Table of Contents

Layered architecture

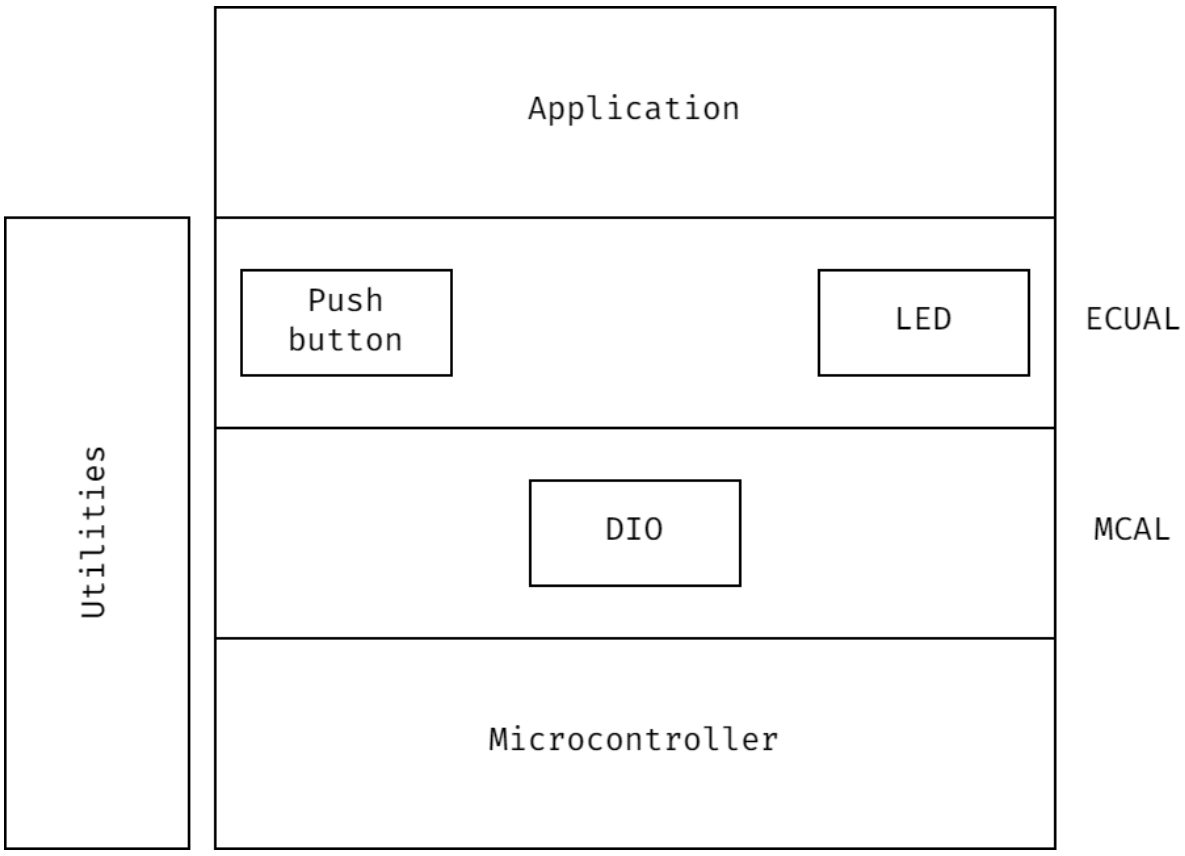

Figure 1: img

Flow chart

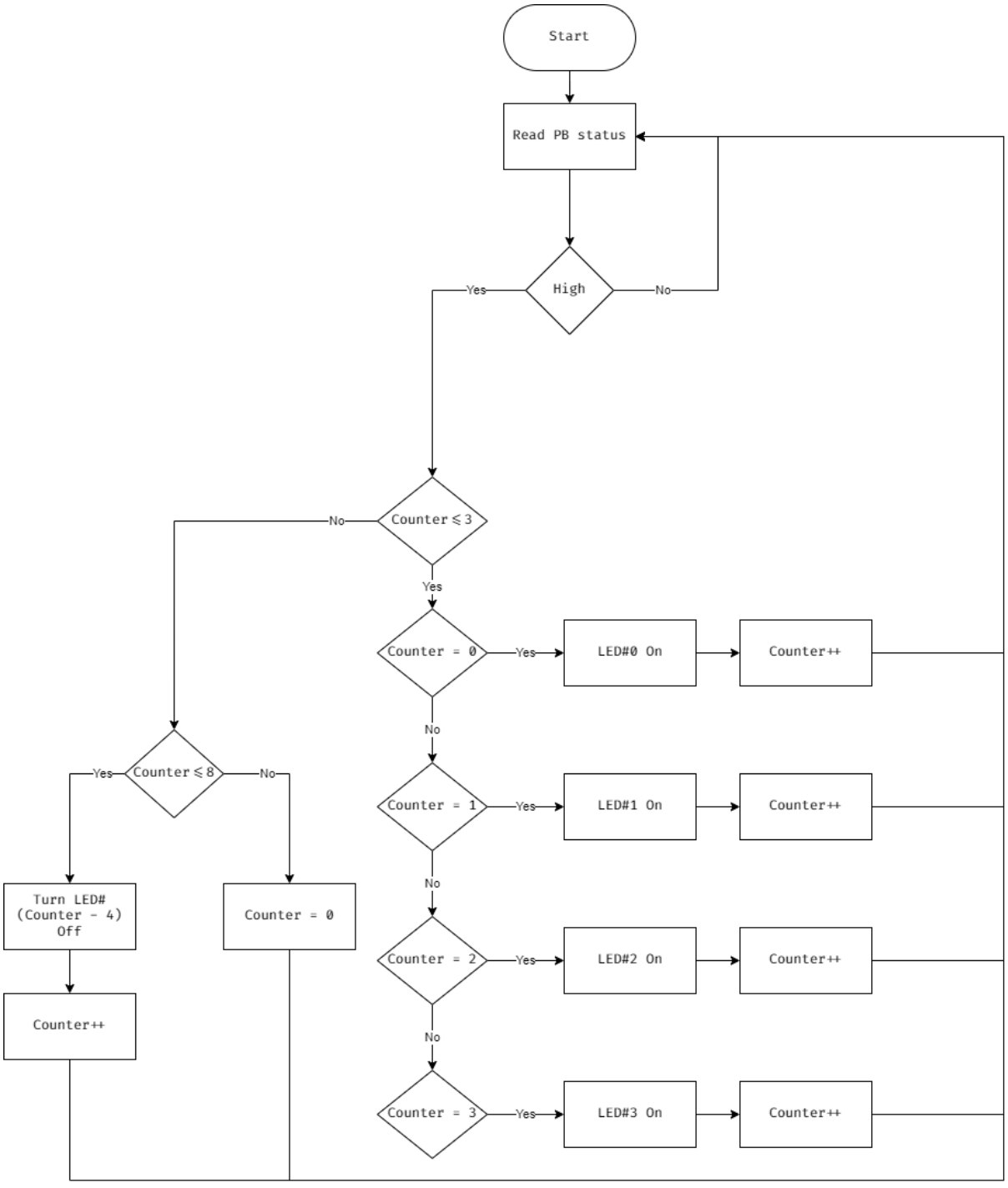
Start
Read PB status
High
Yes
No
Counter ⩽ 3
No
Yes
Counter = 0
Yes
LED#0 On
Counter++
No
Counter ⩽ 8
Yes
No
Counter = 1
Yes
LED#1 On
Counter++
No
Turn LED#
(Counter - 4)
Off
Counter = 0
Counter = 2
Yes
LED#2 On
Counter++
No
Counter++
Counter = 3
Yes
LED#3 On
Counter++

Figure 2: img

System modules/drivers

1. Push button
2. LED
3. DIO
4. Utilities

APIs

DIO

```c
/**
 * @enum EN_DIO_ERROR_STATE
 * @brief Defines the state of DIO functions.
 */
typedef enum EN_DIO_ERROR_STATE {
    DIO_SUCCESS = 0, DIO_PORT_INVALID, DIO_DIRECTION_INVALID,
       DIO_PIN_INVALID
}EN_DIO_ERROR_STATE;

/**
 * @enum EN_DIO_DIRECTION
 * @brief Specifies the state of the pin.
 */
typedef enum EN_DIO_DIRECTION {
    DIO_INPUT = 0, DIO_OUTPUT
}EN_DIO_DIRECTION;

/**
 * @enum EN_DIO_PIN
 * @brief Specifies the number of pin.
 */
typedef enum EN_DIO_PIN {
   PIN0 = 0, PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8
}EN_DIO_PIN;

/**
 * @enum EN_DIO_PORT
 * @brief Specifies the port number.
 * the port number and returns the address of the corresponding port.
 */
typedef enum EN_DIO_PORT {
    PORT_A = 0, PORT_B, PORT_C, PORT_D
}EN_DIO_PORT;

/**
```

```
35   * @enum EN_DIO_LEVEL
36   * @brief Specifies the level of the pin.
37   */
38  typedef enum EN_DIO_LEVEL {
39      DIO_LOW = 0, DIO_HIGH
40  }EN_DIO_LEVEL;
41
42  /**
43   * @struct DIO_Init_t
44   * @brief Holds the configuration of a specific pin of a port.
45   * @var DIO_Init_t::port
46   * Member 'port' sets the port to be configured.
47   * @var DIO_Init_t::pin
48   * Member 'pin' sets the pin to be configured.
49   * @var DIO_Init_t::direction
50   * Member 'direction' sets the direction of the pin.
51   * @var DIO_Init_t::pin_value
52   * Member 'pin_value; contains the value of the pin when it's
       configured as input mode.
53   * @var DIO_Init_t::port_value
54   * Member 'port_value' contains the value to be written to the port
       register if the pin is configured as output.
55   */
56  typedef struct DIO_Init_t {
57      EN_DIO_PORT port;
58      EN_DIO_PIN pin;
59      EN_DIO_DIRECTION direction;
60      union {
61      uint8 pin_value;
62      uint8 port_value;
63      };
64  }DIO_Init_t;
65
66  /**
67   * @brief Initializes the direction of the specified pin.
68   * @param[in] p_config_struct Address of the configuration structure.
69   * @return DIO_PORT_INVALID Port in invalid.
70   * @return DIO_SUCCESS The pin initialization is a success.
71   */
72  EN_DIO_ERROR_STATE DIO_Init(DIO_Init_t *p_config_struct);
73
74  /**
75   * @brief Reads the state of a specific pin.
76   * @param[in] p_config_struct Address of the configuration structure.
77   * @return DIO_PORT_INVALID Port is invalid.
78   * @return DIO_DIRECTION_INVALID Reading from a pin that is configured
       as output.
79   * @return DIO_SUCCESS The read operation is a success.
80   */
81  EN_DIO_ERROR_STATE DIO_ReadPin(DIO_Init_t *p_config_struct);
82
```

```
83 /**
84  * @brief Write a specific level to a specified pin.
85  * @param[in] p_config_struct Address of the configuration structure.
86  * @return DIO_PORT_INVALID Port is invalid.
87  * @return DIO_DIRECTION_INVALID Writing to a pin that is configured as
        input.
88  * @return DIO_SUCCESS The write operation is a success.
89  */
90 EN_DIO_ERROR_STATE DIO_WritePin(DIO_Init_t *p_config_struct);
91
92 /**
93  * @brief Toggles the current level of a pin.
94  * @param[in] p_config_struct Address of the configuration structure.
95  *  @return DIO_PORT_INVALID Port is invalid.
96  * @return DIO_DIRECTION_INVALID Toggle a pin that is configured as
       input.
97  * @return DIO_SUCCESS The toggle operation is a success.
98  */
99 EN_DIO_ERROR_STATE DIO_TogglePin(DIO_Init_t *p_config_struct);
```

LED

```
1  typedef enum EN_LED_API_STATE {
2      LED_SUCCESS = 0, LED_PORT_INVALID, LED_DIRECTION_INVALID
3  }EN_LED_API_STATE;
4
5  typedef enum EN_LED_STATUS {
6      LED_OFF = 0, LED_ON
7  }EN_LED_STATUS;
8
9  /**
10  * @struct LED_Init_t
11  * @brief Holds the port number and the pin number of the LED.
12  * @var LED_Init_t::port
13  * Member 'port' specifies the port number.
14  * @var LED_Init_t::pin
15  * Member 'pin' specifies the pin number.
16  */
17 typedef struct LED_Init_t {
18     EN_DIO_PORT port;
19     EN_DIO_PIN pin;
20     EN_LED_STATUS led_status;
21 }LED_Init_t;
22
23 /**
24  * @brief Initializes the pin attached to the LED.
25  * @param[in] p_config_struct Address of the configuration structure.
26  */
27 EN_LED_API_STATE LED_Init(LED_Init_t *p_led_config_struct);
```

```
28
29 /**
30  * @brief Turns the LED on.
31  * @param[in] p_config_struct Address of the configuration structure.
32  */
33 EN_LED_API_STATE LED_On(LED_Init_t *p_led_config_struct);
34
35 /**
36  * @brief Turns the LED off.
37  * @param[in] p_config_struct Address of the configuration structure.
38  */
39 EN_LED_API_STATE LED_Off(LED_Init_t *p_led_config_struct);
```

Push button

```
1  typedef enum EN_PB_API_STATE {
2      PB_SUCCESS = 0, PB_PORT_INVALID, PB_DIRECTION_INVALID
3  }EN_PB_API_STATE;
4
5  /**
6   * @struct PB_Init_t
7   * @var PB_Init_t::port
8   * Member 'port' specifies the port which the push button is connected
        to.
9   * @var PB_Init::pin
10  * Member 'pin' specifies the pin number which the push button is
        connected to.
11  */
12 typedef struct PB_Init_t {
13     EN_DIO_PORT port;
14     EN_DIO_PIN pin;
15     uint8 pb_status;
16 }PB_Init_t;
17
18 /**
19  * @enum EN_PB_LEVEL
20  * @brief Specifies the state of push button.
21  */
22 typedef enum EN_PB_LEVEL {
23     PB_LOW = 0, PB_HIGH
24 }EN_PB_LEVEL;
25
26 /**
27  * @brief Initializes the state of the pin connected to the push button
        .
28  * @param[in] p_config_struct Address of the configuration structure.
29  */
30 EN_PB_API_STATE PB_Init(PB_Init_t *p_pb_config_struct);
31
```

```
32 /**
33  * @brief Reads the current state of the push button.
34  * @param[in] p_config_struct Address of the configuration structure.
35  * @return The current state of the push button.
36  */
37 EN_PB_API_STATE PB_ReadState(PB_Init_t *p_pb_config_struct);
```