelSetDIR

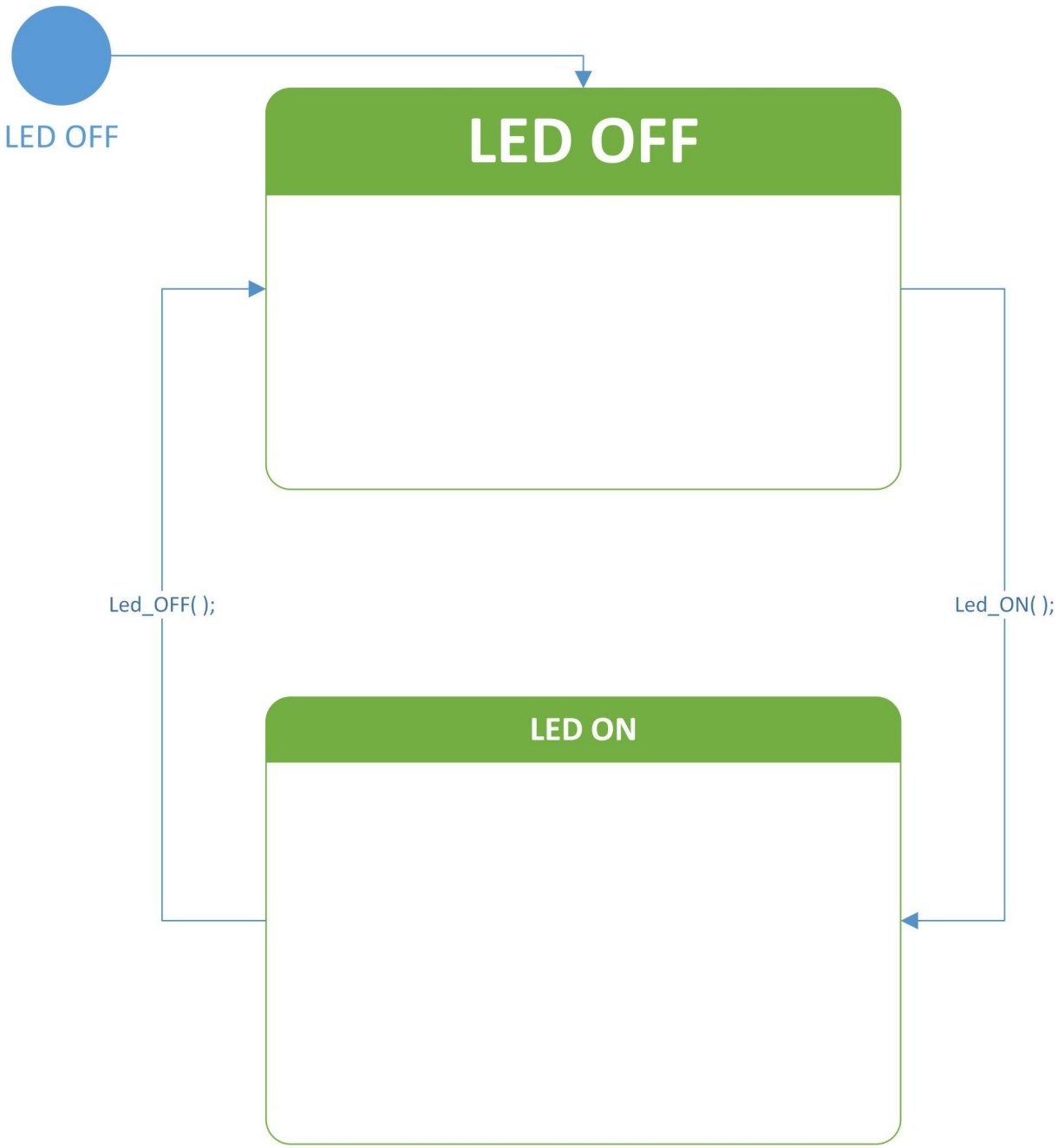
Service name	Dio_ChannelSetDIR		
Syntax	DIO_Errors Dio_ChannelSetDIR(Dio_ChannelType ChannelId, Dio_DIRType dir);		
Parameters (in)	ChannelId	Channel ID	
	dir	Value to be set	STD_INPUT
			STD_OUTPUT
Return	DIO_Errors		DIO_E_OK
			DIO_InvalidPin
Description	This Function sets the Direction of the pin		

- This function shall return DIO_InvalidPin if pin number is invalid.

Onboard APIs

LED API:

State machine :



Type definitions:

- LED_Config_Type

Name	LED_Config_Type
Type	Structure
Description	This is the type of the external data structure containing the overall configuration data for the LED API
Available via	led_types.h

- LED_STATE_type

Name	LED_STATE_type		
Type	Enumeration		
Range	LED_OFF	0x00	LED OFF STATE
	LED_ON	0x01	LED ON STATE
Description	LED State Enum		
Available via	led.h		

- LED_ERROR_type

Name	LED_ERROR_type		
Type	Enumeration		
Range	LED_OK	0x00	ERROR OK
	LED_UNDEFINED	0x01	LED ID not defined
Description	LED Error Enum		
Available via	led.h		

- LED_ID_type

Name	LED_ID_type
Type	Enumeration

Range	LED_1	0x01	LED 1
	LED_2	0x02	LED 2
	LED_3	0x03	LED 3
	LED_4	0x04	LED 4
Description	LED ID Enum		
Available via	led.h		

Services affecting the hardware unit:

- led_Init

Service name	led_Init
Syntax	void led_Init(void);
Return	None
Description	This Function Initialize the LED module

- led_OFF

Service name	led_OFF		
Syntax	LED_ERROR_type led_OFF(LED_ID_type led);		
Parameters (in)	led	LED_1	0x01
		LED_2	0x02
		LED_3	0x03
		LED_4	0x04

Return	LED_ERROR _type	LED_OK	0x00
		LED_UNDEFINED	0x01
Description	This Function Sets a LED OFF.		

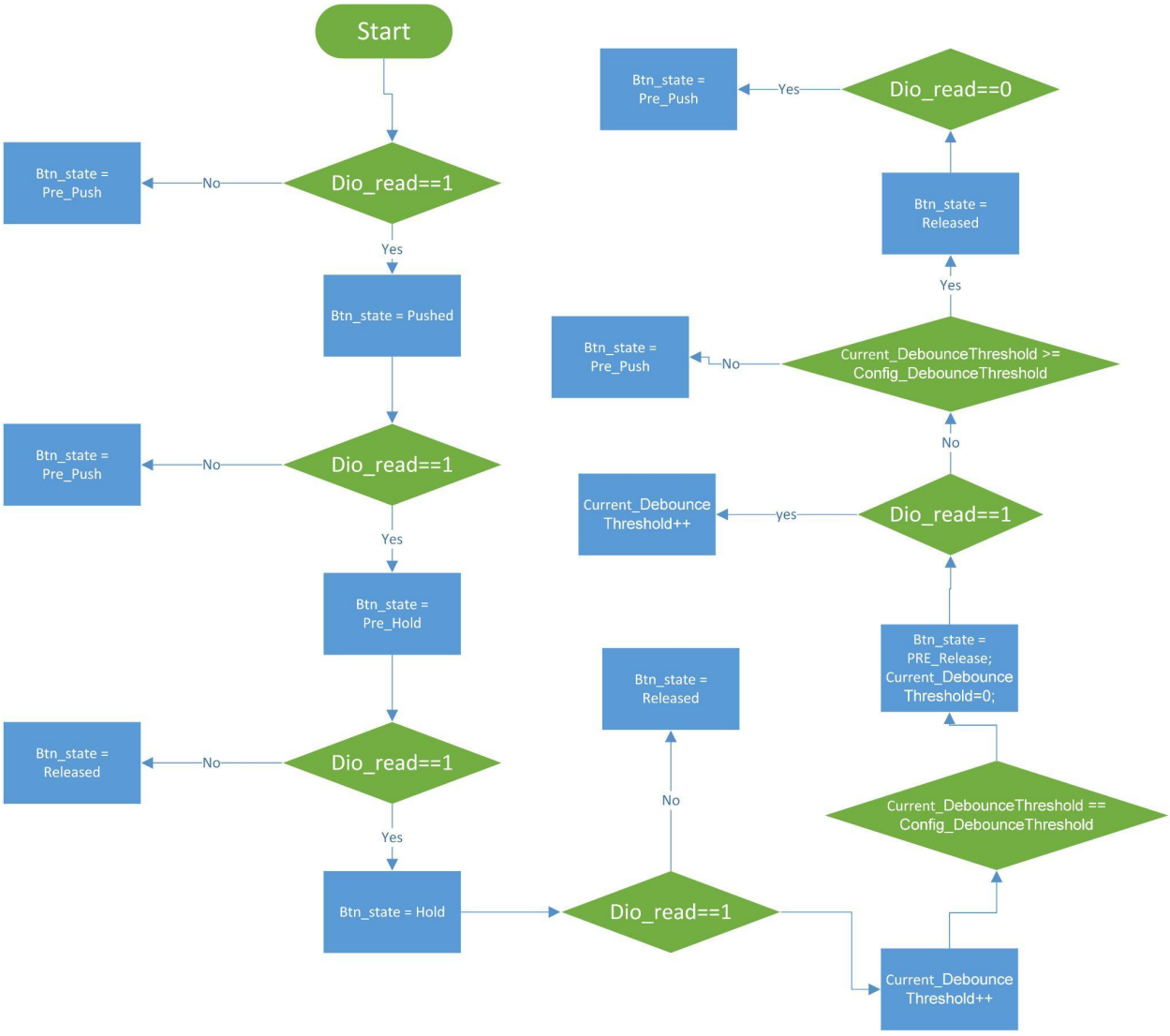
- led_ON

Service name	led_ON		
Syntax	LED_ERROR_type led_ON(LED_ID_type led);		
Parameters (in)	led	LED_1	0x01
		LED_2	0x02
		LED_3	0x03
		LED_4	0x04
Return	LED_ERRO R_type	LED_OK	0x00
		LED_UNDEFINED	0x01
Description	This Function Sets a LED ON.		

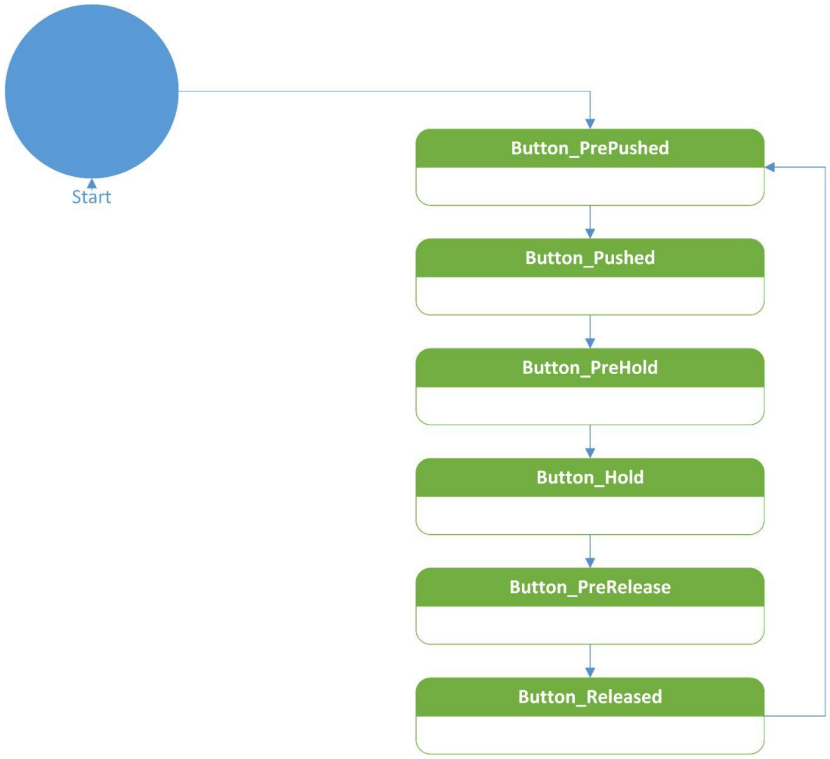
Button API:

Flowchart :

button_Main_Task flowchart


Start
Dio_read==1
No
Btn_state = Pre_Push
Yes
Btn_state = Pushed
Dio_read==1
No
Btn_state = Pre_Push
Yes
Btn_state = Pre_Hold
Dio_read==1
No
Btn_state = Released
Yes
Btn_state = Hold
Dio_read==1
No
Btn_state = Released
Current_DebounceThreshold == Config_DebounceThreshold
Current_Debounce Threshold++
Btn_state = PRE_Release; Current_Debounce Threshold=0;
Dio_read==1
yes
Current_Debounce Threshold++
Current_DebounceThreshold >= Config_DebounceThreshold
No
Btn_state = Pre_Push
Yes
Btn_state = Released
Dio_read==0
Yes
Btn_state = Pre_Push

State machine :



Type definitions:

- Button_configType

Name	Button_configType
Type	Structure
Description	This is the type of the external data structure containing the overall configuration data for the Button API
Available via	Button_Types.h

- Button_LevelType

Name	Button_LevelType		
Type	Enumeration		
Range	BT_PUSH_LEVEL	0x00	Push Level
	BT_RELEASE_LEVEL	0x01	Release Level

Description	Button Level Enum
Available via	Button_Types.h

- Button_StateType

Name	Button_StateType		
Type	Enumeration		
Range	BT_PRE_PUSH	0x00	Pre Push Level
	BT_PUSHED	0x01	Pushed Level
	BT_PRE_HOLD	0x02	Pre Hold Level
	BT_HOLD	0x03	Hold Level
	BT_PRE_RELEASE	0x04	Pre Release Level
	BT_RELEASED	0x05	Released Level
	BT_UNDEFINED	0x06	Undefined
Description	Button state Enum		
Available via	Button.h		

- Button_IdType

Name	Button_IdType		
Type	Enumeration		
Range	Button_Start	0x00	Start Button
Description	Button ID Enum		
Available via	Button.h		

Services affecting the hardware unit:

- getButtonState

Service name	getButtonState	
Syntax	Button_StateTyp getButtonState(Button_IdType enmButtonId);	
Parameters (in)	enmButtonId	Start 0x00
Return	Button_StateTyp	BT_PRE_PUSH BT_PUSHED BT_PRE_HOLD BT_HOLD BT_PRE_RELEASE BT_RELEASED BT_UNDEFINED
Description	This Function gets the Button state.	

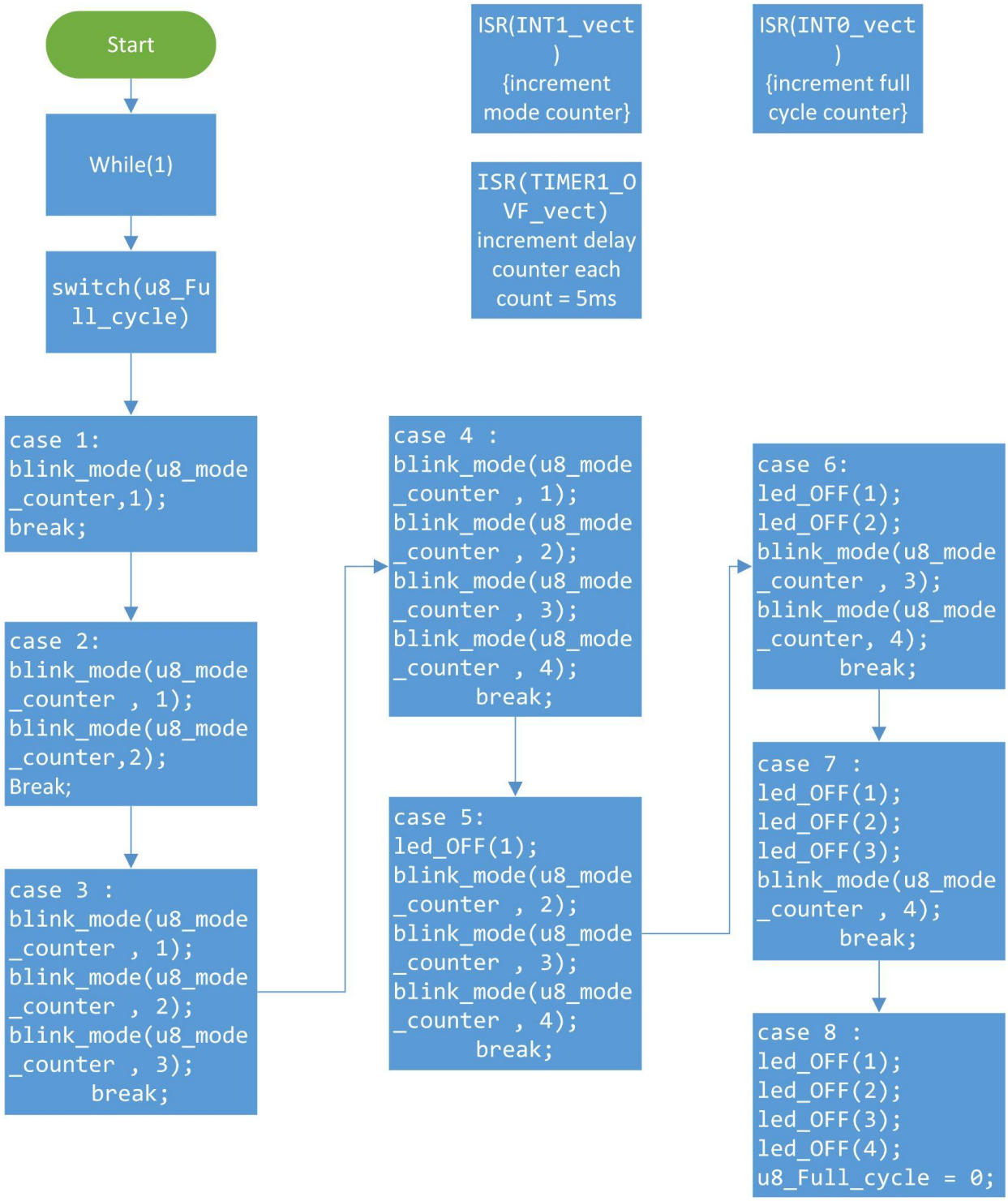
- button_Main_Task

Service name	button_Main_Task
Syntax	void button_Main_Taskt(void);
Parameters (in)	NONE
Return	NONE
Description	This Function update all button states Shall call periodic

App APIs:

App API:

Flowchart :



Services affecting the hardware unit:

- appStart

Service name	appStart
Syntax	void appStart(void);
Parameters (in)	NONE
Return	NONE
Description	This Function Start the application.