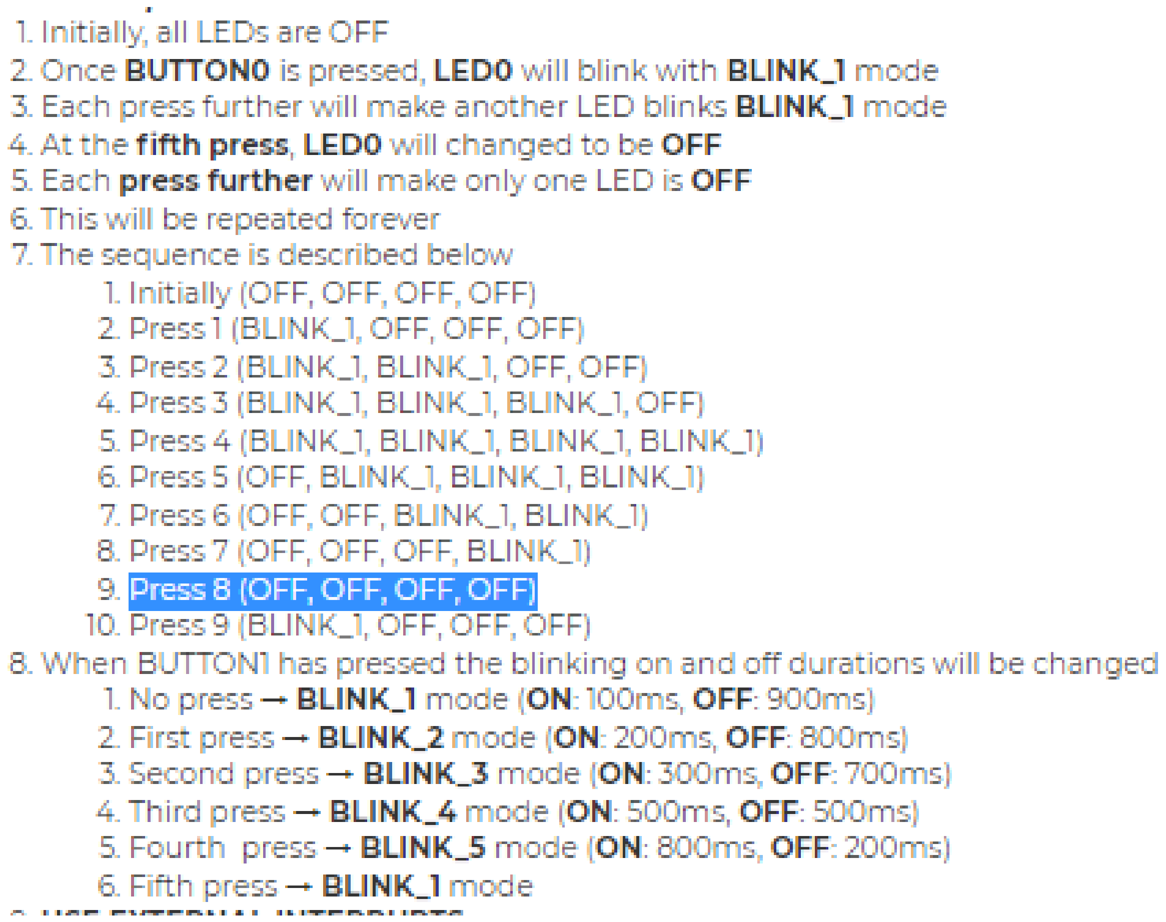
TASK: LED SEQUENCE V3.0

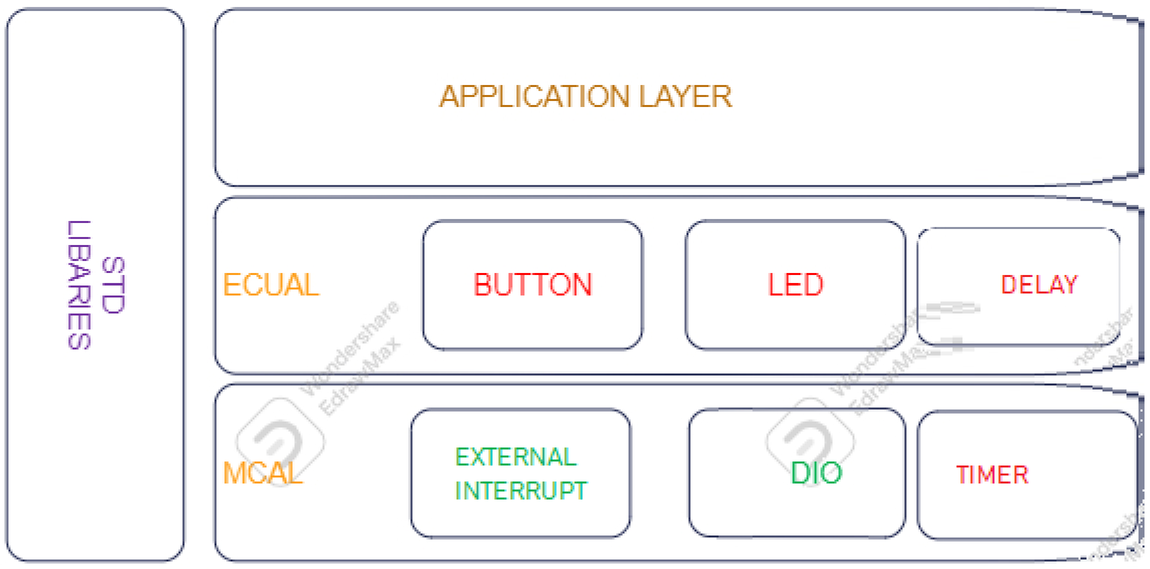
AUTHOR: MOHAMMED ABDEL-WAHAB

DESCRIPTION:

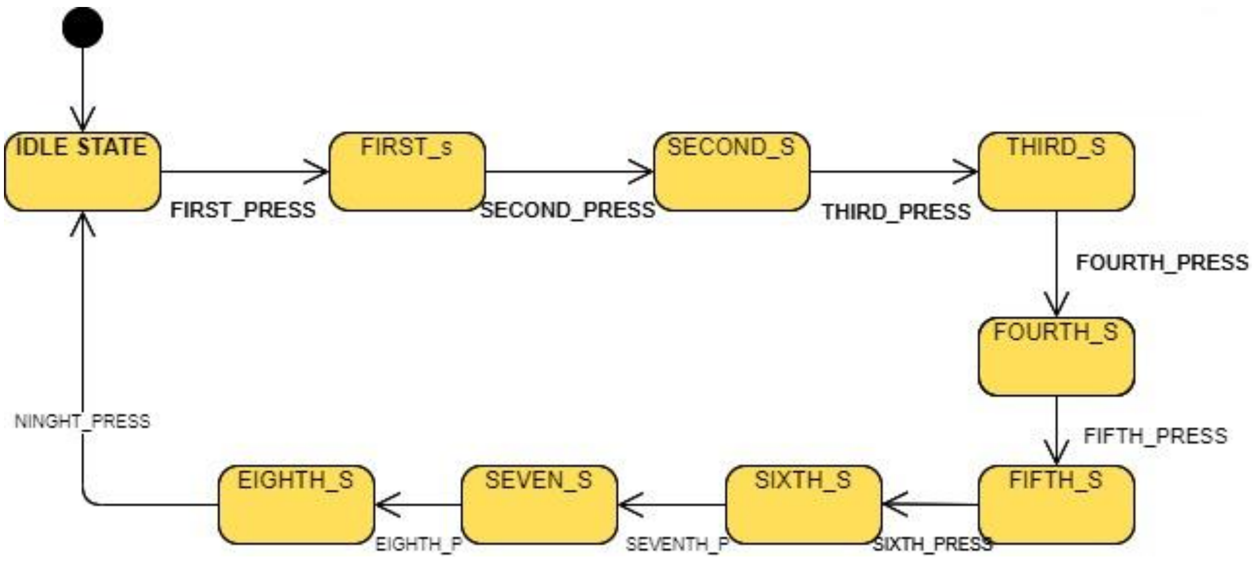


1. Initially, all LEDs are OFF
2. Once **BUTTON0** is pressed, **LED0** will blink with **BLINK_1** mode
3. Each press further will make another LED blinks **BLINK_1** mode
4. At the **fifth press**, **LED0** will changed to be **OFF**
5. Each **press further** will make only one LED is **OFF**
6. This 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

Layere Architecture:

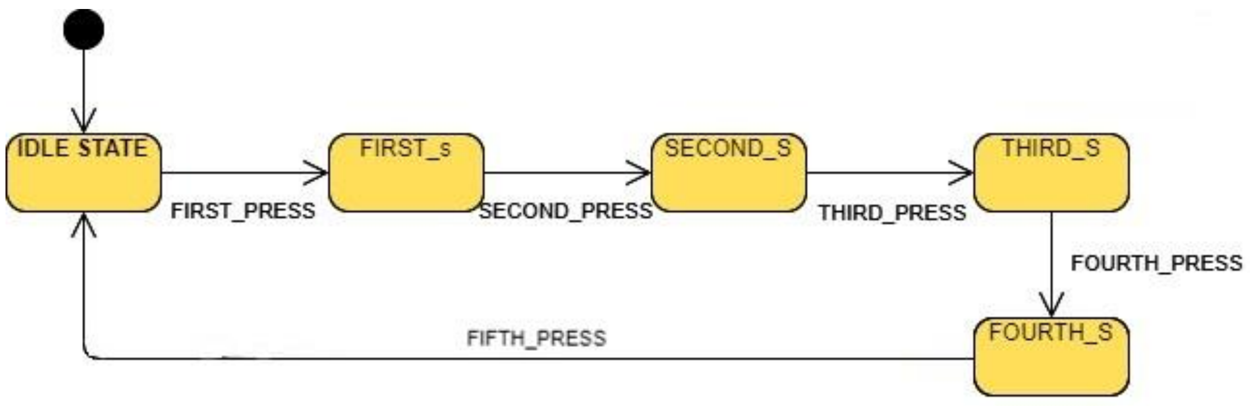


State machine diagram for the LED SEQUANCE OF THE APPLICATION:



- IDLE STATE: ALL LEDS ARE OFF
- FIRST STATE: LED 0 IS ONLY BLINKING
- SECOND STATE: LED 0 & LED 1 ARE BLINKING
- THIRD STATE : LED 0 & LED 1 & LED 2 ARE BLINKING
- FOURTH STATE: ALL LEDS ARE BLINKING
- FIFTH STATE: LED 0 IS ONLY OFF
- SIXTH STATE: LED 0 & LED 1 ARE OFF
- SEVENTH STATE: LED 0 & LED 1 & LED 2 ARE OFF
- EIGHTH STATE: ALL LEDS ARE OFF

<u>State machine diagram for the LED BLINKING MODES OF THE APPLICATION:</u>



- IDLE STATE: No press → BLINK_1 mode (ON: 100ms, OFF: 900ms)
- FIRST STATE: First press → BLINK_2 mode (ON: 200ms, OFF: 800ms)
- SECOND STATE: Second press → BLINK_3 mode (ON: 300ms, OFF: 700ms)
- THIRD STATE : Third press → BLINK_4 mode (ON: 500ms, OFF: 500ms)
- FOURTH STATE: Fourth press → BLINK_5 mode (ON: 800ms, OFF: 200ms)
- FIFTH STATE: Fifth press → BLINK_1 mode

ALL project APIs:

DIO DRIVER APIs:

```
/**
* @brief Initialize the direction of specific pin @ref direction_t
* @param _pin_config A Reference of the pin configuration @pin_config_t
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_pin_direction_intialize(const pin_config_t *pin_config_ptr,direction_t
a_direction);
/**
* @brief Write the logic of specific pin @ref logic_t
* @param _pin_config A Reference of the pin configuration @pin_config_t
* @param logic
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_pin_write_logic(const pin_config_t *pin_config_ptr,const logic_t
a_logic);
/**
* @brief Read the logic of specific pin @ref logic_t
* @param _pin_config A Reference of the pin configuration @pin_config_t
* @param logic
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_pin_read_logic(const pin_config_t *pin_config_ptr, logic_t
*logic_ptr);

/**
* @brief Toggle the logic of specific pin @ref logic_t
* @param _pin_config A Reference of the pin configuration @pin_config_t
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_pin_toggle_logic(const pin_config_t *pin_config_ptr);
/**
* @brief Initialize the direction of specific pin and Initialize its logic
* @param _pin_config A Reference of the pin configuration @pin_config_t
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
/*
Std_ReturnType DIO_pin_intialize(const pin_config_t *pin_config_ptr);
```

```c
*/
/**
*
* @param port_index
* @param direction
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_port_direction_intialize(const port_index_t a_port_index, uint8_t
a_direction);
/**
* @param port_index
* @param logic
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_port_write_logic(const port_index_t a_port_index , uint8_t a_logic);
/**
* @param port_index
* @param logic
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_port_read_logic(const port_index_t a_port_index , uint8_t *const
a_logic_ptr);
/**
* @param port_index
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType DIO_port_toggle_logic(const port_index_t a_port_index);
```

LED DRIVER APIs:

```c
/**
 * @breif  Initialize The led by configuring the pin as output and write low
 * @param Led  The reference of the led module configuration
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType LED_initialize(const led_t *led_ptr);

/**
 * @breif Turn the led on
 * @param led  The reference of the led module configuration
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */

Std_ReturnType LED_turn_on(const led_t *led_ptr);
```

```
/**
 * @breif Turn the led off
 * @param led  The reference of the led module configuration
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */

Std_ReturnType LED_turn_off (const led_t *led_ptr);

/**
 * @breif  Toggle the led
 * @param led  The reference of the led module configuration
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType LED_turn_toggle (const led_t *led_ptr);
```

BUTTON DRIVER APIs:

```
/**
 * @breif Initialize The assigned pin to be input
 * @param btn he reference of the button module configuration
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType BTN_init(const button_t *btn_ptr);

/**
 * @breif Read the push button if is it pressed or released
 * @param btn    The reference of the button module configuration
 * @param btn_state  The reference of the variable that store the button status @ref
button_status_t
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType BTN_read_state(const button_t *btn_ptr, button_status_t *btn_states_ptr);
```

EXTERNAL INTERRUPT APIs:

```
/*
* Description : Call the Call Back function in the application after the edge is detected
* @param A pointer to function & the external interrupt id
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType EXT_INTx_setCallBack(volatile void(*a_fptr)(void), const Interrupt_ID_t
a_interrupt_number );
/*
```

```
* Description : initialize the the dio pin to be an external interrupt
* @param A Reference of the external interrupt configuration
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType EXT_INTx_Init(const Interrupt_Config_t *Interrupt_Config_Ptr );
/*
* Description : set the edge in which the external interrupt will be triggered
* @param edge type & the external interrupt id
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType EXT_INTx_setEdgeType(Interrupt_Edge_type_t a_edgeType ,  Interrupt_ID_t
a_interrupt_Id);
/**
* @brief DeInitialize the interrupt module
* @param the external interrupt id
* @return Status of the a_interrupt_Id
* (E_OK) : The function done successfully
* (E_NOT_OK) : The function has issue to perform this action
*/
Std_ReturnType EXT_INTx_DeInit(const Interrupt_ID_t a_interrupt_Id);
```

TIMER DRIVER APIs:

```
/*
 * Description:  Function to Initialize Timer Driver
 *                 - Working in Interrupt Mode
 *                 - Choose Timer initial value
 *                 - Choose Timer_ID (Timer0, Timer1, Timer2)
 *                 - Choose Timer_Mode (OverFlow, Compare,PWM)
 *                 - if using CTC mode choose Timer compare match value  And choose
output_compare_mode
 *
 *@param A Reference of the Timer configuration
 * @return status of the function
 * E_OK :the function done successfully
 * E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType TIMERx_init(const Timer_Config_t *stPtr_a_Config);
/*
* Description : START COUNTING BY CONFIGURE THE TIMER CLOCK
* @param A Reference of the Timer configuration
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType TIMERx_start(const Timer_Config_t *stPtr_a_Config);
/*
* Description : Call the Call Back function in the application after timer did its job
* @param A pointer to function & the  timer type
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
```

```c
*/
Std_ReturnType TIMERx_setCallBack(volatile void(*a_fptr)(void), const TimerType_t
en_a_timer_type );

/*
* Description :set a certain value on the timer counting register
* @param the  timer type and the initial value to be set
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType TIMERx_setValue(const TimerType_t en_a_timer_type ,const uint16_t
u16_a_timer_value);

/*
* Description :this function sets the offset of the compare unit
* @param timer type and the top value to be compared with the TCNCx
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType TIMERx_CTC_SetCompare(const TimerType_t en_a_timer_type ,const uint16_t
u16_a_compareValue);

/*
* Description :Function to make the timer to start again from beginning(reset)
* @param the  timer type and the initial value to be set
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType TIMERx_reset(const TimerType_t en_a_timer_type);

/*
* Description :Function to Halt the timer (stop)
* @param the  timer type and the initial value to be set
* @return status of the function
* E_OK :the function done successfully
* E_NOT_OK :the function has issues performing the function
*/
Std_ReturnType TIMERx_stop(const TimerType_t en_a_timer_type);
```

DELAY DRIVER API:

```c
/**
 * @brief  generate a delay (busy wait using timer) in ms
 * @param u16_a_delay_ms the delay value in ms
 * @return status of the function
 *          E_OK :the function done successfully
 *          E_NOT_OK :the function has issues performing the function
 */
Std_ReturnType DELAY_ms(const uint16_t u16_a_delay_ms);
```