LED Sequence V3

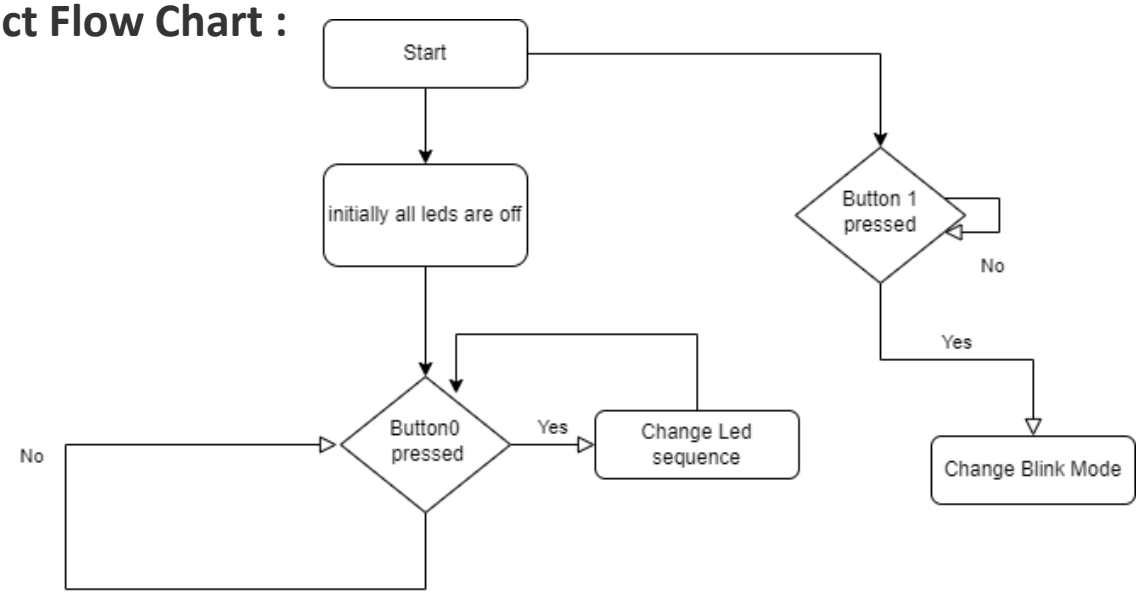
By Sharpel Malak

Project Description :

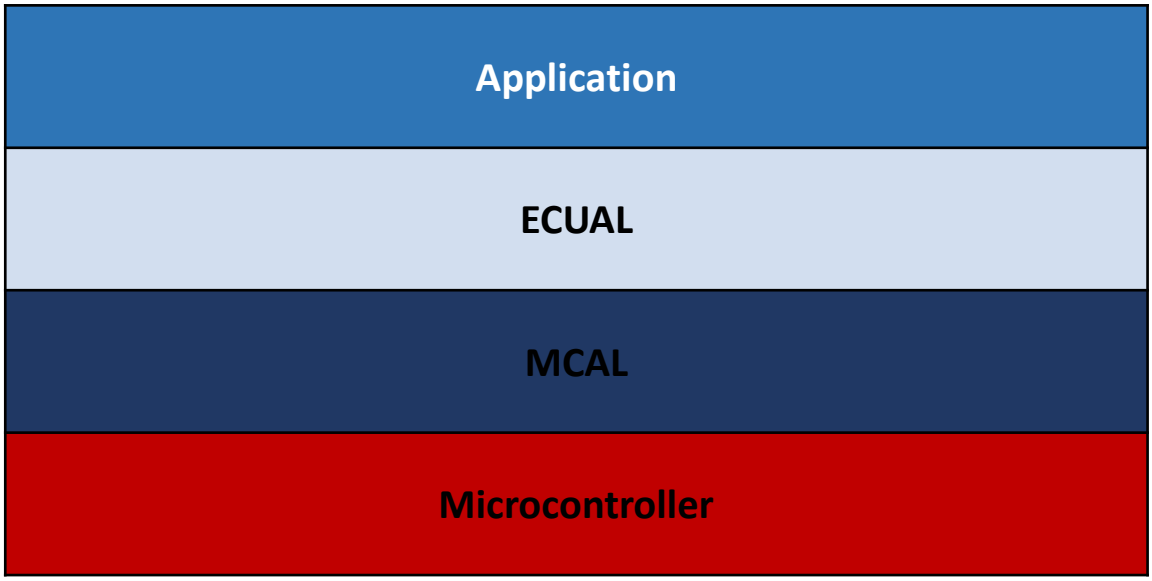
The project contains Four LEDs (LED0, LED1, LED2, LED3)
and two buttons (button0,Button1)
- Initially, all LEDs are OFF
- Once **BUTTON0** is pressed, **LED0** will blink with **BLINK_1** mode
- Each press further will make another LED blinks **BLINK_1** mode
- At the **fifth press**, **LED0** will changed to be **OFF**
- Each **press further** will make only one LED is **OFF**
- This will be repeated forever
- When BUTTON1 has pressed the blinking on and off durations will be changed

No press → **BLINK_1** mode (**ON**: 100ms, **OFF**: 900ms)
First press → **BLINK_2** mode (**ON**: 200ms, **OFF**: 800ms)
Second press → **BLINK_3** mode (**ON**: 300ms, **OFF**: 700ms)
Third press → **BLINK_4** mode (**ON**: 500ms, **OFF**: 500ms)
Fourth press → **BLINK_5** mode (**ON**: 800ms, **OFF**: 200ms)
Fifth press → **BLINK_1** mode

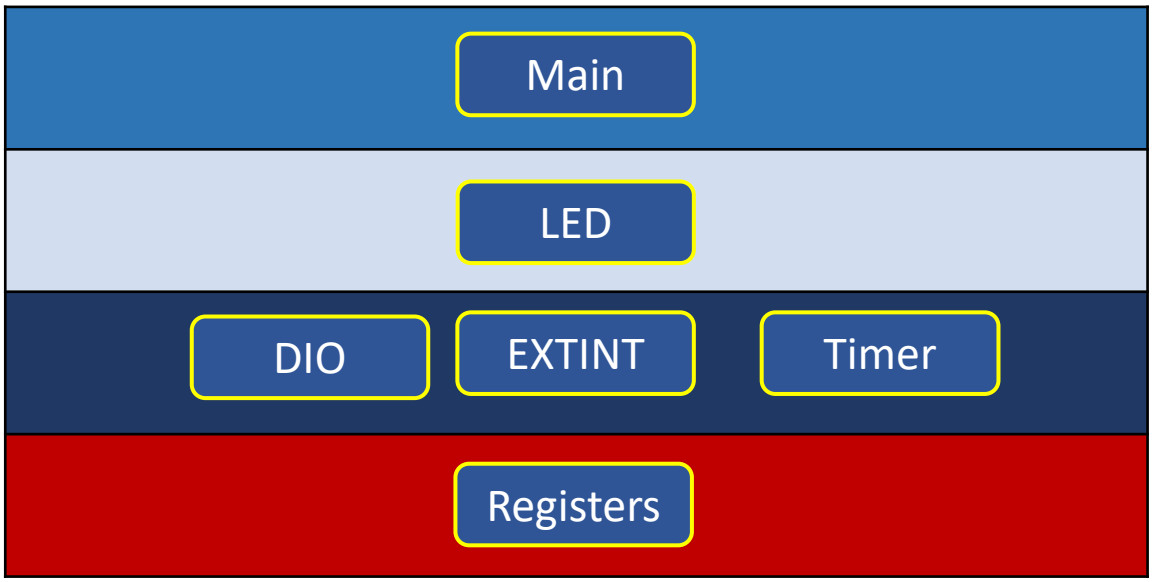
Project Flow Chart :



1 : Layered Architecture :



2 : System modules



3 : Project Modules APIs

3.1 - DIO

```
// DIO TYPEDEFS
typedef enum EN_DIO_ERROR{
   DIO_OK=0,
   DIO_NOT_OK
}EN_DIO_ERROR;
typedef enum EN_DIO_PINS{
   DIO_PIN0=0,
   DIO_PIN1,
   DIO_PIN2,
   DIO_PIN3,
   DIO_PIN4,
   DIO_PIN5,
   DIO_PIN6,
   DIO_PIN7,
}EN_DIO_PINS;
typedef enum EN_DIO_PORTS{
   DIO_PORTA=0,
   DIO_PORTB,
   DIO_PORTC,
   DIO_PORTD
}EN_DIO_PORTS;
typedef enum EN_DIO_DIRECTION{
   INPUT=0,
   OUTPUT
}EN_DIO_DIRECTION;
typedef enum EN_DIO_LEVEL{
   LOW=0,
   HIGH
}EN_DIO_LEVEL;

// DIO FUNCTIONS PROTOTYPES

// Description : This function initialize PIN and set it's direction
//     ARGS       : take PIN Number and PORT Number and Direction (INPUT,OUTPUT)
//     return     : return DIO_OK if the PIN initializes correctly, DIO_NOT_OK otherwise
EN_DIO_ERROR DIO_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_DIRECTION direction);

// Description : This function write on PIN and set it's level
// ARGS       : take PIN Number and PORT Number and level (LOW,HIGH)
// return     : return DIO_OK if the PIN level sets correctly, DIO_NOT_OK otherwise
EN_DIO_ERROR DIO_write(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_LEVEL level);

// Description : This function toggles PIN level
// ARGS       : take PIN Number and PORT Number
// return     : return DIO_OK if the PIN toggles correctly, DIO_NOT_OK otherwise
EN_DIO_ERROR DIO_toggle(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);

// Description : This function reads PIN level and store it in the variable
// ARGS       : take PIN Number and PORT Number and pointer to the variable
// return     : return DIO_OK if the PIN value stored correctly , DIO_NOT_OK otherwise
EN_DIO_ERROR DIO_read(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber, uint8_t * value);
```

3.2 – External Interrupt

```
// EXT_INT TYPEDEFS
typedef enum EN_EXTINT_ERROR {
EXTINT_OK=0,
EXTINT_NOT_OK
}EN_EXTINT_ERROR;
typedef enum EN_Sence_Control {
LOW_LEVEL=0,
FALLING_EDGE,
RISING_EDGE,
ANY_LOGICAL_CHANGE
}EN_Sence_Control;

typedef enum EN_EXINT_NUMBER{
EXTINT0=0,
EXTINT1,
EXTINT2,
}EN_EXINT_NUMBER;

typedef enum EN_GLOBAL_INT{
DISABLE=0,
ENABLE
}EN_GLOBAL_INT;
// EXT_INT prototypes
EN_EXTINT_ERROR SET_GLOBAL_INTERRUPT(EN_GLOBAL_INT state);
EN_EXTINT_ERROR EXTINT_init(EN_EXINT_NUMBER INTx ,EN_Sence_Control INTxSense);
EN_EXTINT_ERROR EXTINT_CallBack(EN_EXINT_NUMBER INTx,void(*ptrfunc)(void));
```

3.3 - Timer

```
void timer0_init(void);
void timer0_start(void);
void timer0_stop(void);
void timer0_set_delay(uint32_t delay_ms);
void TIMER_INT_CallBack(void(*ptrfunc)(void));
```

3.4 - LED

```
typedef enum EN_LED_Error_t
{
   LED_OK = 0,
   LED_NOT_OK
}EN_LED_Error_t;

EN_LED_Error_t LED_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_on(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_off(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
EN_LED_Error_t LED_toggle(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);
```

4 : APIs State Machine