LED Sequence V2.0

By: Alaa Hisham

Project Description

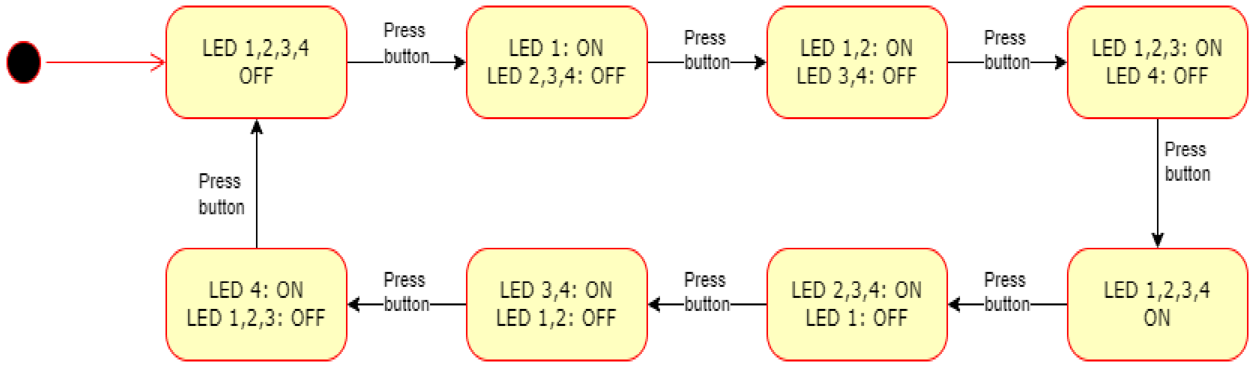
An App designed to display the following sequence:

1. Initially (OFF, OFF, OFF, OFF)
2. Press 1 (ON, OFF, OFF, OFF)
3. Press 2 (ON, ON, OFF, OFF)
4. Press 3 (ON, ON, ON, OFF)
5. Press 4 (ON, ON, ON, ON)
6. Press 5 (OFF, ON, ON, ON)
7. Press 6 (OFF, OFF, ON, ON)
8. Press 7 (OFF, OFF, OFF, ON)
9. Press 8 (OFF, OFF, OFF, OFF)
10. Press 9 (ON, OFF, OFF, OFF)

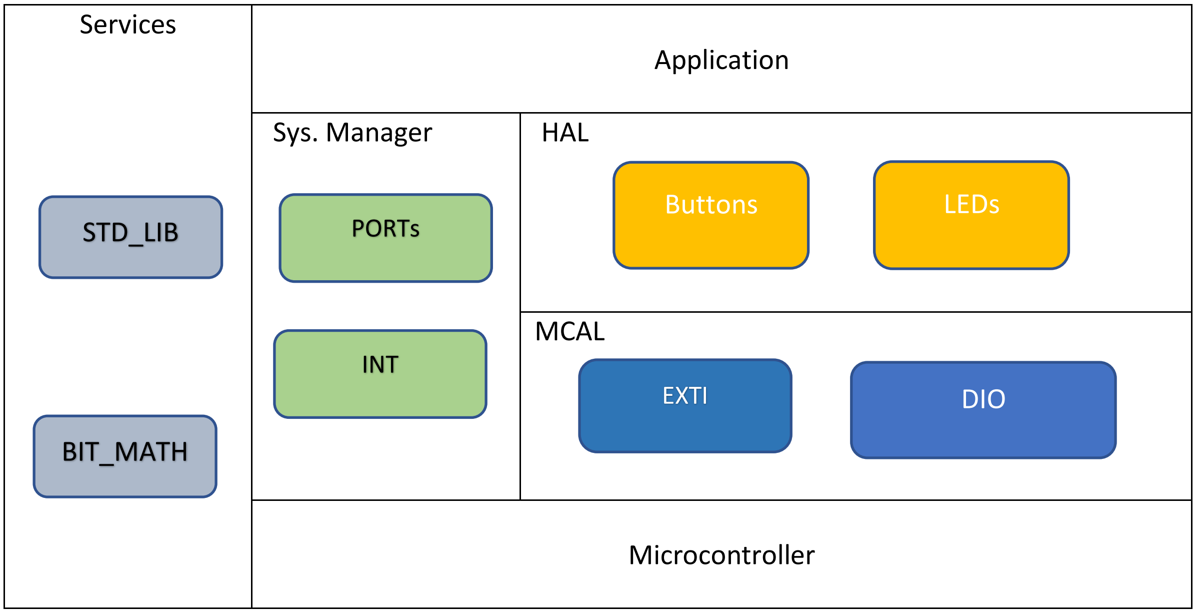
on four LEDs at each press of a push button using external interrupts.

The system is divided into layers and modules as follows.

Project State Machine



Layered Architecture



Modules & APIs

System

- **Ports APIs:**

```
/**
@def configure the direction and initial value of all IO pins
*/
void Sys_PortInit(void);
```

- **INT APIs:**

```
/**
 * enables displaying output on given led
*/
void Sys_GeneralIntEnable(void);

/**
 * Disables displaying output on given led
 */
void Sys_GeneralIntDisable(void);
```

MCAL

- **DIO APIs:**

```
/**
 * @def function to configure a single DIO pin as input/output
 * @param Copy_Port the port of the required pin
 * @param Copy_Pin the pin number in the given port
 * @param Copy_Value desired value (high or low) to set the pin to
 * @return error status
 */
EN_DIOErrorState_t DIO_SetPinVal(uint8_t Copy_Port,uint8_t Copy_Pin, uint8_t Copy_Value);

/**
 * @def function to configure a single DIO pin as input/output
 * @param Copy_Port the port of the required pin
 * @param Copy_Pin the pin number in the given port
 * @param Copy_Value desired value (high or low) to set the entire port to
 * @return error status
 */
EN_DIOErrorState_t DIO_SetPortVal(uint8_t Copy_Port, uint8_t Copy_Value);

/**
 * @def function to get the value of a single DIO pin whether high or low
 * @param Copy_Port the port of the required pin
 * @param Copy_Pin the pin number in the given port
 * @param Copy_pu8Val pointer to a variable to store pin value (0-255)
 * @return error status
 */
EN_DIOErrorState_t DIO_GetPinVal(uint8_t Copy_Port, uint8_t Copy_Pin, uint8_t*
Copy_pVal);
```

```
/**
 * @def function to toggle the value of the given pin
 * @param Copy_Port the port of the required pin
 * @param Copy_Pin the pin number in the given port
 * @return error status
 */
EN_DIOErrorState_t DIO_TogglePin(uint8_t Copy_Port,uint8_t Copy_Pin);
```

- **EXTI APIs:**

```
/**
 * Initializes given External interrupt with given mode
 */
EN_EXTIErrorState_t EXTI_Init(EN_EXTI_t Copy_Int, EN_EXTISenseMode_t Copy_Mode);

/**
 * enables specific interrupt of given EXTI number
 */
EN_EXTIErrorState_t EXTI_Enable(EN_EXTI_t Copy_Int);

/**
 * disables specific interrupt of given EXTI number
 */
EN_EXTIErrorState_t EXTI_Disable(EN_EXTI_t Copy_Int);

/**
 * Sets given function to be called when given EXTI is triggered
 */
EN_EXTIErrorState_t EXTI_SetCallback(EN_EXTI_t Copy_Int, void (*Copy_pCallbackFn)(void));
```

HAL

- **LEDs APIs**

```
/**
 * enables displaying output on given led
*/
EN_LEDErrorState_t LED_EnableLED(ST_LED* Copy_LED);

/**
 * Disables displaying output on given led
 */
EN_LEDErrorState_t LED_DisableLED(ST_LED* Copy_LED);

/**
 * Set the state of the given led to On/Off
 */
EN_LEDErrorState_t LED_setState(ST_LED* Copy_LED, EN_LEDState Copy_LEDState);

/**
 * Toggles the given given led
 */
EN_LEDErrorState_t LED_Toggle(ST_LED* Copy_Led);
```

- **Buttons APIs**

```
/**
 * enables reading from given switch
 */
EN_SWError_t SW_EnableSwitch(ST_Switch* Copy_Switch);

/**
 * Disables reading from given switch
 */
EN_SWError_t SW_DisableSwitch(ST_Switch* Copy_Switch);

/**
 * Read the value on the switch pin and stores it in given reference
 */
EN_SWError_t SW_ReadSwitch(ST_Switch* Copy_Switch, EN_SWValue_t* Copy_SwitchValue);

/**
 * Initialize switch as External Interrupt source
 */
EN_SWError_t SW_EXTIMode(ST_Switch* Copy_Switch, EN_SW_Interrupt_t Copy_IntEvent,
void (*Copy_pvCallbackFn)(void));
```

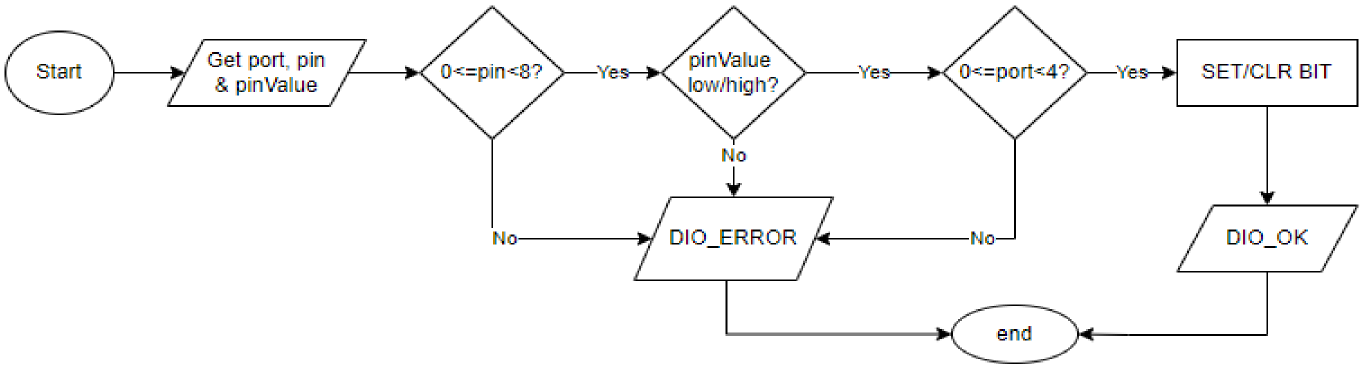
APPLICATION

```
/**
 * initialize the configured io ports
 * and enable global interrupt
 */
void App_Init(void);

/**
 * The app main logic
 */
void App(void);

/************** Button Callback Function *************/
/**
 * increments the count of button presses to enter the
 * right state
 */
void Button0_Callback(void);
```

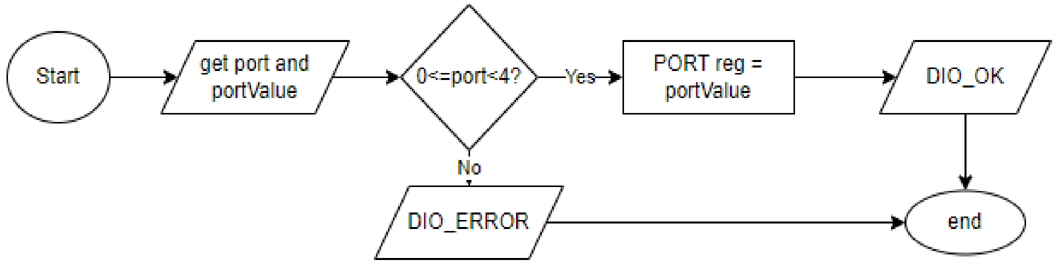
API Design Diagrams

- **DIO APIs**

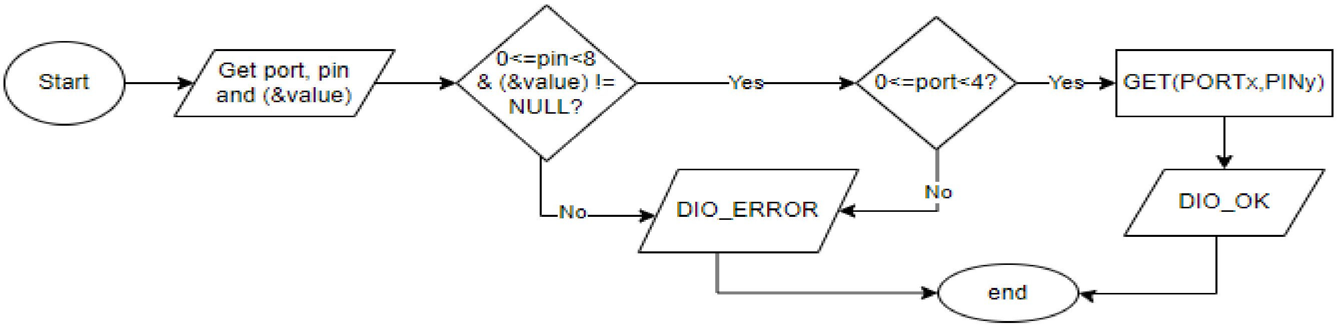
DIO_SetPinVal



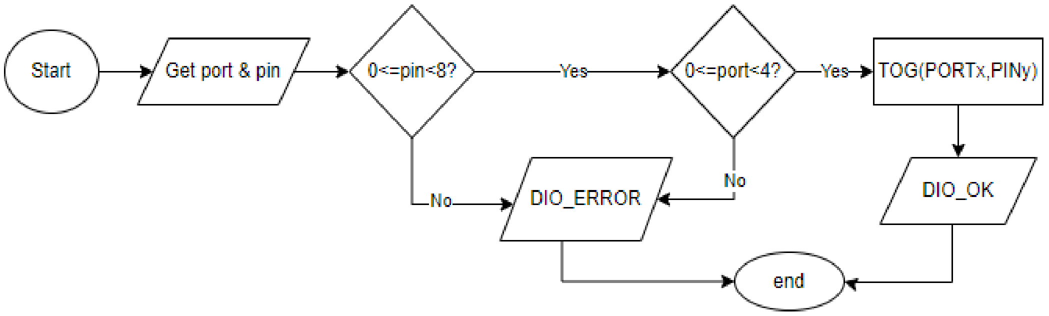
DIO_SetPortVal



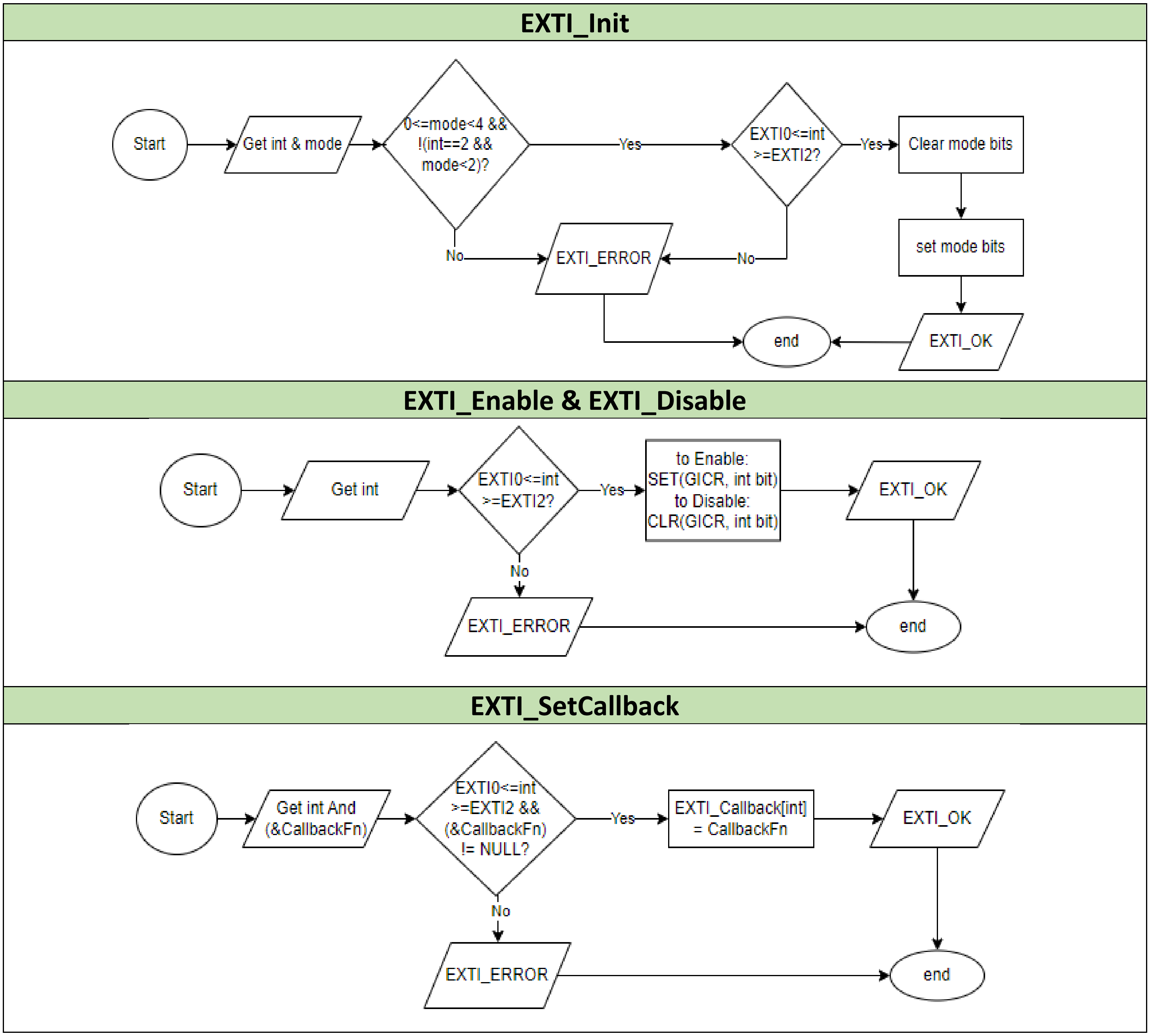
DIO_GetPinVal



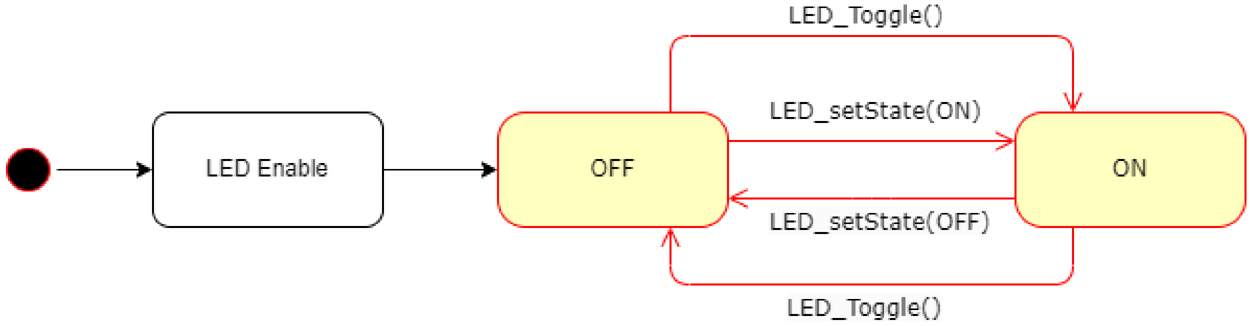
DIO_TogglePin



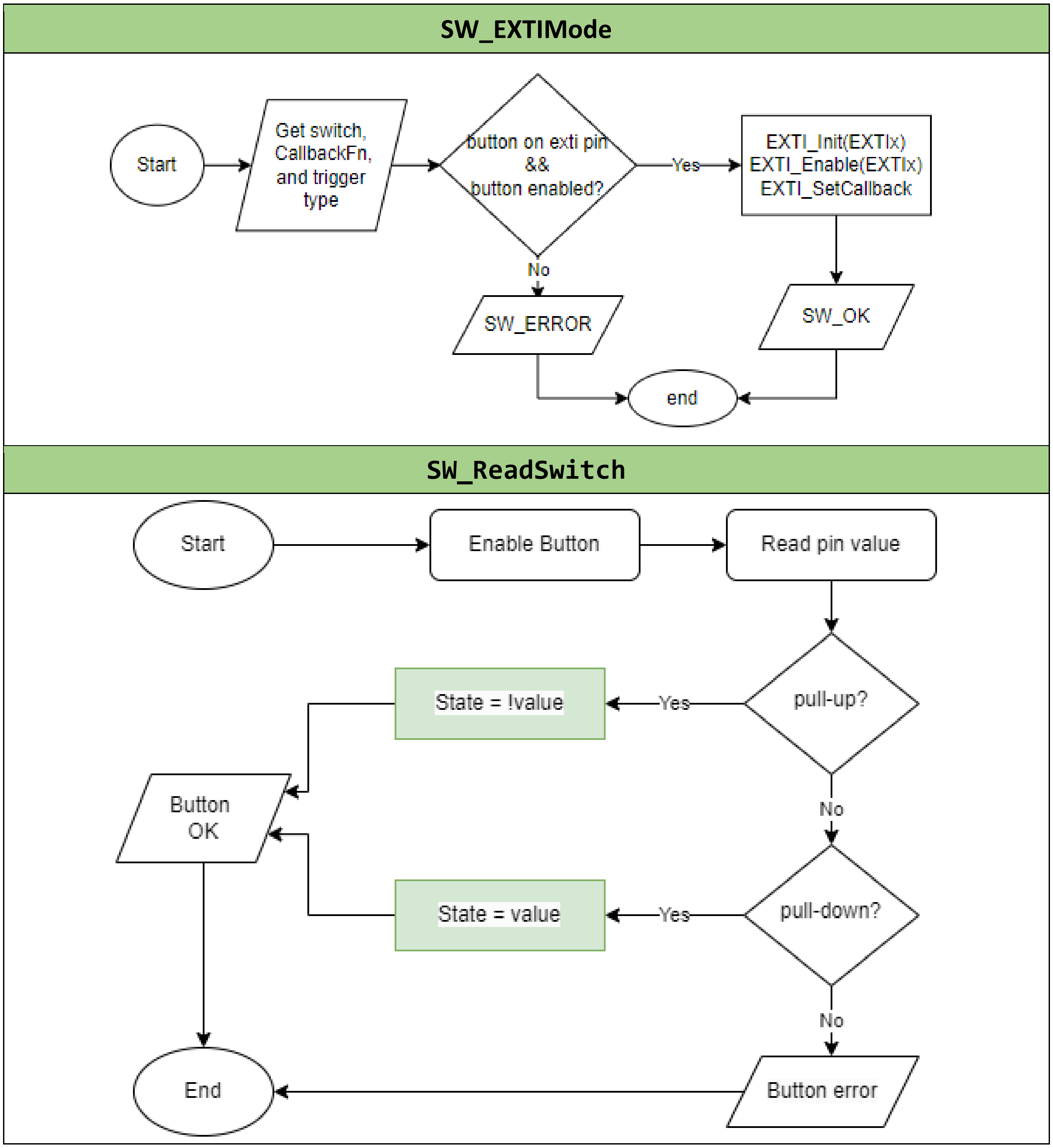
- **EXTI APIs**


EXTI_Init
Start
Get int & mode
0<=mode<4 && !(int==2 && mode<2)?
Yes
No
EXTI0<=int >=EXTI2?
Yes
No
Clear mode bits
set mode bits
EXTI_ERROR
EXTI_OK
end
EXTI_Enable & EXTI_Disable
Start
Get int
EXTI0<=int >=EXTI2?
Yes
No
to Enable: SET(GICR, int bit) to Disable: CLR(GICR, int bit)
EXTI_OK
EXTI_ERROR
end
EXTI_SetCallback
Start
Get int And (&CallbackFn)
EXTI0<=int >=EXTI2 && (&CallbackFn) != NULL?
Yes
No
EXTI_Callback[int] = CallbackFn
EXTI_OK
EXTI_ERROR
end

- **LED APIs State Machine**



- **Button APIs**



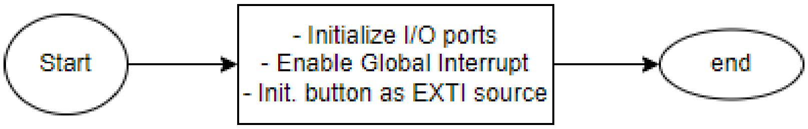
SW_EXTIMode

Start
Get switch, CallbackFn, and trigger type
button on exti pin && button enabled?
Yes
EXTI_Init(EXTIx) EXTI_Enable(EXTIx) EXTI_SetCallback
No
SW_ERROR
SW_OK
end

SW_ReadSwitch

- **APP APIs**

App_Init:



App() : Main Logic, same as project state machine.