Author: Mustafa Mohammed Abdou Soliman

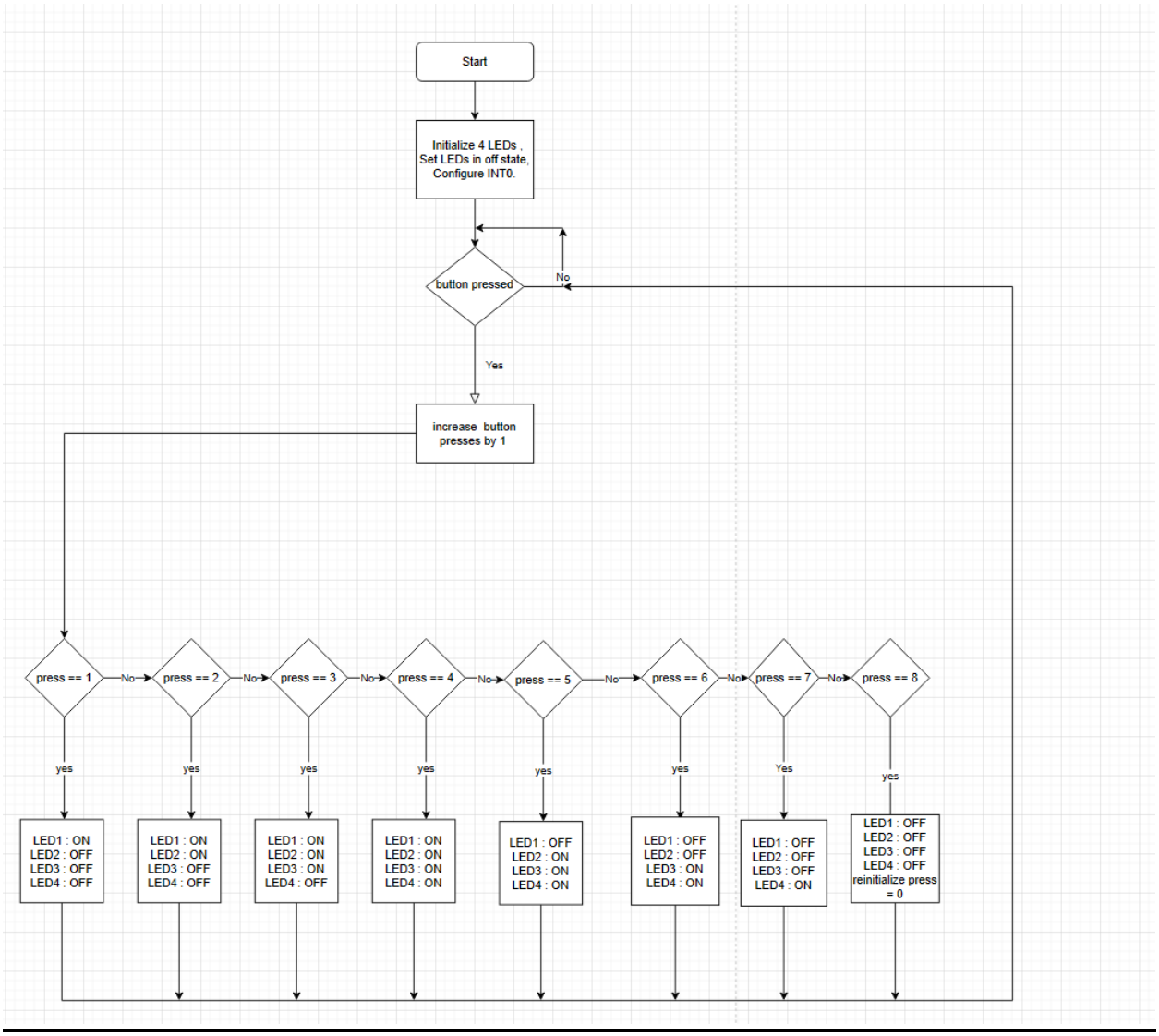
Project: LEDs sequence V1.0

Description:

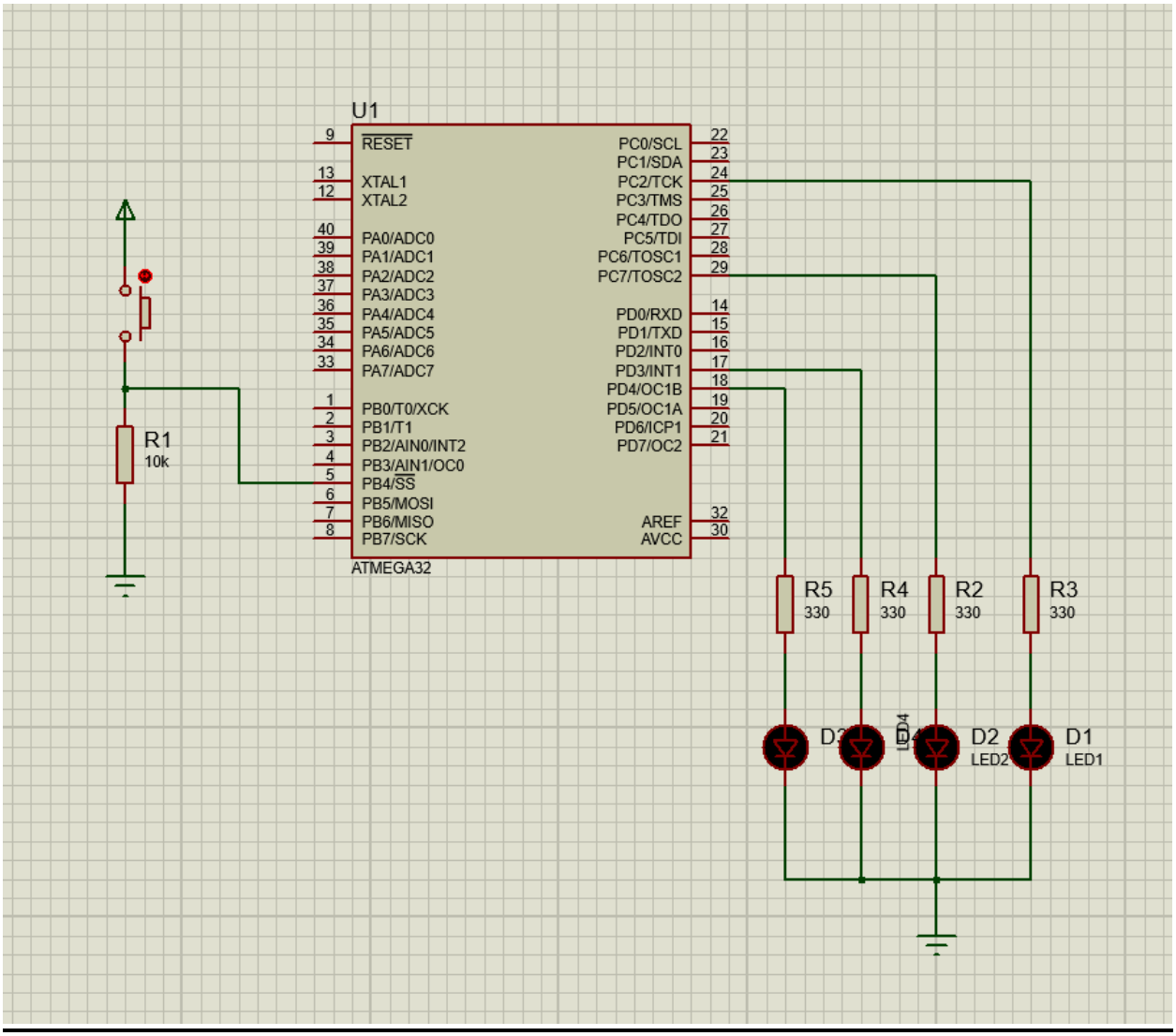
1. ***Hardware Requirements***
    1. *Four LEDs (LED0, LED1, LED2, LED3)*
    2. One button (BUTTON0)

2. ***Software Requirements***
    1. Initially, all LEDs are OFF
    2. Once BUTTON0 is pressed, LED0 will be ON
    3. Each press further will make another LED is ON
    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 (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)

Project flowchart:



Schematic capture:



Layered architecture:

1- Libraries: standard_types , common macros

2- MCAL: DIO

3- HAL: LED , button

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);
```

2- HAL: button (button_interface.h , button_config.h , button_private.h , button_program.c)

Button_interface.h

```c
/** TYPE DEFINITION FOR BUTTON RETURN ERROR */
typedef enum { Button_Notpressed  , Button_pressed} button_t;


/*****************************************************************/
/* FUNCTION TO INITIALIZE THE BUTTON                             */
/* ARGUMENTS : TAKES THE BUTTON PIN                              */
/* RETURN    : void                                              */
/*****************************************************************/
void Button_init(uint8_t Button_port , uint8_t Button_pin);


/*****************************************************************/
/* FUNCTION TO CHECK THE BUTTON STATUS PRESSED OR NOT            */
/* ARGUMENTS : TAKES THE BUTTON PIN                              */
/* RETURN    : RETURNS BUTTON_t type                             */
/*****************************************************************/
button_t Is_pressed(uint8_t Button_port , uint8_t Button_pin , uint8_t * value);
```

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);
```