Author: Mustafa Mohammed Abdou Soliman

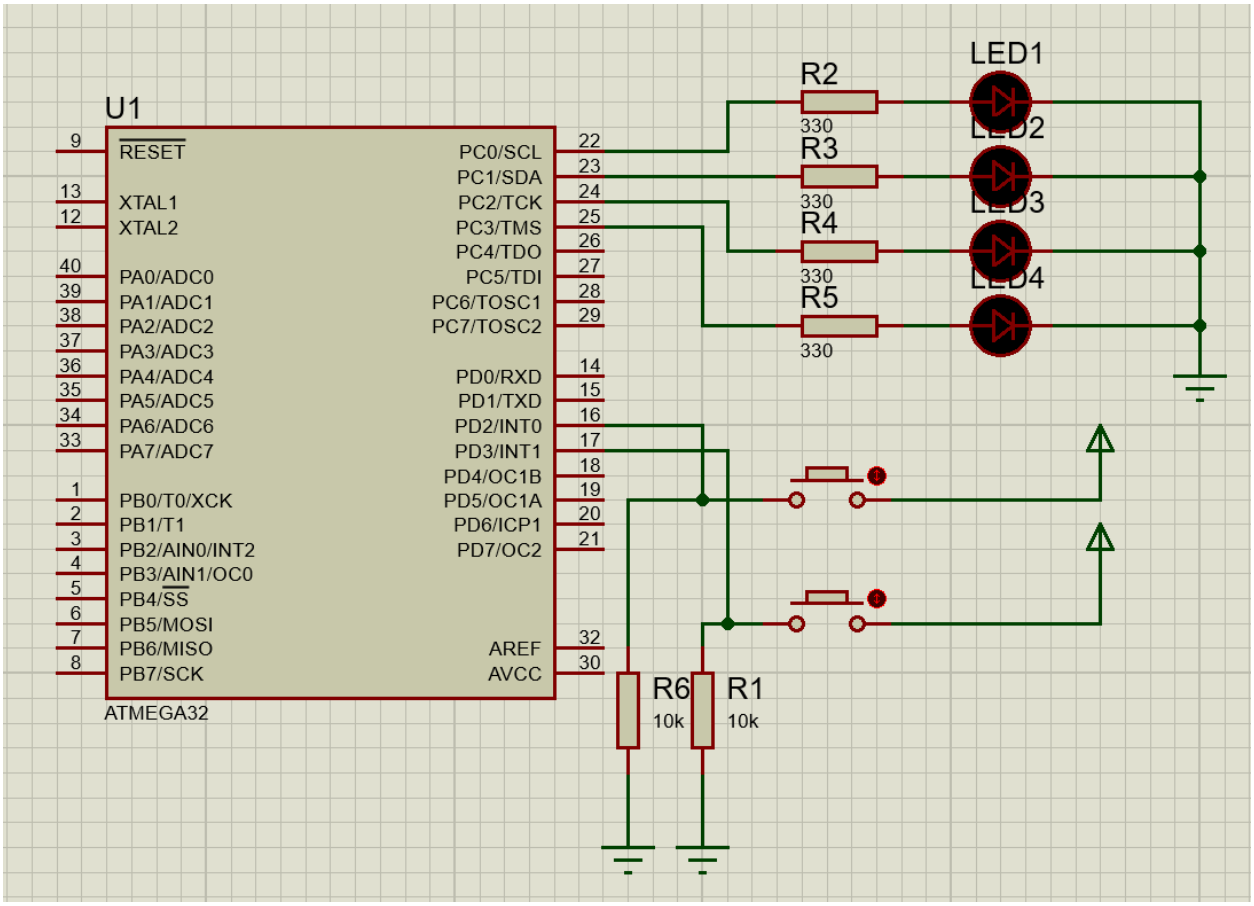
Project: LEDs sequence V3.0

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

Schematic capture:



Layered architecture:

1- Libraries: standard_types , common macros
2- MCAL: DIO, INT , TMR0
3- HAL: LED
4- APP: main.c

1- DIO module contains (DIO_interface.h , DIO_private.h , DIO_config.h , DIO_program.c)

DIO_interface.h

```
2-  /** TYPE DEFINITION FOR PIN DIRECTION TO RETURN ITS STATE **/
3-  typedef enum { VALID_DIRECTION , NOT_VALID_DIRECTION } PinDirection_t ;
4-
5-  /** TYPE DEFINITION FOR PIN VALUE TO RETURN ITS STATE  **/
6-  typedef enum { VALID_VALUE , NOT_VALID_VALUE } PinValue_t ;
7-
8-  /** TYPE DEFINITION FOR PIN READ STATUS  **/
9-  typedef enum { VALID_READ , NOT_VALID_READ } PinRead_t ;
10-
11- /*****************************************************************/
12- /* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PIN   */
13- /* INPUT   : PORT , PINID , DIRECTION                            */
14- /* RETURNS : PinDirection_t                                      */
15- /*****************************************************************/
16- PinDirection_t DIO_SETPINDIR(uint8_t portID , uint8_t pinID , uint8_t dir);
17-
18-
19- /*****************************************************************/
20- /* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PORT  */
21- /* INPUT   : PORTID , DIRECTION                                  */
22- /* RETURNS : VOID                                                */
23- /*****************************************************************/
24- //void DIO_SETPORTDIR(uint8_t portID , uint8_t dir);
25-
26-
27- /*****************************************************************/
28- /* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PIN       */
29- /* INPUT   : PORT , PINID , DIRECTION                            */
30- /* RETURNS : PinValue_t                                          */
31- /*****************************************************************/
32- PinValue_t DIO_SETPINVAL(uint8_t portID , uint8_t pinID , uint8_t val);
33-
34- /*****************************************************************/
35- /* DESCRIBTION  : FUNCTION TO GET THE VALUE OF SPECIFIC PIN       */
36- /* INPUT   : PORTID , PINID , POINTER TO SET THE VALUE IN IT      */
37- /* RETURNS : PinRead_t                                           */
38- /*****************************************************************/
    PinRead_t DIO_READPIN(uint8_t portID , uint8_t pinID , uint8_t* val);
```

MCAL: INT (INT_interface.h , INT_config.h , INT_private.h , INT_program.c)

INT_interface.h

```
/*******************************************************/
/** FUNCTION TO SET THE GLOBAL INTERRUPT ENABLE FLAG    */
/** ARGUMENTS  : VOID                                   */
/** RETURNS    : VOID                                   */
/*******************************************************/
void SET_GLOBALINTERRUPT(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT0                         */
/** ARGUMENTS  : VOID                                   */
/** RETURNS    : VOID                                   */
/*******************************************************/
void INT0_INIT(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT1                         */
/** ARGUMENTS  : VOID                                   */
/** RETURNS    : VOID                                   */
/*******************************************************/
void INT1_INIT(void);
```

MCAL TMR0 Module (TMR0_config.h , TMR0_private.h , TMR0_interface.h , TMR0_program.c)

TMR0_interface.h

```
typedef enum { VALID_INIT  , NOT_VALID_INIT} TMR0_init ;
/*************************************************************/
/** FUNCTION TO INITIALIZE TMR0 WITH SOME CONFIGURATIONS     */
/** ARGUMENTS  : VOID                                        */
/** RETURNS    : TMR0_init                                   */
/*************************************************************/
TMR0_init TMR0_INIT(void);


typedef enum {VALID_START , NOT_VALID_START } TMR0_start;
/*****************************************************************************/
/** FUNCTION TO LET TIMER 0 START WORK BY ASSIGN PRESCALLER OR CLOCK SOURCE   */
/** ARGUMENTS  : VOID                                                        */
/** RETURNS    : TMR0_start                                                  */
/*****************************************************************************/
TMR0_start TMR0_START(void);



typedef enum {VALID_STOP , NOT_VALID_STOP } TMR0_stop;
/*****************************************************************************/
/** FUNCTION TO STOP TIMER 0                                                 */
/** ARGUMENTS  : VOID                                                        */
/** RETURNS    : TMR0_stop                                                   */
/*****************************************************************************/
TMR0_stop TMR0_STOP(void);


typedef enum {VALID_DELAY , NOT_VALID_DELAY } TMR0_delay ;
/*****************************************************************************/
/** FUNCTION TO SET DELAY USING TIMER 0                                      */
/** ARGUMENTS  : TAKES DELAY IN ms                                           */
/** RETURNS    : TMR0_delay                                                  */
/*****************************************************************************/
TMR0_delay TMR0_DELAY_MS(uint32_t DELAY_MS);
```

HAL: LED (LED_interface.h , LED_config.h , LED_program.c , LED_private.h)

LED_interface.h

```
/************************************************/
/** FUNCTION TO INITIALIZE A PIN                **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : VOID                              **/
/************************************************/
void LED_INIT(uint8_t led_port , uint8_t ledpin);


/************************************************/
/** FUNCTION TO SET A LED AS ON                 **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : VOID                              **/
/************************************************/
void LED_ON(uint8_t led_port , uint8_t ledpin);

/************************************************/
/** FUNCTION TO SET A LED AS OFF                **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : VOID                              **/
/************************************************/
void LED_OFF(uint8_t led_port , uint8_t ledpin);


/************************************************/
/** FUNCTION TO TOGLE A LED                     **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : VOID                              **/
/************************************************/
void LED_TOGGLE(uint8_t led_port , uint8_t ledpin);
```