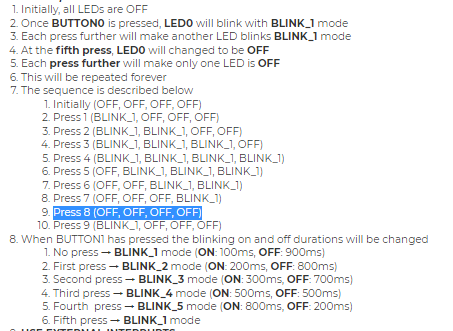
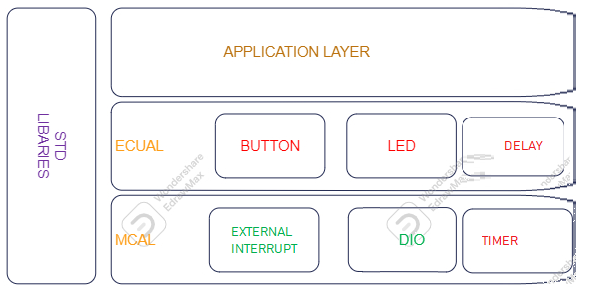
TASK: LED SEQUENCE V3.0

AUTHOR: MOHAMMED ABDEL-WAHAB

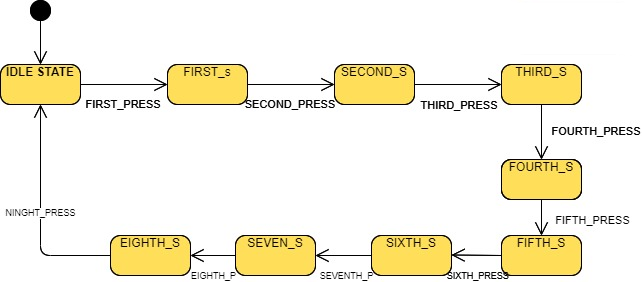
**DESCRIPTION**:



**Layere Architecture:**

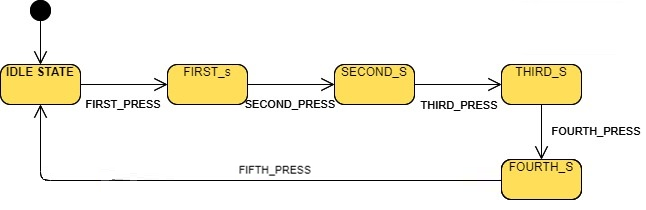


**State machine diagram for the LED SEQUANCE OF THE APPLICATION:**



* 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

**State machine diagram for the LED BLINKING MODES OF THE APPLICATION:**

****

* 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);

\*/

/\*\*

\*

\* @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:

/\*\*

\* @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

\*/

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:

/\*\*

\* @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);