High Level Design

Modules description

DIO Module

This module is responsible for the initialization of the MCU pins by specifying its direction, if also provides methods to read and write data on the pins.

Driver's documentation

DIO Module

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

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

/**
 * @enum en_DIO_pinNum
 * @brief Specifies the number of pin.
 */
typedef enum {
   DIO_PIN0 = 0, DIO_PIN1, DIO_PIN2, DIO_PIN3, DIO_PIN4, DIO_PIN5,
DIO_PIN6, DIO_PIN7, DIO_PIN8
}en_DIO_pinNum;
```

```
/**
 * @enum en_DIO_portNum
 * @brief Specifies the port number.
 * the port number and returns the address of the corresponding port.
 */
typedef enum {
      DIO_PORT_A = 0, DIO_PORT_B, DIO_PORT_C, DIO_PORT_D
}en_DIO_portNum;

/**
 * @enum en_DIO_pinLevel
 * @brief Specifies the level of the pin.
 */
typedef enum {
      DIO_LOW = 0, DIO_HIGH
}en_DIO_pinLevel;

/**
 * @struct st_DIO_config
 * @brief Holds the configuration of a specific pin of a port.
 * @var st_DIO_config::port
 * Member 'port' sets the port to be configured.
 * @var st_DIO_config::pin
 * Member 'pin' sets the pin to be configured.
 * @var st_DIO_config::direction
 * Member 'direction' sets the direction of the pin.
 * @var st_DIO_config::pin_value
 * Member 'pin_value; contains the value of the pin when it's configured as
input mode.
 * @var st_DIO_config::port_value
 * Member 'port_value' contains the value to be written to the port
register if the pin is configured as output.
 */
typedef struct {
      en_DIO_portNum port;
      en_DIO_pinNum pin;
      en_DIO_direction direction;
      union readWrite{
      uint8 pin_value;
      uint8 port_value;
      };
```

```c
}st_DIO_config;

/**
 * @brief Initializes the direction of the specified pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port in invalid.
 * @return DIO_SUCCESS The pin initialization is a success.
 */
en_DIO_errorState DIO_Init(st_DIO_config *p_config_struct);

/**
 * @brief Reads the state of a specific pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Reading from a pin that is configured as
output.
 * @return DIO_SUCCESS The read operation is a success.
 */
en_DIO_errorState DIO_ReadPin(st_DIO_config *p_config_struct);

/**
 * @brief Write a specific level to a specified pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Writing to a pin that is configured as
input.
 * @return DIO_SUCCESS The write operation is a success.
 */
en_DIO_errorState DIO_WritePin(st_DIO_config *p_config_struct);

/**
 * @brief Toggles the current level of a pin.
 * @param[in] p_config_struct Address of the configuration structure.
 *  @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Toggle a pin that is configured as input.
 * @return DIO_SUCCESS The toggle operation is a success.
 */
en_DIO_errorState DIO_TogglePin(st_DIO_config *p_config_struct);
```

LED module

```c
/**
 * @enum en_LED_errorState
 * @brief Defines the state of LED functions.
 */
typedef enum EN_LED_API_STATE {
      LED_SUCCESS = 0, LED_PORT_INVALID, LED_STATUS_INVALID
}en_LED_errorState;

/**
 * @enum en_LED_state
 * @brief Defines the LED status.
 */
typedef enum EN_LED_STATUS {
      LED_OFF = 0, LED_ON
}en_LED_state;

/**
 * @struct st_LED_config
 * @brief Holds the port number and the pin number of the LED.
 * @var st_LED_config::port
 * Member 'port' specifies the port number.
 * @var st_LED_config::pin
 * Member 'pin' specifies the pin number.
 * @var LED_INIT_t::led_status
 * Member 'led_status' specifies the status of the LED.
 */
typedef struct {
      en_DIO_portNum port;
      en_DIO_pinNum pin;
      en_LED_state led_status;
}st_LED_config;

/**
 * @brief Initializes the pin attached to the LED.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return LED_SUCCESS Initialization is done successfully.
 */
en_LED_errorState LED_Init(st_LED_config *p_led_config_struct);
```

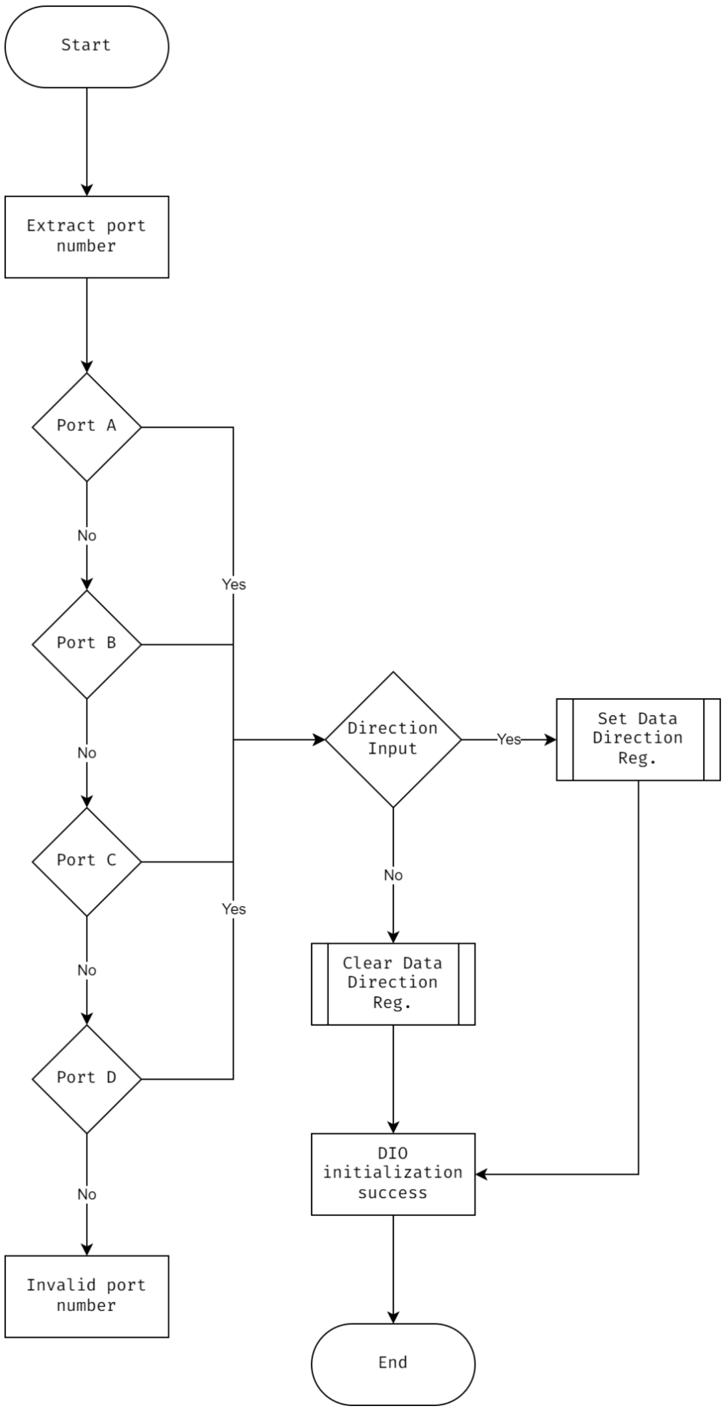
```c
/**
 * @brief Turns the LED on.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return LED_PORT_INVALID
 * @return LED_SUCCESS
 */
en_LED_errorState LED_On(st_LED_config *p_led_config_struct);

/**
 * @brief Turns the LED off.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return LED_PORT_INVALID
 * @return LED_SUCCESS
 */
en_LED_errorState LED_Off(st_LED_config *p_led_config_struct);
```

Low Level Design

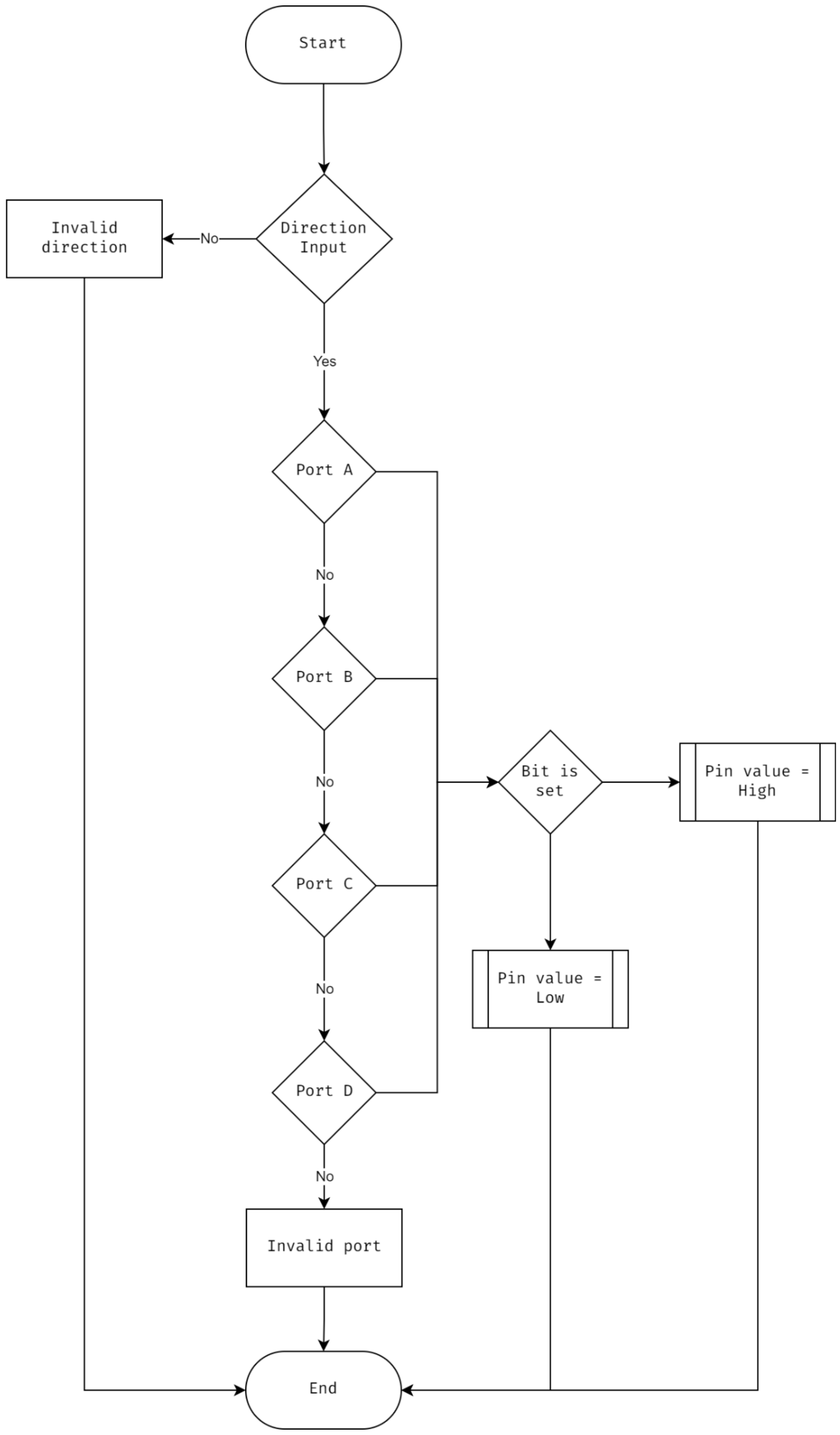
Overview of MCAL layer

DIO module

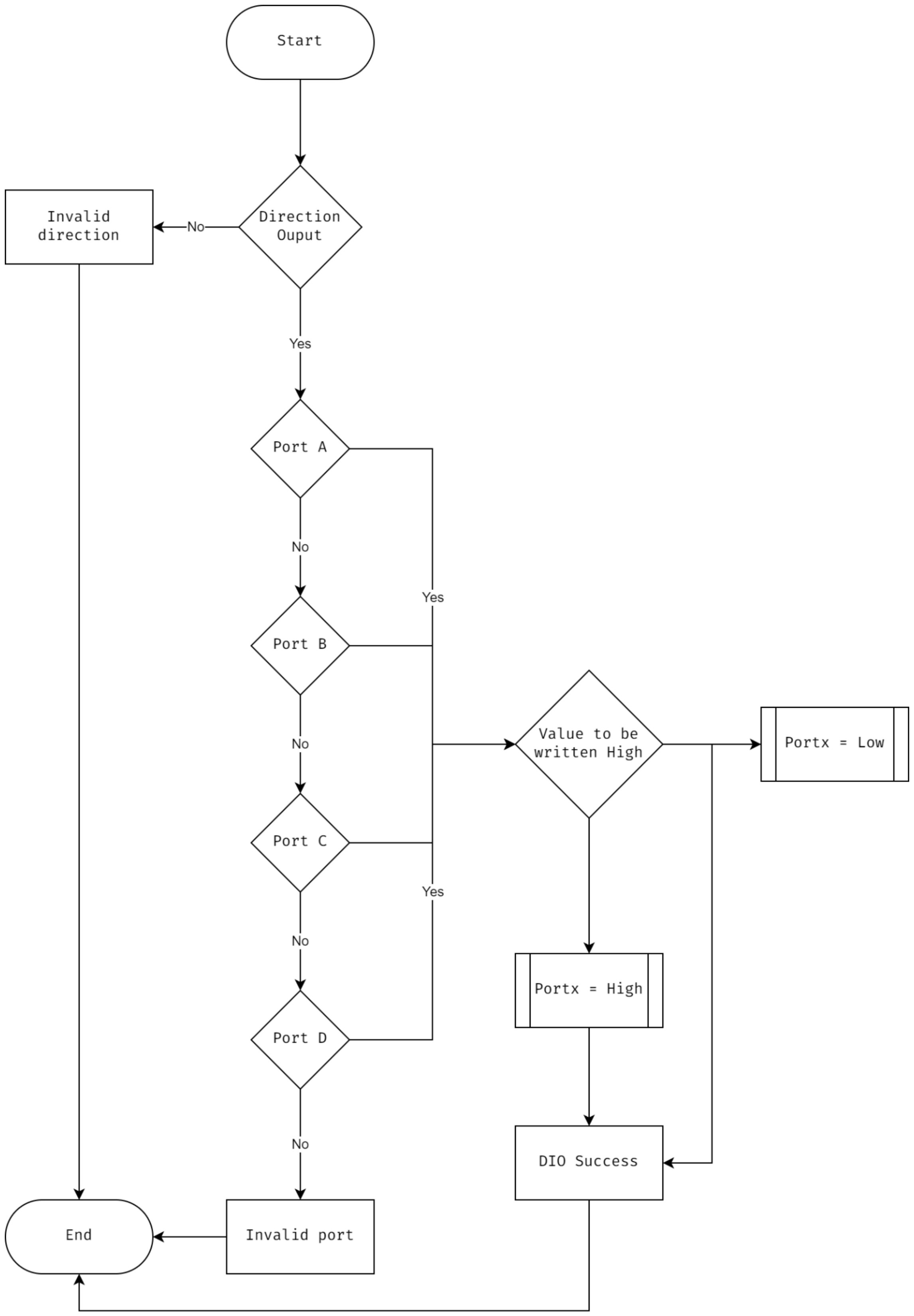
Dio_Init flowchart



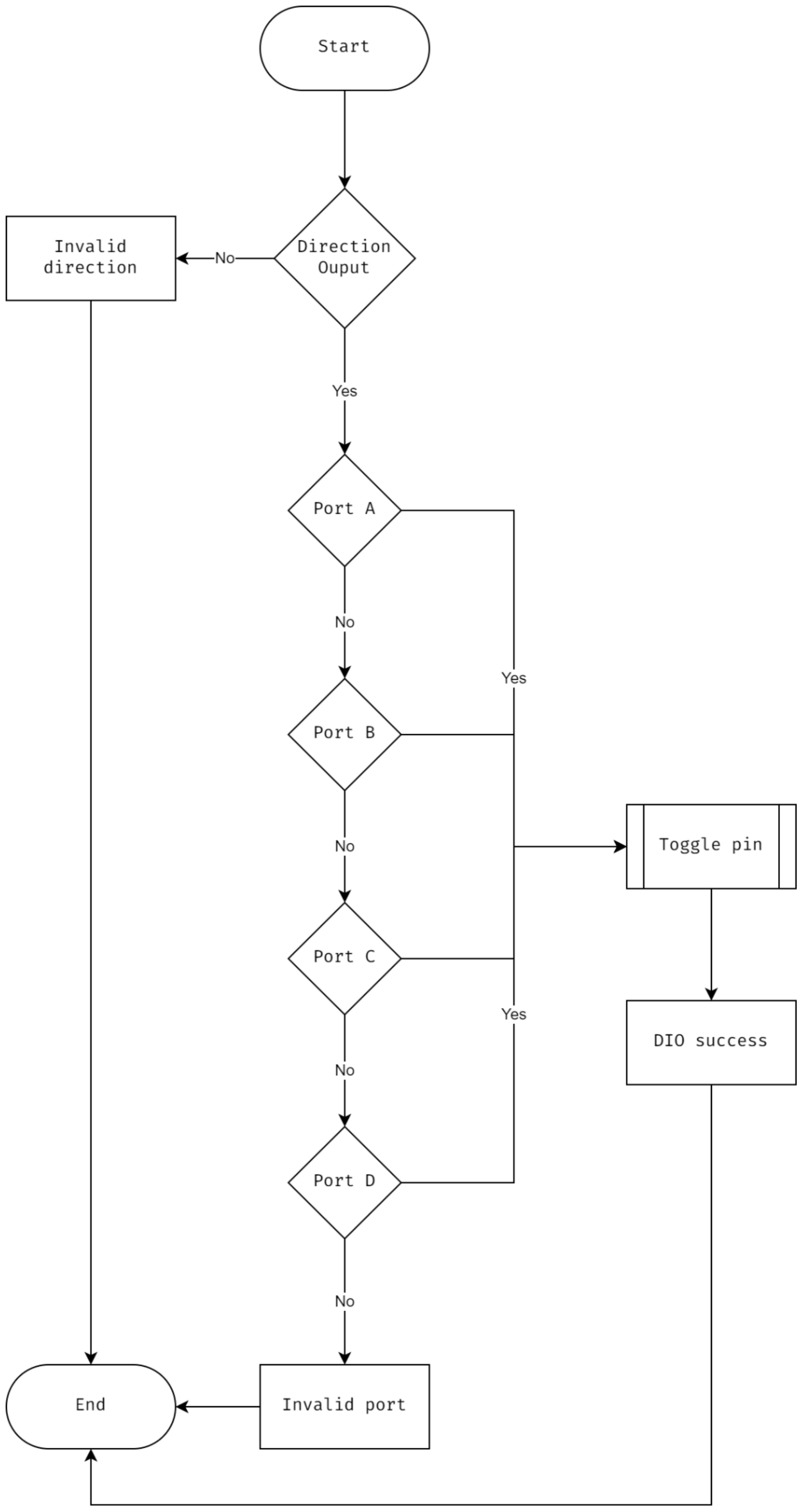
DIO_ReadPin


Start
Direction Input
No
Invalid direction
Yes
Port A
No
Port B
No
Port C
No
Port D
No
Invalid port
Bit is set
Pin value = High
Pin value = Low
End

DIO_WritePin

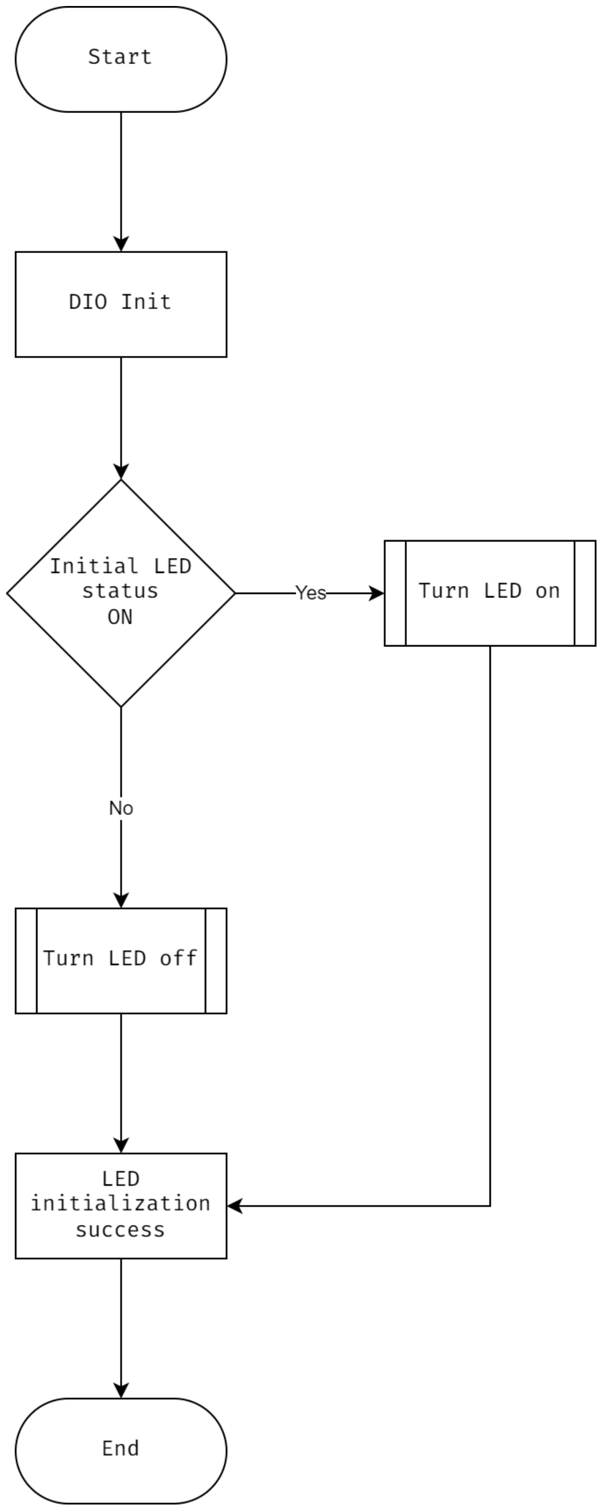


DIO_TogglePin

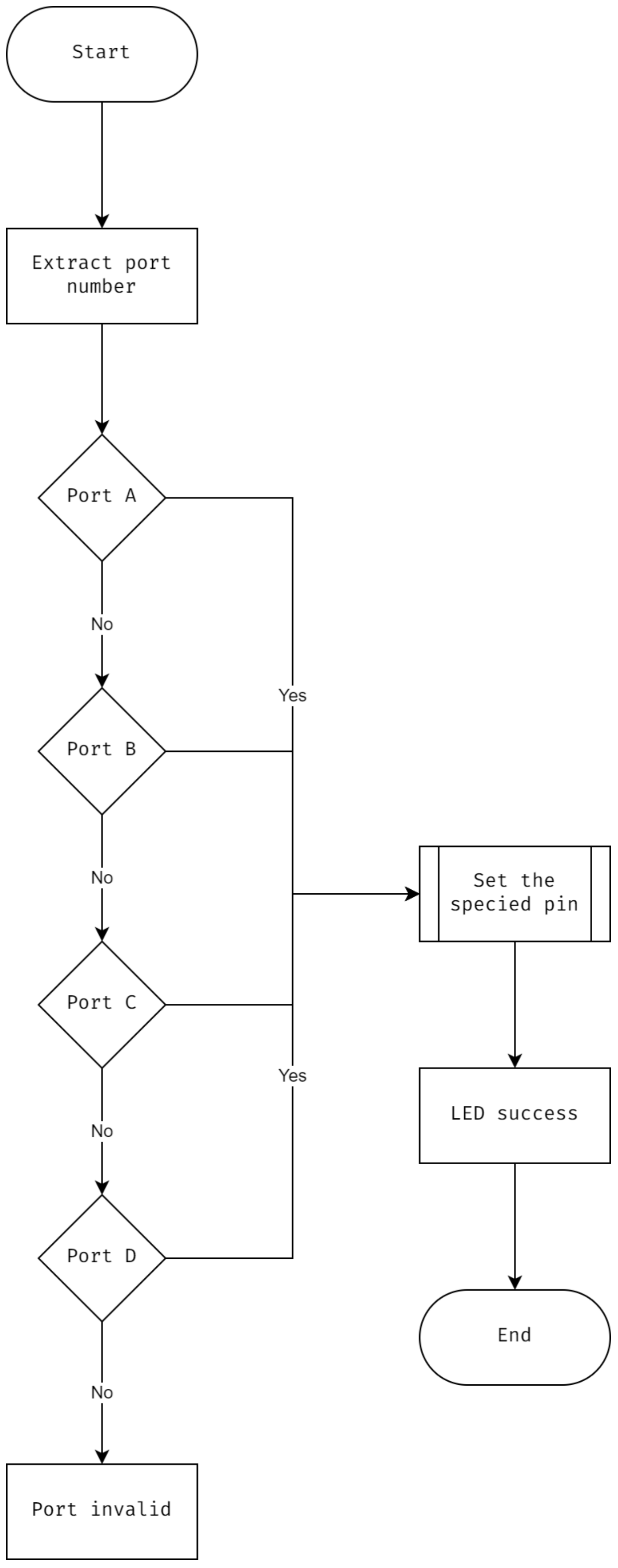


LED module

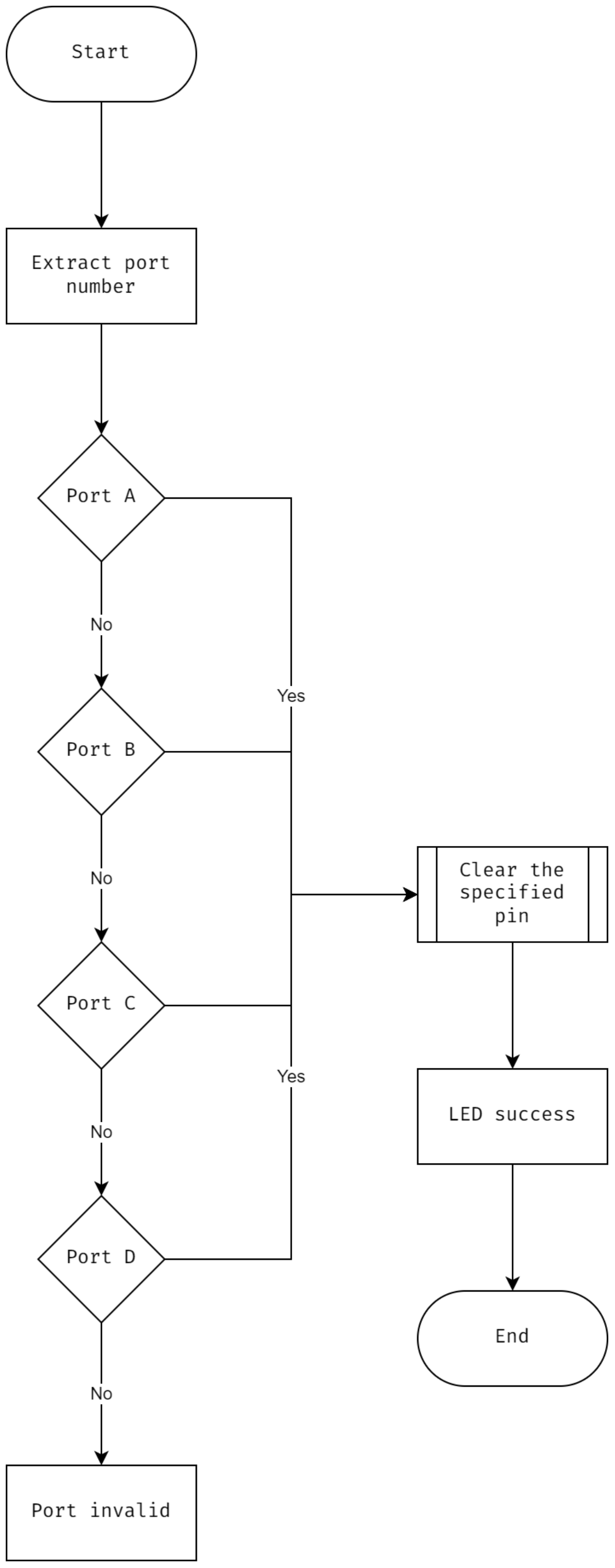
LED_Init



LED_On


Start
Extract port number
Port A
No
Yes
Port B
No
Set the specied pin
Port C
Yes
No
LED success
Port D
End
No
Port invalid

LED_Off


Start
Extract port number
Port A
No
Yes
Port B
No
Clear the specified pin
Port C
Yes
No
LED success
Port D
No
End
Port invalid