Trafic Light LED

Generated by Doxygen 1.9.5

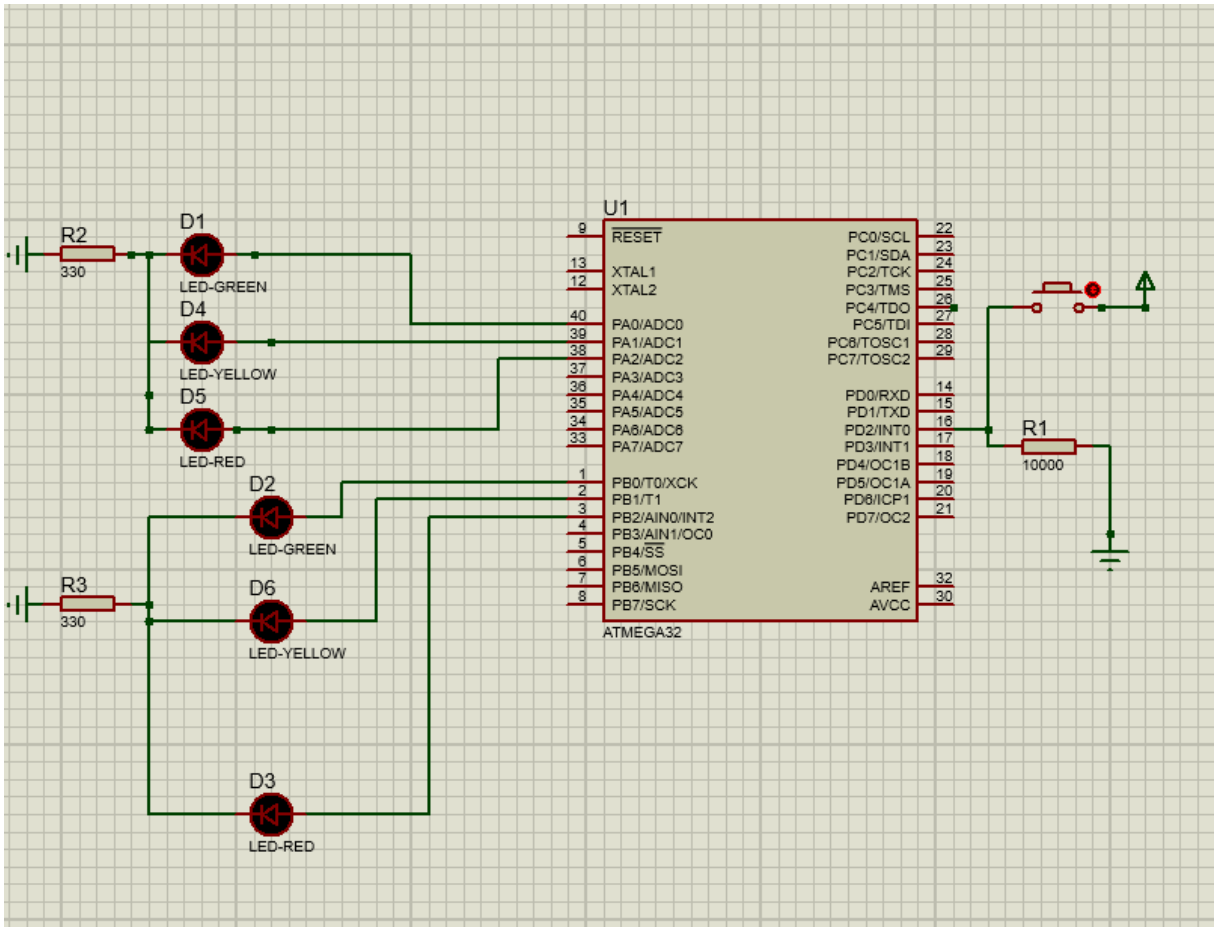
Chapter 1

System Design Traffic Light project

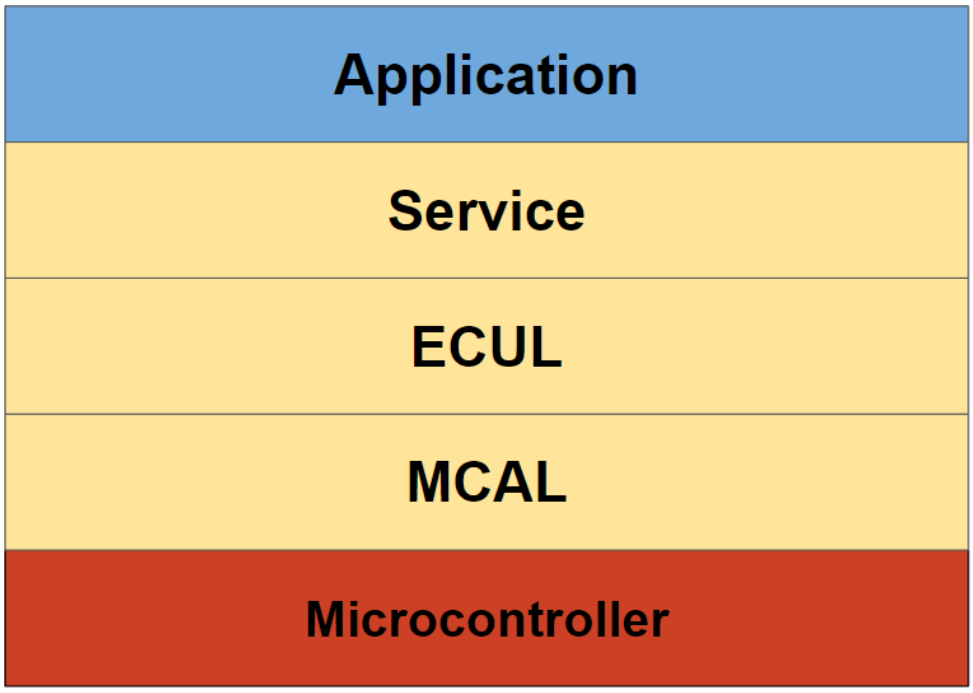
1.1 Introduction

The purpose of this document is to provide the illustration of the Traffic Light prject in the following:
1-System Design (proteus)
2-System layers
3-System Drivers
4-state machine

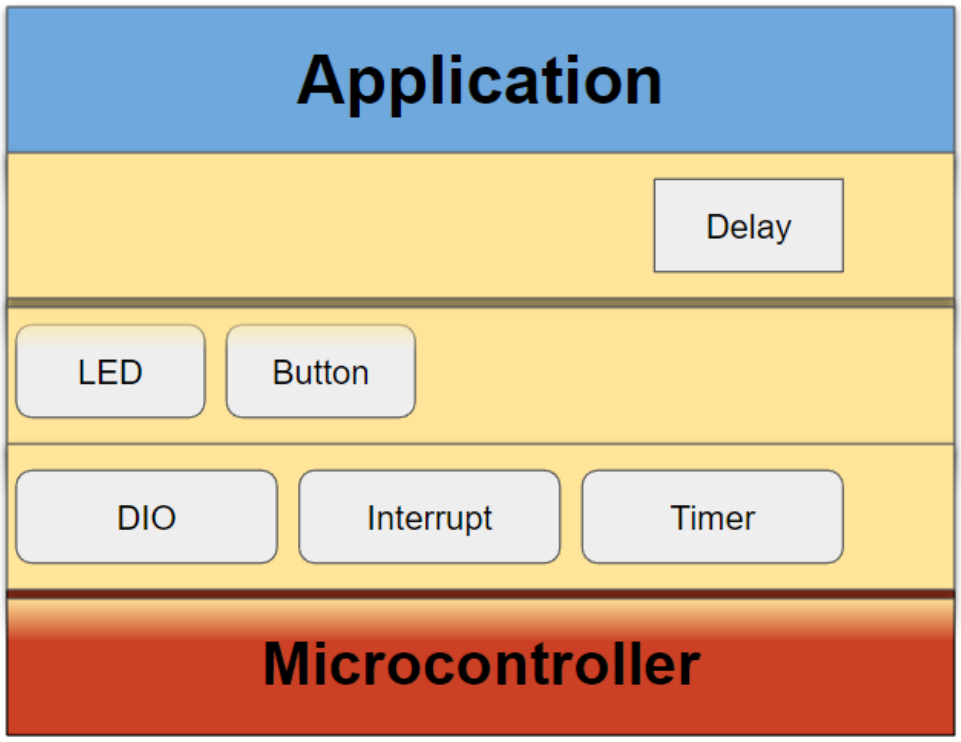
1.2 System Design (proteus)



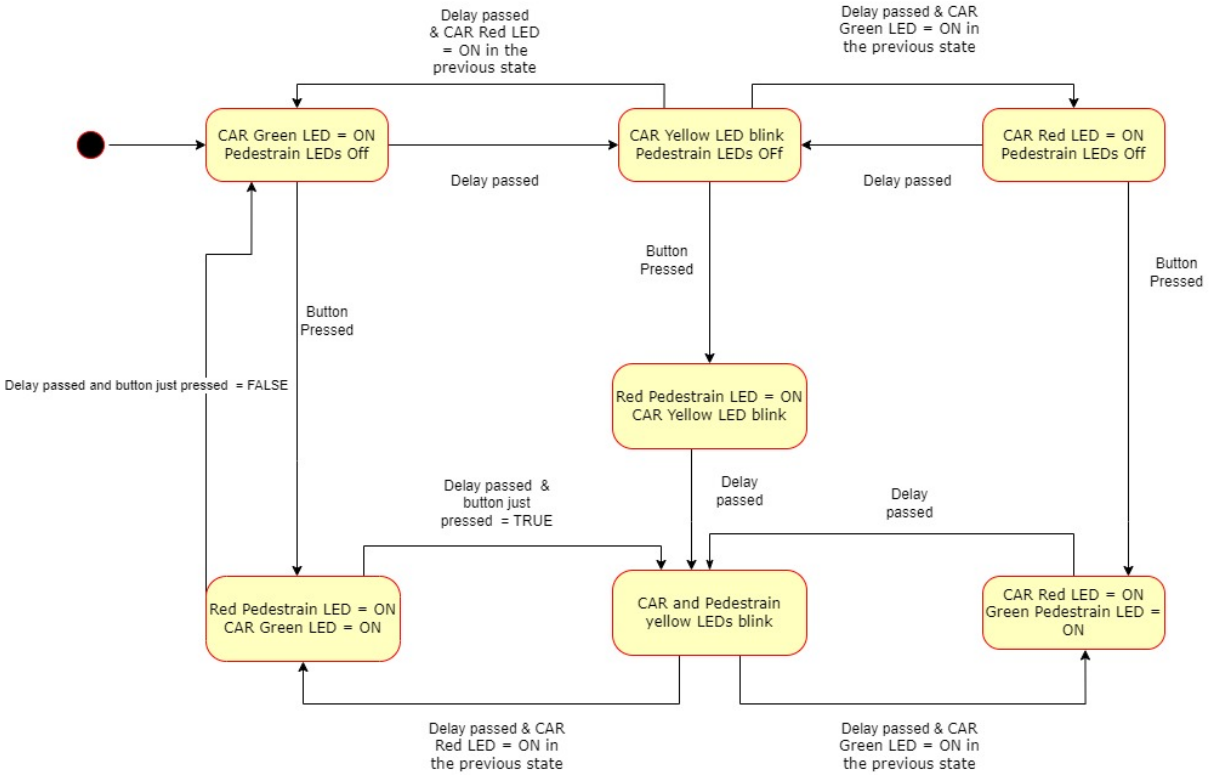
1.3 System layers



1.4 System Drivers



1.5 state machine



Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

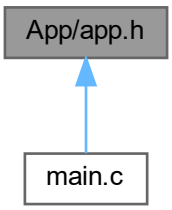
Chapter 3

File Documentation

3.1 App/app.h File Reference

Coniatins the needed definition of the macros, variables and functions used by the application.

This graph shows which files directly or indirectly include this file:



Macros

- #define ON_DELAY 5000

  The delay time for the LEDs.
- #define TOGGLE_DELAY 1000

  The delay used for toggling of the yellow led.

Typedefs

- typedef enum Mode Mode

  enum to hold in which mode the program is
- typedef enum LedMode LedMode

  enum to hold in which led state the program is

Enumerations

- enum Mode { Normal , Pedestrian }

  enum to hold in which mode the program is
- enum LedMode { Green , Yellow , Red , NON }

  enum to hold in which led state the program is

Functions

- void AppStart ()

  AppStart determine which mode the program is on and in return will detemine the illumination of the car and pedestrian LEDs accordingly.
- Status LEDS_OFF ()
- Status Blink_CAR_YELLOW ()

  blink the car yellow LED for 5 seconds
- Status Blink_Both_YELLOW ()

  blink the both yellow LEDs for 5 seconds

3.1.1 Detailed Description

Coniatins the needed definition of the macros, variables and functions used by the application.

Author

Moataz

Date

September 2022

3.1.2 Macro Definition Documentation

3.1.2.1 ON_DELAY

```
#define ON_DELAY 5000
```

The delay time for the LEDs.

3.1.2.2 TOGGLE_DELAY

```
#define TOGGLE_DELAY 1000
```

The delay used for toggling of the yellow led.

3.1.3 Typedef Documentation

3.1.3.1 LedMode

```
typedef enum LedMode LedMode
```

enum to hold in which led state the program is

3.1.3.2 Mode

```
typedef enum Mode Mode
```

enum to hold in which mode the program is

3.1.4 Enumeration Type Documentation

3.1.4.1 LedMode

```
enum LedMode
```

enum to hold in which led state the program is

Enumerator

Green	The Green led is On.
Yellow	The yellow led is blinking.
Red	The red led is on.
NON	Non of the LEDS are on.

3.1.4.2 Mode

```
enum Mode
```

enum to hold in which mode the program is

Enumerator

Normal	The program is in Car normal mode.
Pedestrian	The program is in pedestrian mode.

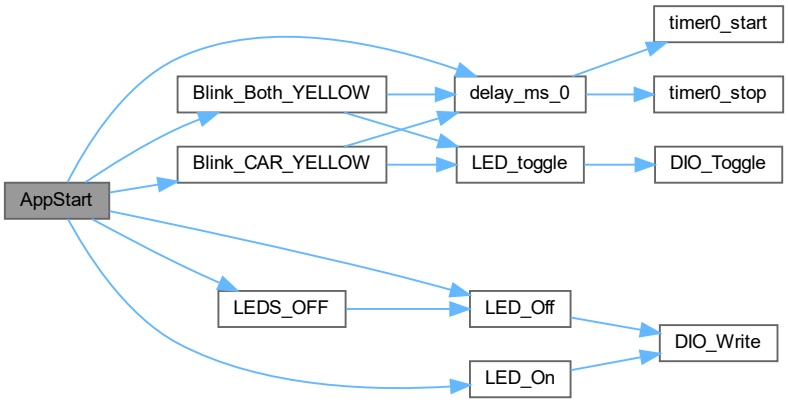
3.1.5 Function Documentation

3.1.5.1 AppStart()

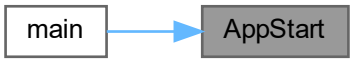
```
void AppStart ( )
```

AppStart determine which mode the program is on and in return will detemine the illumination of the car and pedestrian LEDs accordingly.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.5.2 Blink_Both_YELLOW()

```
Status Blink_Both_YELLOW ( )
```

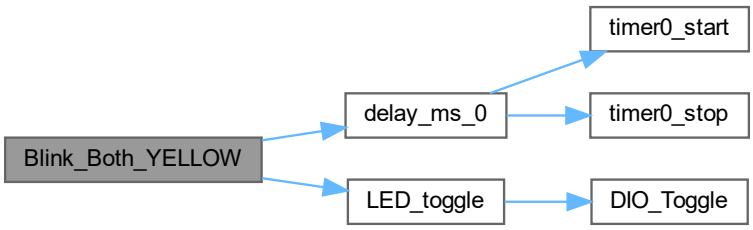
blink the both yellow LEDs for 5 seconds

calls LED toggle passing the car yellow LED parameters then delay 1 ms in a loop of 5 iterations

Returns

Status returns not ok if failure occurs during toggling or delay, otherwise return Ok

Here is the call graph for this function:



Here is the caller graph for this function:

3.1.5.3 Blink_CAR_YELLOW()

```
Status Blink_CAR_YELLOW ( )
```

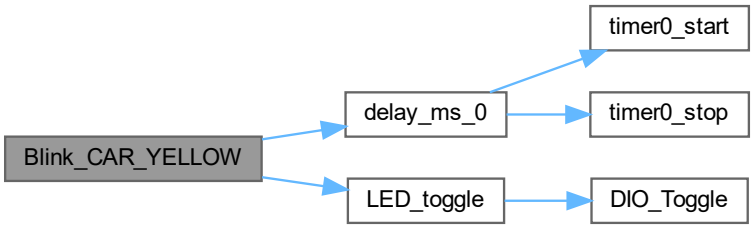
blink the car yellow LED for 5 seconds

calls LED toggle passing the car yellow LED parameters then delay 1 ms in a loop of 5 iterations

Returns

Status returns not ok if failure occurs during toggling or delay, otherwise return Ok

Here is the call graph for this function:



Here is the caller graph for this function:

3.1.5.4 LEDS_OFF()

```
Status LEDS_OFF ( )
```

Returns

Status

Here is the call graph for this function:

Here is the caller graph for this function:

3.2 app.h

Go to the documentation of this file.

```
1 /*****************************************************************/
10 #ifndef APP_H_
11 #define APP_H_
12
17 typedef enum Mode{
19     Normal,
21     Pedestrian
22     }Mode;
23
28 typedef enum  LedMode{
30     Green,
32     Yellow,
34     Red,
36     NON
37     }LedMode;
38
40 #define ON_DELAY 5000
42 #define TOGGLE_DELAY 1000
43
44 //Function Prototypes
45 void AppStart();
46 Status LEDS_OFF();
47 Status Blink_CAR_YELLOW();
48 Status Blink_Both_YELLOW();
49 #endif /* APP_H_ */
```

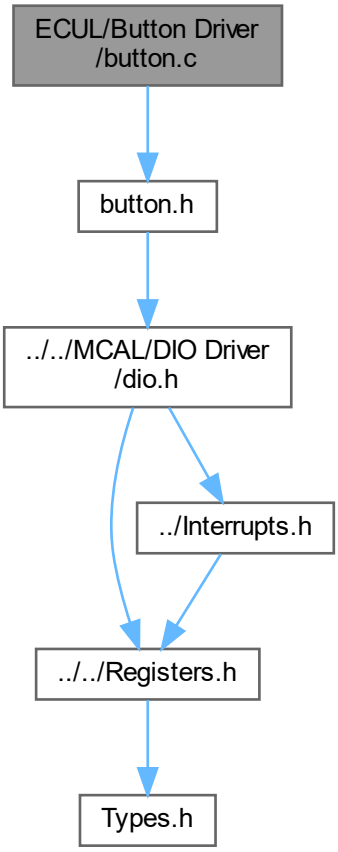
3.3 Doc_pages/System Design.md File Reference

3.4 ECUL/Button Driver/button.c File Reference

button Driver that intialize and operates the DIO Pins as inputs connected to buttons

```
#include "button.h"
```

Include dependency graph for button.c:



Functions

- Status BUTTON_Init (uint8_t buttonPort, uint8_t buttonPin)

  Initialize DIO PIN as input.
- Status BUTTON_read (uint8_t buttonPort, uint8_t buttonPin, volatile uint8_t *value)

  Reads the input comming from the button connected to the microcontroller pin.

3.4.1 Detailed Description

button Driver that intialize and operates the DIO Pins as inputs connected to buttons

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

3.4.2 Function Documentation

3.4.2.1 BUTTON_Init()

```
Status BUTTON_Init (
            uint8_t buttonPort,
            uint8_t buttonPin )
```

Initialize DIO PIN as input.

calls the DIO_init passing the dirction value as IN to set provided pin in the port as input

Parameters

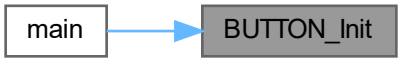
buttonPort	contains in which port the pin will be intialized as an input
buttonPin	contains which pin will be intialized as an input

Returns

Status returns Not_Ok if the intiailization fails, ekse returns Ok.

Here is the call graph for this function:

Here is the caller graph for this function:



3.4.2.2 BUTTON_read()

```
Status BUTTON_read (
            uint8_t buttonPort,
            uint8_t buttonPin,
            volatile uint8_t * value )
```

Reads the input comming from the button connected to the microcontroller pin.

calls the DIO passing the pointer value to get the status of the PIN

Parameters

buttonPort	port to be accessed (PORTA, PORTB, PORTC, PORTD)
buttonPin	Pin to be accessed
value	pointer that holds the state of the pin (high or low)

Returns

Status

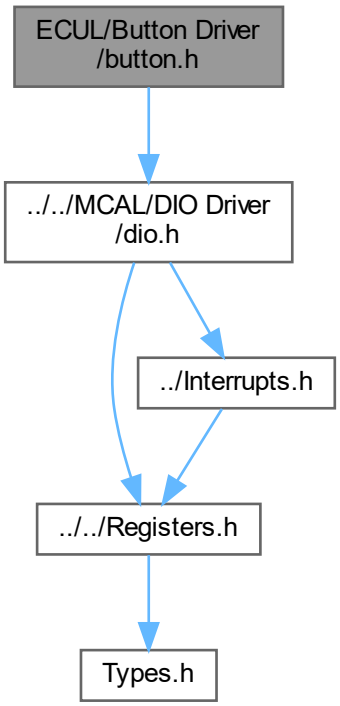
Here is the call graph for this function:

3.5 ECUL/Button Driver/button.h File Reference

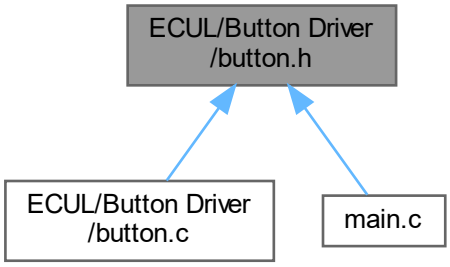
button Driver that intialize and operates the DIO Pins as inputs connected to buttons

```
#include "../../MCAL/DIO Driver/dio.h"
```

Include dependency graph for button.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define PEDESTRIAN_BUTTON PIN2

  Definition of the Pedestrian button to be PIN2 in PORTB.

Functions

- Status BUTTON_Init (uint8_t buttonPort, uint8_t buttonPin)
  Initialize DIO PIN as input.
- Status BUTTON_read (uint8_t buttonPort, uint8_t buttonPin, volatile uint8_t *value)
  Reads the input comming from the button connected to the microcontroller pin.

3.5.1 Detailed Description

button Driver that intialize and operates the DIO Pins as inputs connected to buttons

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

Copyright (c) 2022

3.5.2 Macro Definition Documentation

3.5.2.1 PEDESTRIAN_BUTTON

```
#define PEDESTRIAN_BUTTON PIN2
```

Definition of the Pedestrian button to be PIN2 in PORTB.

3.5.3 Function Documentation

3.5.3.1 BUTTON_Init()

```
Status BUTTON_Init (
            uint8_t buttonPort,
            uint8_t buttonPin )
```

Initialize DIO PIN as input.

calls the DIO_init passing the dirction value as IN to set provided pin in the port as input

Parameters

buttonPort	contains in which port the pin will be intialized as an input
buttonPin	contains which pin will be intialized as an input

Returns

Status returns Not_Ok if the intiailization fails, ekse returns Ok.

Here is the call graph for this function:

Here is the caller graph for this function:

3.5.3.2 BUTTON_read()

```
Status BUTTON_read (
            uint8_t buttonPort,
            uint8_t buttonPin,
            volatile uint8_t * value )
```

Reads the input comming from the button connected to the microcontroller pin.

calls the DIO passing the pointer value to get the status of the PIN

Parameters

buttonPort	port to be accessed (PORTA, PORTB, PORTC, PORTD)
buttonPin	Pin to be accessed
value	pointer that holds the state of the pin (high or low)

Returns

Status

Here is the call graph for this function:

3.6 button.h

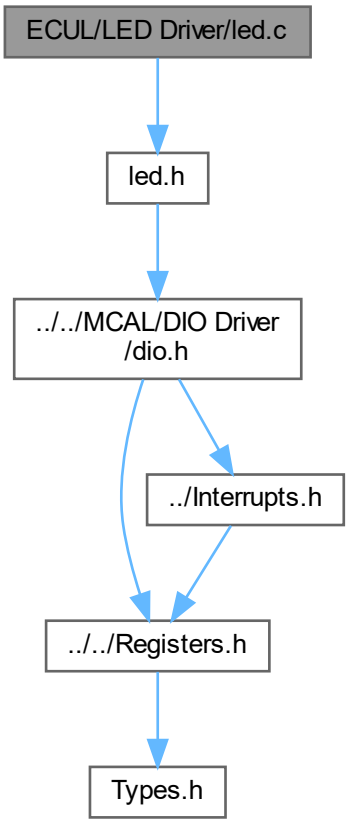
Go to the documentation of this file.

```
1
13 #ifndef BUTTON_H_
14 #define BUTTON_H_
15
16 //includes
17 #include "../../MCAL/DIO Driver/dio.h"
18
19 //Defines
20
22 #define PEDESTRIAN_BUTTON   PIN2
23
24
25 //function prototypes
26 Status BUTTON_Init(uint8_t buttonPort, uint8_t buttonPin);
27 Status BUTTON_read(uint8_t buttonPort, uint8_t buttonPin, volatile uint8_t *value);
28
29
30 #endif /* INCFILE1_H_ */
```

3.7 ECUL/LED Driver/led.c File Reference

```
#include "led.h"
```

Include dependency graph for led.c:



Functions

- Status LED_Init (uint8_t ledPort, uint8_t ledPin)

  Initialize the LED PIN.
- Status LED_On (uint8_t ledPort, uint8_t ledPin)

  Turns LED on.
- Status LED_Off (uint8_t ledPort, uint8_t ledPin)

  Turns LED Off.
- Status LED_toggle (uint8_t ledPort, uint8_t ledPin)

  Toggle the status of the LED.

3.7.1 Detailed Description

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

3.7.2 Function Documentation

3.7.2.1 LED_Init()

```
Status LED_Init (
            uint8_t ledPort,
            uint8_t ledPin )
```

Initialize the LED PIN.

Calls DIO_Init with specific port and pin with OUT direction

Parameters

ledPort	Port containing the PIN that will be set as output
ledPin	PIN to be set as output

See also

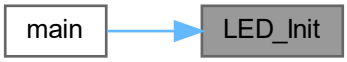
DIO_init()

Returns

Status NOT_OK if initialization failed, return OK otherwise

Here is the call graph for this function:

Here is the caller graph for this function:



3.7.2.2 LED_Off()

```
Status LED_Off (
            uint8_t ledPort,
            uint8_t ledPin )
```

Turns LED Off.

Calls DIO_Write with specific port and pin to set the pin to LOW

Parameters

ledPort	Port containing the PIN that will be set to low
ledPin	PIN to be set as output

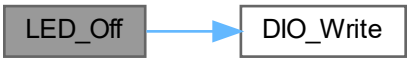
See also

DIO_Write()

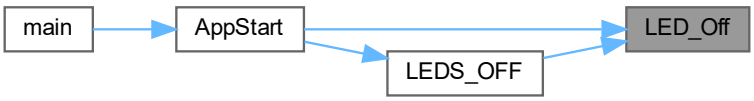
Returns

Status return if turning off the LED fails, return OK otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



3.7.2.3 LED_On()

```
Status LED_On (
            uint8_t ledPort,
            uint8_t ledPin )
```

Turns LED on.

Calls DIO_Write with specific port and pin to set the pin to high

Parameters

ledPort	Port containing the PIN that will be set to HIGH
ledPin	PIN to be set as output

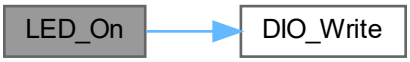
See also

DIO_Write()

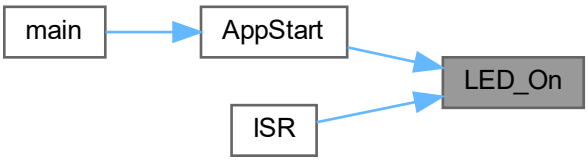
Returns

Status return if turning on the LED fails, return OK otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



3.7.2.4 LED_toggle()

```
Status LED_toggle (
            uint8_t ledPort,
            uint8_t ledPin )
```

Toggle the status of the LED.

Calls DIO_Toggle with specific port and pin to flip the status of the pin

Parameters

ledPort	Port containing the PIN that will be flipped
ledPin	PIN to be set as output

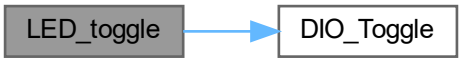
See also

DIO_Toggle()

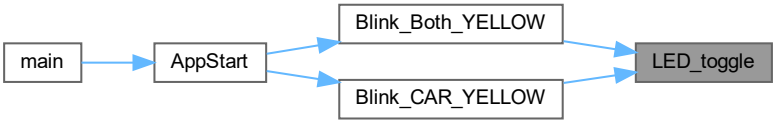
Returns

Status return if toggling the LED fails, return OK otherwise

Here is the call graph for this function:



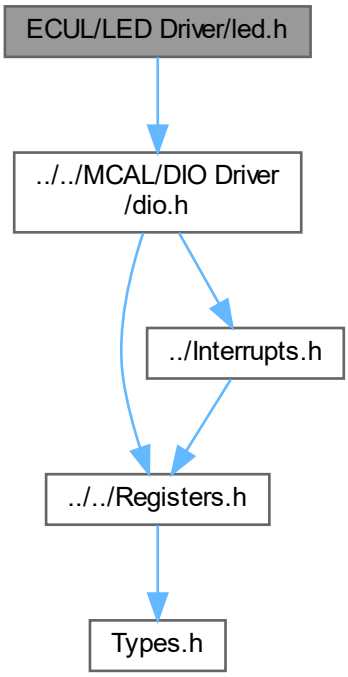
Here is the caller graph for this function:



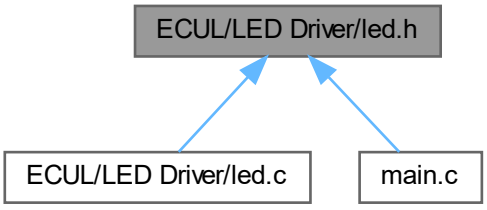
3.8 ECUL/LED Driver/led.h File Reference

```
#include "../../MCAL/DIO Driver/dio.h"
```

Include dependency graph for led.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define CAR_LED_GREEN PIN0
- #define CAR_LED_YELLOW PIN1
- #define CAR_LED_RED PIN2
- #define PEDES_LED_GREEN PIN0
- #define PEDES_LED_YELLOW PIN1
- #define PEDES_LED_RED PIN2

Functions

- Status LED_Init (uint8_t ledPort, uint8_t ledPin)

  Initialize the LED PIN.
- Status LED_On (uint8_t ledPort, uint8_t ledPin)

  Turns LED on.
- Status LED_Off (uint8_t ledPort, uint8_t ledPin)

  Turns LED Off.
- Status LED_toggle (uint8_t ledPort, uint8_t ledPin)

  Toggle the status of the LED.

3.8.1 Detailed Description

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

Copyright (c) 2022

3.8.2 Macro Definition Documentation

3.8.2.1 CAR_LED_GREEN

```
#define CAR_LED_GREEN PIN0
```

3.8.2.2 CAR_LED_RED

```
#define CAR_LED_RED PIN2
```

3.8.2.3 CAR_LED_YELLOW

```
#define CAR_LED_YELLOW PIN1
```

3.8.2.4 PEDES_LED_GREEN

```
#define PEDES_LED_GREEN PIN0
```

3.8.2.5 PEDES_LED_RED

```
#define PEDES_LED_RED PIN2
```

3.8.2.6 PEDES_LED_YELLOW

```
#define PEDES_LED_YELLOW PIN1
```

3.8.3 Function Documentation

3.8.3.1 LED_Init()

```
Status LED_Init (
            uint8_t ledPort,
            uint8_t ledPin )
```

Initialize the LED PIN.

Calls DIO_Init with specific port and pin with OUT direction

Parameters

ledPort	Port containing the PIN that will be set as output
ledPin	PIN to be set as output

See also

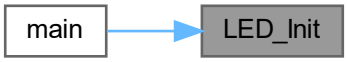
DIO_init()

Returns

Status NOT_OK if initialization failed, return OK otherwise

Here is the call graph for this function:

Here is the caller graph for this function:



3.8.3.2 LED_Off()

```
Status LED_Off (
            uint8_t ledPort,
            uint8_t ledPin )
```

Turns LED Off.

Calls DIO_Write with specific port and pin to set the pin to LOW

Parameters

ledPort	Port containing the PIN that will be set to low
ledPin	PIN to be set as output

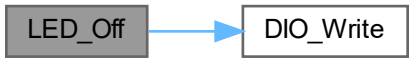
See also

DIO_Write()

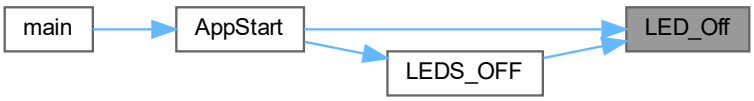
Returns

Status return if turning off the LED fails, return OK otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



3.8.3.3 LED_On()

```
Status LED_On (
            uint8_t ledPort,
            uint8_t ledPin )
```

Turns LED on.

Calls DIO_Write with specific port and pin to set the pin to high

Parameters

ledPort	Port containing the PIN that will be set to HIGH
ledPin	PIN to be set as output

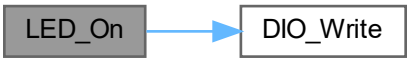
See also

DIO_Write()

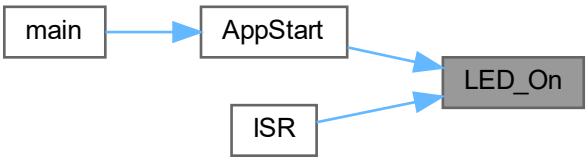
Returns

Status return if turning on the LED fails, return OK otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



3.8.3.4 LED_toggle()

```
Status LED_toggle (
            uint8_t ledPort,
            uint8_t ledPin )
```

Toggle the status of the LED.

Calls DIO_Toggle with specific port and pin to flip the status of the pin

Parameters

ledPort	Port containing the PIN that will be flipped
ledPin	PIN to be set as output

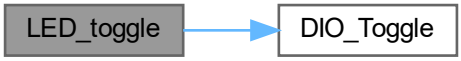
See also

DIO_Toggle()

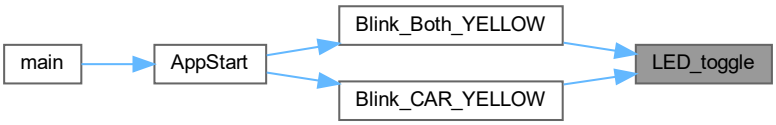
Returns

Status return if toggling the LED fails, return OK otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



3.9 led.h

Go to the documentation of this file.

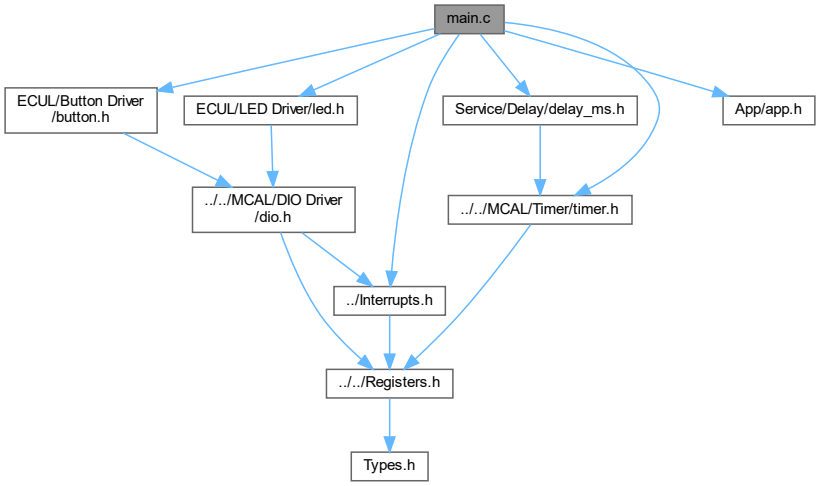
```
1
12 #ifndef LED_H_
13 #define LED_H_
14
15 #include "../../MCAL/DIO Driver/dio.h"
16
17
18 #define CAR_LED_GREEN       PIN0
19 #define CAR_LED_YELLOW      PIN1
20 #define CAR_LED_RED         PIN2
21
22 #define PEDES_LED_GREEN     PIN0
23 #define PEDES_LED_YELLOW    PIN1
24 #define PEDES_LED_RED       PIN2
25
26 Status LED_Init(uint8_t ledPort, uint8_t ledPin);
27 Status LED_On(uint8_t ledPort, uint8_t ledPin);
28 Status LED_Off(uint8_t ledPort, uint8_t ledPin);
29 Status LED_toggle(uint8_t ledPort, uint8_t ledPin);
30
31
32
33
34 #endif /* LED_H_ */
```

3.10 main.c File Reference

the application main function and App_start function

```
#include "ECUL/Button Driver/button.h"
#include "ECUL/LED Driver/led.h"
#include "Service/Delay/delay_ms.h"
#include "MCAL/Timer/timer.h"
#include "MCAL/Interrupts.h"
#include "App/app.h"
```

Include dependency graph for main.c:



Macros

- #define F_CPU 1000000U

  Clock frequency.

Functions

- int main (void)

  Program main function.
- void AppStart ()

  AppStart determine which mode the program is on and in return will detemine the illumination of the car and pedestrian LEDs accordingly.
- Status LEDS_OFF ()
- Status Blink_CAR_YELLOW ()

  blink the car yellow LED for 5 seconds
- Status Blink_Both_YELLOW ()

  blink the both yellow LEDs for 5 seconds
- ISR (EXT_INT_0)

  ISR will shift the state from Normal to Pedestrian Upon button press.

Variables

- Mode currentMode = Normal

  The state of the program.
- LedMode current_carLed_Mode = Green

  The variable holds the currently illuminated LED.
- LedMode previous_carLed_Mode = Green

  The variable holds the previously illuminated LED.
- uint8_t Flag = FALSE

  The flag which determine if the interrupt is fired or not.
- LedMode FirstState

  The variable shows which LED was illuminated when the interrupt was fired.
- Status App_state

  holds the status of the running application

3.10.1 Detailed Description

the application main function and App_start function

Author

Moataz

Date

September 2022

3.10.2 Macro Definition Documentation

3.10.2.1 F_CPU

```
#define F_CPU 1000000U
```

Clock frequency.

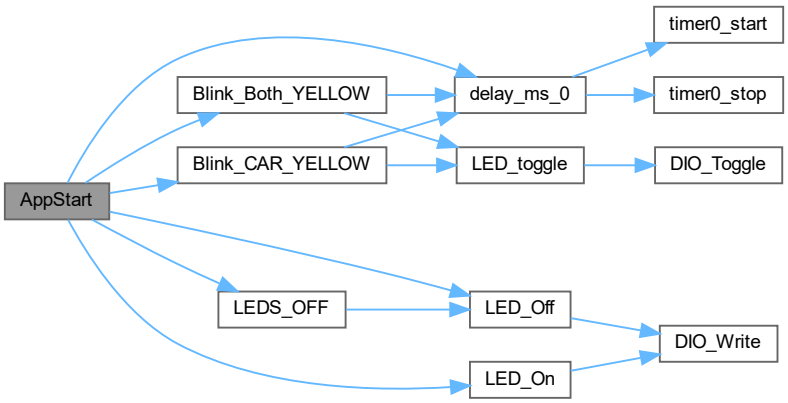
3.10.3 Function Documentation

3.10.3.1 AppStart()

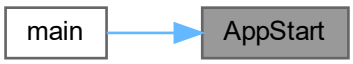
```
void AppStart ( )
```

AppStart determine which mode the program is on and in return will detemine the illumination of the car and pedestrian LEDs accordingly.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.3.2 Blink_Both_YELLOW()

```
Status Blink_Both_YELLOW ( )
```

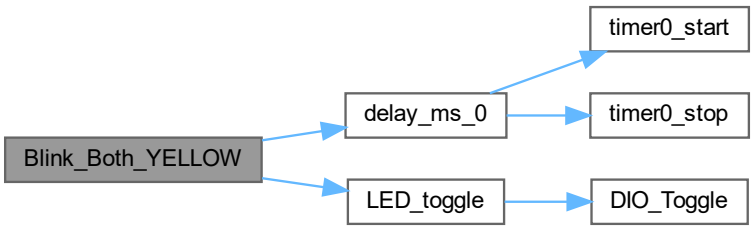
blink the both yellow LEDs for 5 seconds

calls LED toggle passing the car yellow LED parameters then delay 1 ms in a loop of 5 iterations

Returns

Status returns not ok if failure occurs during toggling or delay, otherwise return Ok

Here is the call graph for this function:



Here is the caller graph for this function:

3.10.3.3 Blink_CAR_YELLOW()

```
Status Blink_CAR_YELLOW ( )
```

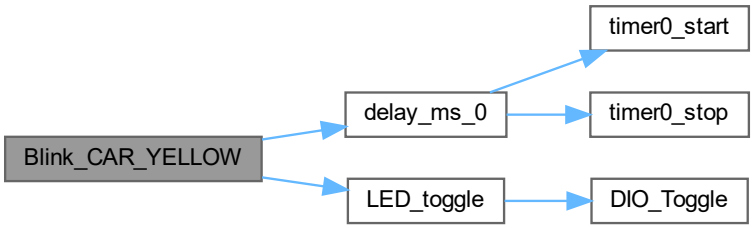
blink the car yellow LED for 5 seconds

calls LED toggle passing the car yellow LED parameters then delay 1 ms in a loop of 5 iterations

Returns

Status returns not ok if failure occurs during toggling or delay, otherwise return Ok

Here is the call graph for this function:



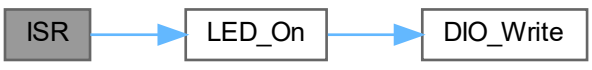
Here is the caller graph for this function:

3.10.3.4 ISR()

```
ISR (
            EXT_INT_0  )
```

ISR will shift the state from Normal to Pedestrian Upon button press.

the ISR will be fired with the button is pressed Here is the call graph for this function:



3.10.3.5 LEDS_OFF()

```
Status LEDS_OFF ( )
```

Returns

Status

Here is the call graph for this function:

Here is the caller graph for this function:

3.10.3.6 main()

```
int main (
            void  )
```

Program main function.

Initializers the needed drivers for the application calls the Appstart to start the application

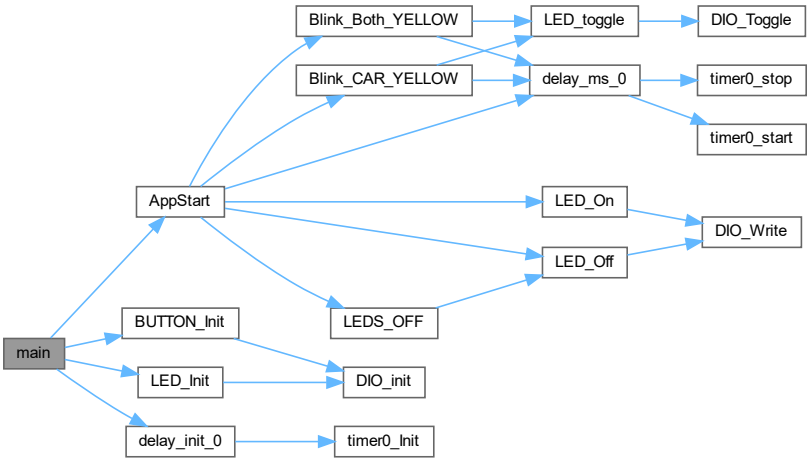
See also

LED_Init() BUTTON_Init() delay_init_0() AppStart()

Returns

int 0 if the programs has no errors,

Here is the call graph for this function:



3.10.4 Variable Documentation

3.10.4.1 App_state

```
Status App_state
```

holds the status of the running application

3.10.4.2 current_carLed_Mode

```
LedMode current_carLed_Mode = Green
```

The variable holds the currently illuminated LED.

3.10.4.3 currentMode

```
Mode currentMode = Normal
```

The state of the program.

3.10.4.4 FirstState

```
LedMode FirstState
```

The variable shows which LED was illuminated when the interrupt was fired.

3.10.4.5 Flag

```
uint8_t Flag = FALSE
```

The flag which determine if the interrupt is fired or not.

3.10.4.6 previous_carLed_Mode

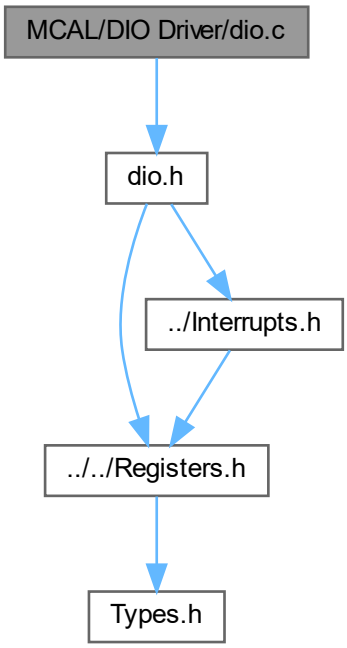
```
LedMode previous_carLed_Mode = Green
```

The variable holds the previously illuminated LED.

3.11 MCAL/DIO Driver/dio.c File Reference

```
#include "dio.h"
```

Include dependency graph for dio.c:



Functions

- Status DIO_init (uint8_t pinNumber, uint8_t portNumber, uint8_t direction)

  Initialize the pin of the port with the required direction.
- Status DIO_Write (uint8_t pinNumber, uint8_t portNumber, uint8_t value)

  write a value in the pin
- Status DIO_Toggle (uint8_t pinNumber, uint8_t portNumber)

  Toggle the value of the output pin.
- Status DIO_Read (uint8_t pinNumber, uint8_t portNumber, uint8_t *value)

  Read the value of a certain PIN.

3.11.1 Detailed Description

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

Copyright (c) 2022

3.11.2 Function Documentation

3.11.2.1 DIO_init()

```
Status DIO_init (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t direction )
```

Initialize the pin of the port with the required direction.

initialize a pin in a certain port with the direction

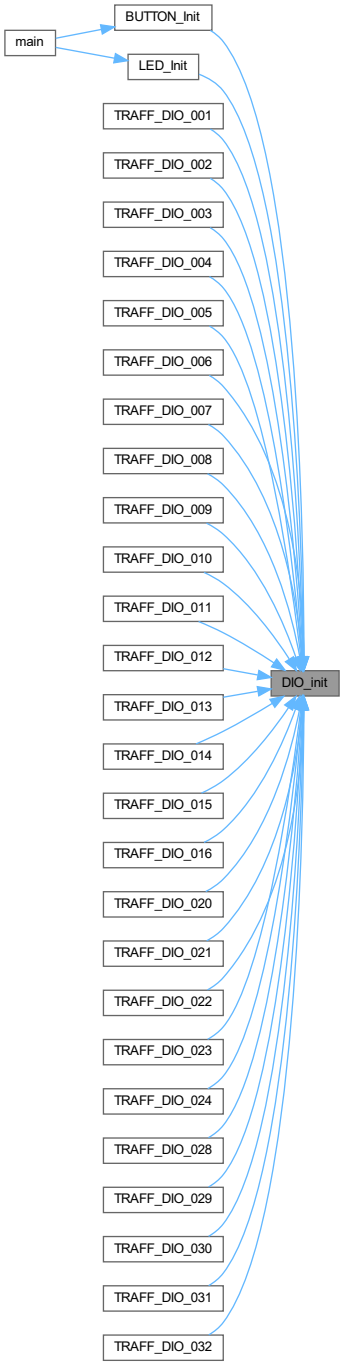
Parameters

pinNumber	PIN that will be set
portNumber	Port containing the PIN that will be set
direction	(IN//OUT)

Returns

Status return Not_Ok if the passed port is not in range (A->D) or pin number > 7

Here is the caller graph for this function:



3.11.2.2 DIO_Read()

```
Status DIO_Read (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t * value )
```

Read the value of a certain PIN.

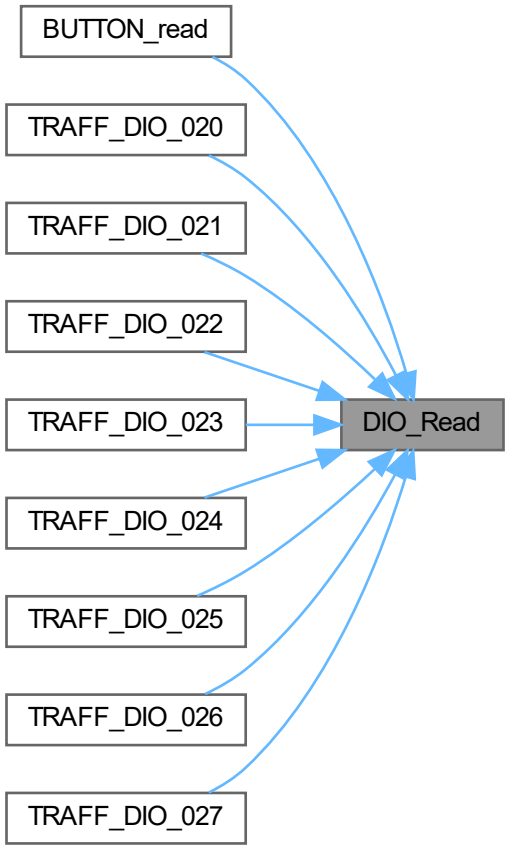
Parameters

pinNumber	PIN that will be read
portNumber	Port containing the PIN that will be read
value	pointer holds the state of the PIN (HIGH or LOW)

Returns

Status return Not_Ok if the passed pin direction is in not set as input or pin number $>$ 7, otherwise return Ok

Here is the caller graph for this function:



3.11.2.3 DIO_Toggle()

```
Status DIO_Toggle (
            uint8_t pinNumber,
            uint8_t portNumber )
```

Toggle the value of the output pin.

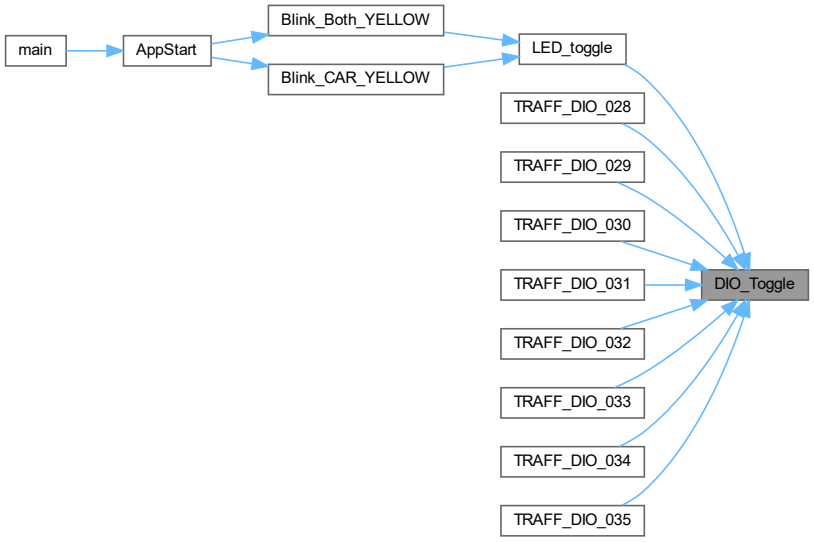
Parameters

pinNumber	PIN that will be toggled
portNumber	Port containing the PIN that will be toggled

Returns

Status return Not_Ok if the passed pin direction is in not set as output or pin number $>$ 7, otherwise return Ok

Here is the caller graph for this function:



3.11.2.4 DIO_Write()

```
Status DIO_Write (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t value )
```

write a value in the pin

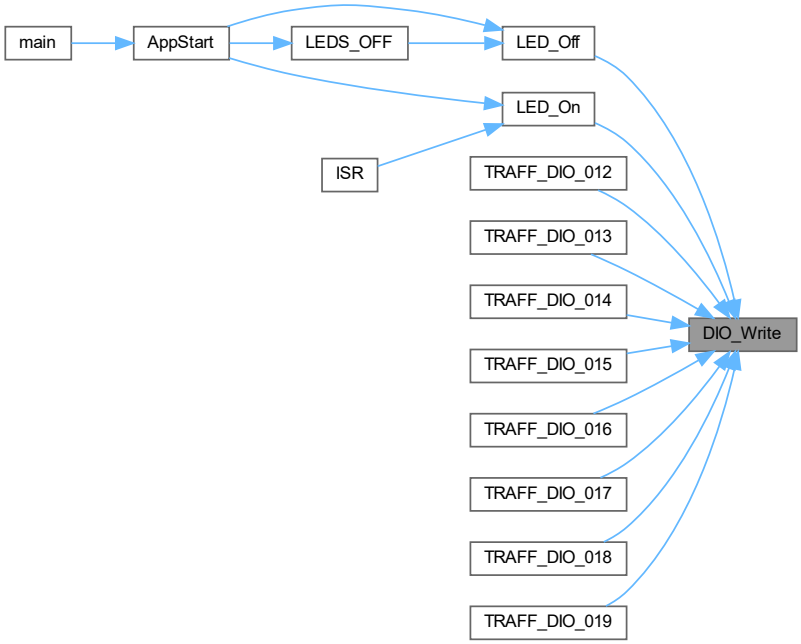
Parameters

pinNumber	PIN that will be set
portNumber	Port containing the PIN that will be set
value	(HIGH/LOW)

Returns

Status return Not_Ok if the passed pin direction is in not set as output or pin number $>$ 7, otherwise return Ok

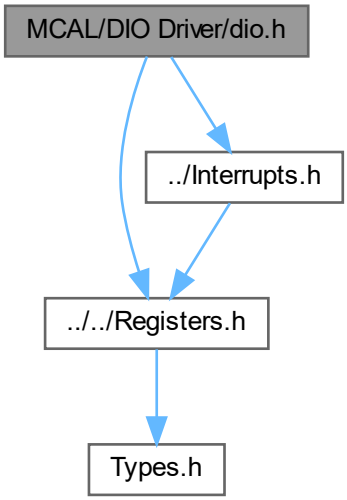
Here is the caller graph for this function:



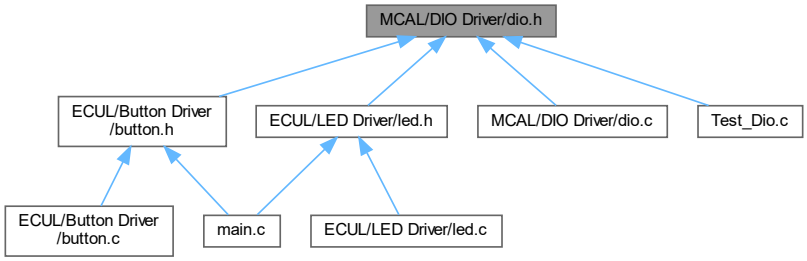
3.12 MCAL/DIO Driver/dio.h File Reference

```
#include "../../Registers.h"
#include "../Interrupts.h"
```

Include dependency graph for dio.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define Port_A 'A'
- #define Port_B 'B'
- #define Port_C 'C'
- #define Port_D 'D'
- #define IN 0
- #define OUT 1
- #define LOW 0
- #define HIGH 1

Functions

- Status DIO_init (uint8_t pinNumber, uint8_t portNumber, uint8_t direction)

  Initialize the pin of the port with the required direction.
- Status DIO_Write (uint8_t pinNumber, uint8_t portNumber, uint8_t value)

  write a value in the pin
- Status DIO_Toggle (uint8_t pinNumber, uint8_t portNumber)

  Toggle the value of the output pin.
- Status DIO_Read (uint8_t pinNumber, uint8_t portNumber, uint8_t *value)

  Read the value of a certain PIN.

3.12.1 Macro Definition Documentation

3.12.1.1 HIGH

```
#define HIGH 1
```

3.12.1.2 IN

```
#define IN 0
```

3.12.1.3 LOW

```
#define LOW 0
```

3.12.1.4 OUT

```
#define OUT 1
```

3.12.1.5 Port_A

```
#define Port_A 'A'
```

3.12.1.6 Port_B

```
#define Port_B 'B'
```

3.12.1.7 Port_C

```
#define Port_C 'C'
```

3.12.1.8 Port_D

```
#define Port_D 'D'
```

3.12.2 Function Documentation

3.12.2.1 DIO_init()

```
Status DIO_init (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t direction )
```

Initialize the pin of the port with the required direction.

initialize a pin in a certain port with the direction

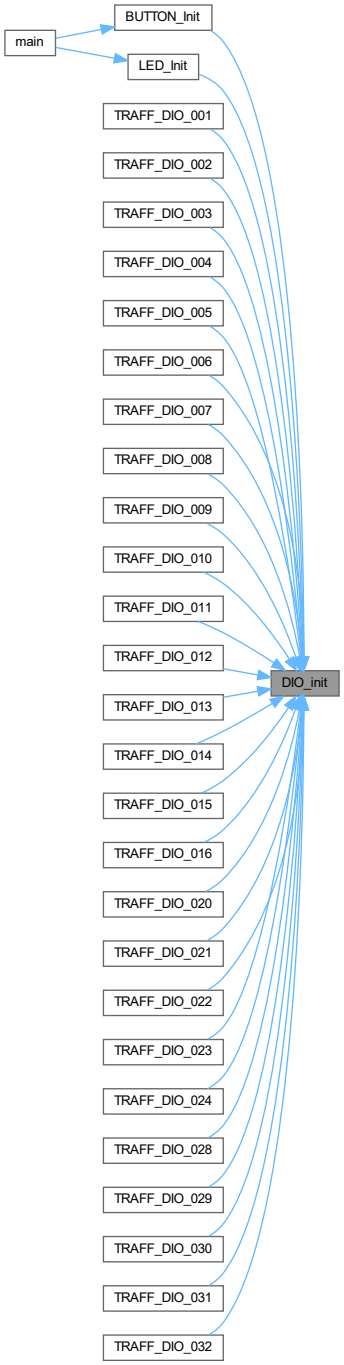
Parameters

pinNumber	PIN that will be set
portNumber	Port containing the PIN that will be set
direction	(IN//OUT)

Returns

Status return Not_Ok if the passed port is not in range (A->D) or pin number > 7

Here is the caller graph for this function:



3.12.2.2 DIO_Read()

```
Status DIO_Read (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t * value )
```

Read the value of a certain PIN.

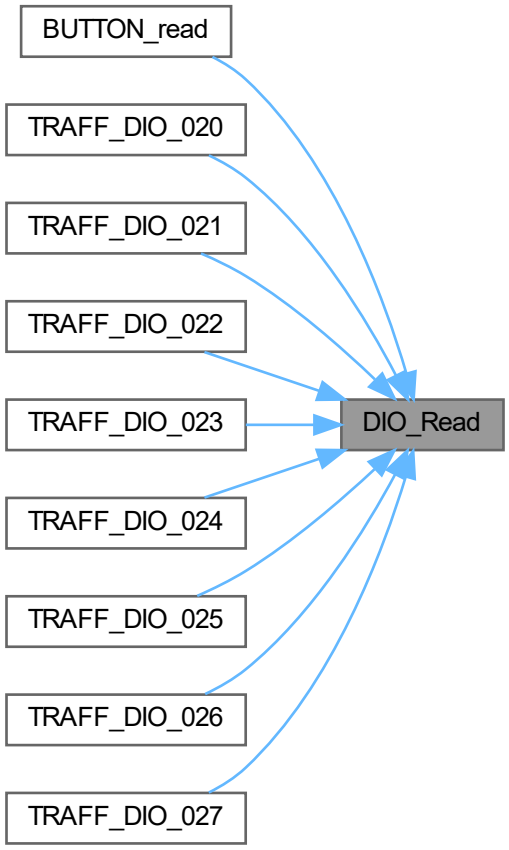
Parameters

pinNumber	PIN that will be read
portNumber	Port containing the PIN that will be read
value	pointer holds the state of the PIN (HIGH or LOW)

Returns

Status return Not_Ok if the passed pin direction is in not set as input or pin number $>$ 7, otherwise return Ok

Here is the caller graph for this function:



3.12.2.3 DIO_Toggle()

```
Status DIO_Toggle (
            uint8_t pinNumber,
            uint8_t portNumber )
```

Toggle the value of the output pin.

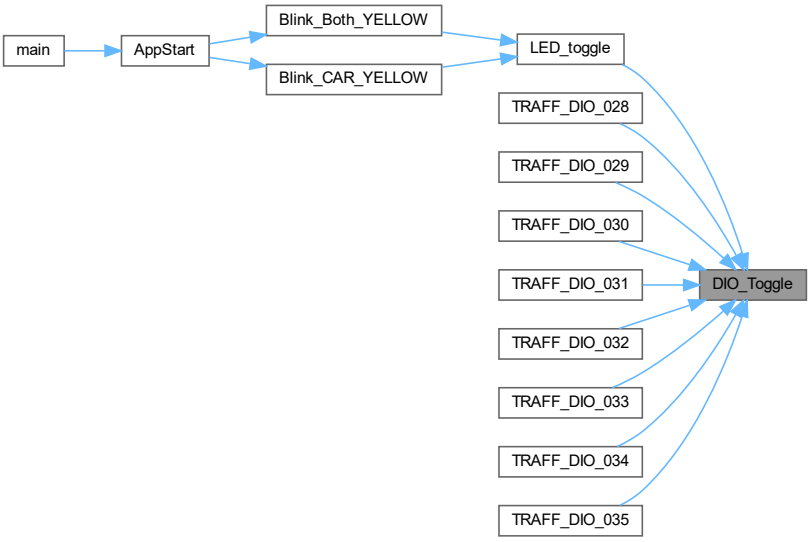
Parameters

pinNumber	PIN that will be toggled
portNumber	Port containing the PIN that will be toggled

Returns

Status return Not_Ok if the passed pin direction is in not set as output or pin number $>$ 7, otherwise return Ok

Here is the caller graph for this function:



3.12.2.4 DIO_Write()

```
Status DIO_Write (
            uint8_t pinNumber,
            uint8_t portNumber,
            uint8_t value )
```

write a value in the pin

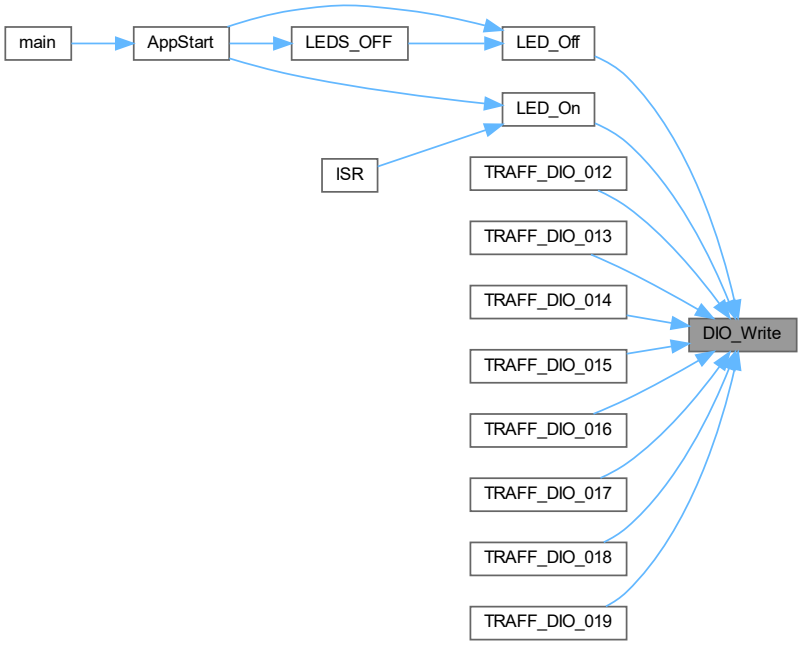
Parameters

pinNumber	PIN that will be set
portNumber	Port containing the PIN that will be set
value	(HIGH/LOW)

Returns

Status return Not_Ok if the passed pin direction is in not set as output or pin number $>$ 7, otherwise return Ok

Here is the caller graph for this function:



3.13 dio.h

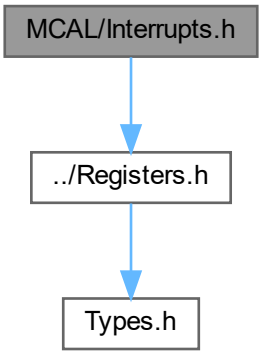
Go to the documentation of this file.

```
1 /*
2 * dio.h
3 *
4 * Created:  9/6/2022 10:36:59 PM
5 *  Author:  Moataz
6 */
7
8
9 #ifndef DIO_H_
10 #define DIO_H_
11
12 #include "../../Registers.h"
13 #include "../Interrupts.h"
14
15 //all internal driver typedefs
16 //all driver macros
17
18 #define Port_A 'A'
```

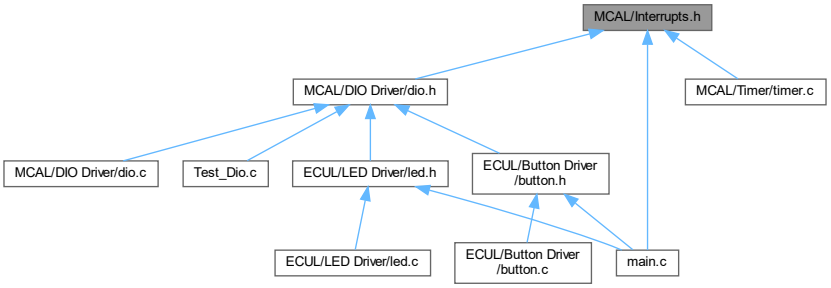
```
19 #define Port_B 'B'
20 #define Port_C 'C'
21 #define Port_D 'D'
22
23 //Pin Direction
24 #define IN 0
25 #define OUT 1
26
27 //PIN Value
28 #define LOW 0
29 #define HIGH 1
30
31 //all driver function prototypes
32
33 Status DIO_init(uint8_t pinNumber, uint8_t portNumber, uint8_t direction);
34
35 Status DIO_Write(uint8_t pinNumber, uint8_t portNumber, uint8_t value);
36 Status DIO_Toggle(uint8_t pinNumber, uint8_t portNumber);
37 Status DIO_Read(uint8_t pinNumber, uint8_t portNumber, uint8_t* value);
38
39
40
41 #endif /* DIO_H_ */
```

3.14 MCAL/Interrupts.h File Reference

`#include "../Registers.h"`
Include dependency graph for Interrupts.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define TIMER0_OVF __vector_12
- #define EXT_INT_0 __vector_1
- #define EXT_INT_1 __vector_2
- #define EXT_INT_2 __vector_3
- #define sei() __asm__ __volatile__("sei" ::: "memory")

  Enable the interrupt in the status register.
- #define cli() __asm__ __volatile__("cli" ::: "memory")

  clear the interrupt in the status register
- #define ISR(INT_VECT)

  Interrupt definition.

3.14.1 Macro Definition Documentation

3.14.1.1 cli

```
#define cli( ) __asm__ __volatile__("cli" :::  "memory")
```

clear the interrupt in the status register

3.14.1.2 EXT_INT_0

```
#define EXT_INT_0 __vector_1
```

3.14.1.3 EXT_INT_1

```
#define EXT_INT_1 __vector_2
```

3.14.1.4 EXT_INT_2

```
#define EXT_INT_2 __vector_3
```

3.14.1.5 ISR

```
#define ISR(
              INT_VECT )
```

Value:

```
void INT_VECT(void)  __attribute__((signal,used));\
void INT_VECT(void)
```

Interrupt definition.

3.14.1.6 sei

```
#define sei( ) __asm__ __volatile__("sei" :::  "memory")
```

Enable the interrupt in the status register.

3.14.1.7 TIMER0_OVF

```
#define TIMER0_OVF __vector_12
```

3.15 Interrupts.h

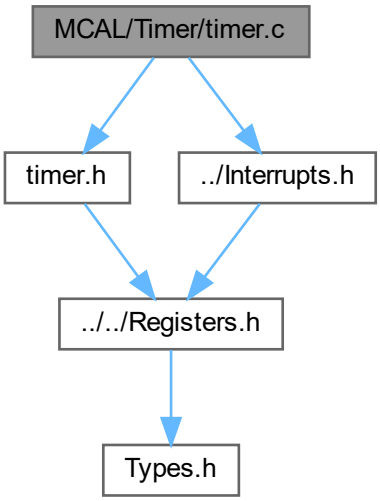
Go to the documentation of this file.

```
1 /*
2 * Interrupts.h
3 *
4 * Created:  9/9/2022 12:34:52 AM
5 *  Author:  Moataz
6 */
7
8
9
10 #ifndef INTERRUPTS_H_
11 #define INTERRUPTS_H_
12
13 #include "../Registers.h"
14
15 #define TIMER0_OVF  __vector_12
16 #define EXT_INT_0   __vector_1
17 #define EXT_INT_1   __vector_2
18 #define EXT_INT_2   __vector_3
19
21 #define sei() __asm__ __volatile__("sei" :::  "memory")
22
24 #define cli() __asm__ __volatile__("cli" :::  "memory")
25
27 #define ISR(INT_VECT)void INT_VECT(void)  __attribute__((signal,used));\
28 void INT_VECT(void)
29
30
31
32 #endif /* INTERRUPTS_H_ */
33
```

3.16 MCAL/Timer/timer.c File Reference

```
#include "timer.h"
#include "../Interrupts.h"
```

Include dependency graph for timer.c:



Functions

- Status timer0_Init (Timer_Mode mode, COM_Mode comMode)
  Initialize the timer 0.
- Status timer0_start (uint8_t CmpValue, clk_source_T0 prescalor)
  Start the timer 0.
- Status timer0_stop ()
  stop the timer 0

3.16.1 Detailed Description

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

3.16.2 Function Documentation

3.16.2.1 timer0_Init()

```
Status timer0_Init (
            Timer_Mode mode,
            COM_Mode comMode )
```

Initialize the timer 0.

Initialize the TCCR0 set the Compare Match Output Mode set the WGM01 and WGM00

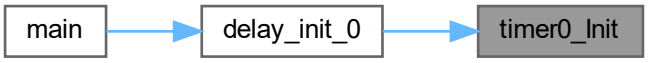
Parameters

mode	determine which mode will be used for counting and overflow of the interrupt
comMode	determine the compare match output mode

Returns

Status Ok for no error occurs, Not_Ok for error during intialization

Here is the caller graph for this function:



3.16.2.2 timer0_start()

```
Status timer0_start (
            uint8_t CmpValue,
            clk_source_T0 prescalor )
```

Start the timer 0.

set the OCR0 with the cmp value set the prescalor in TCCR0

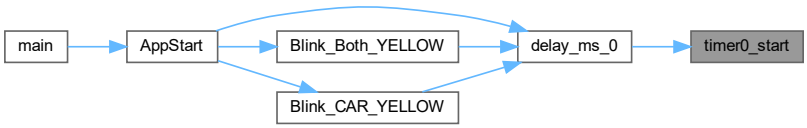
Parameters

CmpValue	the compare value at with the reset will occur
prescalor	the value of the prescalor (in the project it prescalor 64 will be used)

Returns

Status returns ok after it starts the timer.

Here is the caller graph for this function:



3.16.2.3 timer0_stop()

```
Status timer0_stop ( )
```

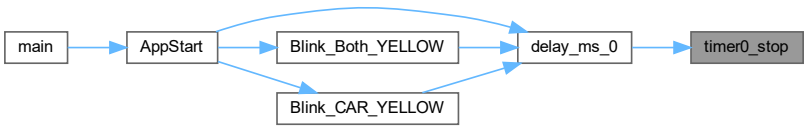
stop the timer 0

Clear TCCR0 and OCR0

Returns

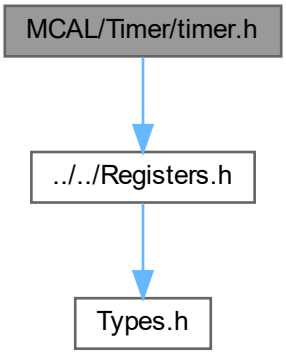
Status returns ok after it stops the timer.

Here is the caller graph for this function:

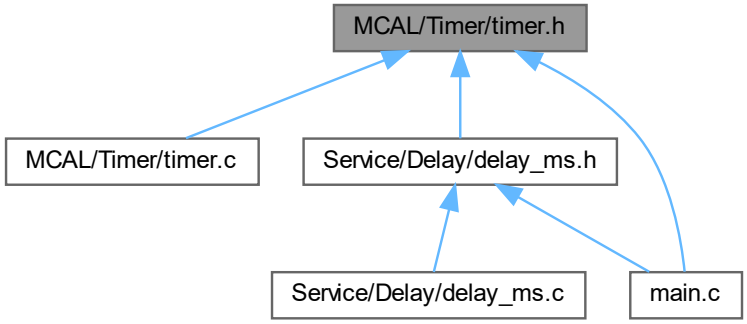


3.17 MCAL/Timer/timer.h File Reference

`#include "../../Registers.h"`
Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define WGM01 PIN3
- #define WGM00 PIN6

Typedefs

- typedef enum Timer_Mode Timer_Mode
  timer mode enum
- typedef enum COM_Mode COM_Mode
  Compare Match Output Mode.
- typedef enum clk_source_T0 clk_source_T0
  prescalor enum for timer 0

Enumerations

- enum Timer_Mode { Timer_Normal = 0 , Phase_Correct , CTC , Fast_PWM }

  timer mode enum
- enum COM_Mode { NO_OUTPUT =0 , Toggle , Clear , Set }

  Compare Match Output Mode.
- enum clk_source_T0 {
  T0_No_Clk_Source =0 , T0_No_prescalor , T0_Precalor_8 , T0_Precalor_64 ,
  T0_Precalor_256 , T0_Precalor_1024 , T0_Ext_falling , T0_Ext_Rising }

  prescalor enum for timer 0

Functions

- Status timer0_Init (Timer_Mode mode, COM_Mode comMode)

  Initialize the timer 0.
- Status timer0_start (uint8_t CmpValue, clk_source_T0 prescalor)

  Start the timer 0.
- Status timer0_stop ()

  stop the timer 0

3.17.1 Macro Definition Documentation

3.17.1.1 WGM00

```
#define WGM00 PIN6
```

3.17.1.2 WGM01

```
#define WGM01 PIN3
```

3.17.2 Typedef Documentation

3.17.2.1 clk_source_T0

```
typedef enum clk_source_T0 clk_source_T0
```

prescalor enum for timer 0

3.17.2.2 COM_Mode

```
typedef enum COM_Mode COM_Mode
```

Compare Match Output Mode.

3.17.2.3 Timer_Mode

```
typedef enum Timer_Mode Timer_Mode
```

timer mode enum

3.17.3 Enumeration Type Documentation

3.17.3.1 clk_source_T0

```
enum clk_source_T0
```

prescalor enum for timer 0

Enumerator

T0_No_Clk_Source	
T0_No_prescalor	
T0_Precalor_8	
T0_Precalor_64	
T0_Precalor_256	
T0_Precalor_1024	
T0_Ext_falling	
T0_Ext_Rising	

3.17.3.2 COM_Mode

```
enum COM_Mode
```

Compare Match Output Mode.

Enumerator

NO_OUTPUT	
Toggle	
Clear	
Set	

3.17.3.3 Timer_Mode

```
enum Timer_Mode
```

timer mode enum

Enumerator

Timer_Normal	
Phase_Correct	
CTC	
Fast_PWM	

3.17.4 Function Documentation

3.17.4.1 timer0_Init()

```
Status timer0_Init (
            Timer_Mode mode,
            COM_Mode comMode )
```

Initialize the timer 0.

Initialize the TCCR0 set the Compare Match Output Mode set the WGM01 and WGM00

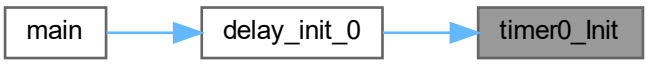
Parameters

mode	determine which mode will be used for counting and overflow of the interrupt
comMode	determine the compare match output mode

Returns

Status Ok for no error occurs, Not_Ok for error during intialization

Here is the caller graph for this function:



3.17.4.2 timer0_start()

```
Status timer0_start (
            uint8_t CmpValue,
            clk_source_T0 prescalor )
```

Start the timer 0.

set the OCR0 with the cmp value set the prescalor in TCCR0

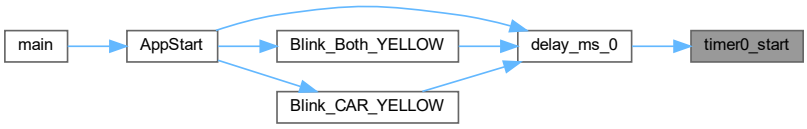
Parameters

CmpValue	the compare value at with the reset will occur
prescalor	the value of the prescalor (in the project it prescalor 64 will be used)

Returns

Status returns ok after it starts the timer.

Here is the caller graph for this function:



3.17.4.3 timer0_stop()

```
Status timer0_stop ( )
```

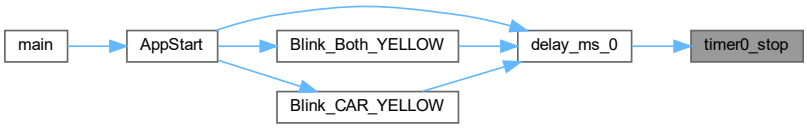
stop the timer 0

Clear TCCR0 and OCR0

Returns

Status returns ok after it stops the timer.

Here is the caller graph for this function:



3.18 timer.h

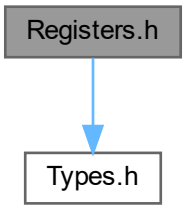
Go to the documentation of this file.

```
1 /*
2 * timer1.h
3 *
4 * Created:  9/8/2022 9:41:41 PM
5 *  Author:  Moataz
6 */
7
8
9 #ifndef TIMER0_H_
10 #define TIMER0_H_
11
12
13 #include "../../Registers.h"
14
15
16 #define WGM01 PIN3
17 #define WGM00 PIN6
18
20 typedef enum Timer_Mode{
21     Timer_Normal = 0,
22     Phase_Correct,
23     CTC,
24     Fast_PWM
25     }Timer_Mode;
26
28 typedef enum COM_Mode{
29     NO_OUTPUT =0,
30     Toggle,//only applicable in CTC mode
31     Clear,
32     Set
33     }COM_Mode;
34
36 typedef enum clk_source_T0 {
37     T0_No_Clk_Source =0,
38     T0_No_prescalor,
39     T0_Precalor_8,
40     T0_Precalor_64,
41     T0_Precalor_256,
42     T0_Precalor_1024,
43     T0_Ext_falling,
44     T0_Ext_Rising
45     }clk_source_T0;
46
47
48
49 Status timer0_Init(Timer_Mode mode,COM_Mode comMode);
50 Status timer0_start(uint8_t CmpValue,clk_source_T0 prescalor);
51 Status timer0_stop();
52
53
54
55
56 #endif /* TIMER1_H_ */
```

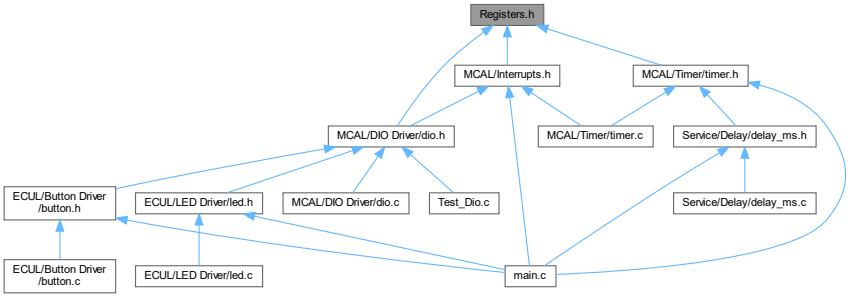
3.19 Registers.h File Reference

```
#include "Types.h"
```

Include dependency graph for Registers.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define PORTA (∗(volatile uint8_t∗) 0x3B)
- #define DDRA (∗(volatile uint8_t∗)0x3A)
- #define PINA (∗(volatile uint8_t∗) 0x39)
- #define PORTB (∗(volatile uint8_t∗) 0x38)
- #define DDRB (∗(volatile uint8_t∗) 0x37)
- #define PINB (∗(volatile uint8_t∗) 0x36)
- #define PORTC (∗(volatile uint8_t∗) 0x35)
- #define DDRC (∗(volatile uint8_t∗)0x34)
- #define PINC (∗(volatile uint8_t∗) 0x33)
- #define PORTD (∗(volatile uint8_t∗) 0x32)
- #define DDRD (∗(volatile uint8_t∗) 0x31)
- #define PIND (∗(volatile uint8_t∗) 0x30)
- #define TCCR0 (∗(volatile uint8_t∗) 0x53)
- #define TCNT0 (∗(volatile uint8_t∗) 0x52)
- #define OCR0 (∗(volatile uint8_t∗) 0x5C)
- #define TCCR2 (∗(volatile uint8_t∗) 0x45)

- #define TCNT2 (∗(volatile uint8_t∗) 0x44)
- #define OCR2 (∗(volatile uint8_t∗) 0x43)
- #define TIFR (∗(volatile uint8_t∗) 0x58)
- #define SREG (∗(volatile uint8_t∗) 0x5F)
- #define MCUCR (∗(volatile uint8_t∗) 0x55)
- #define MCUCSR (∗(volatile uint8_t∗) 0x54)
- #define GICR (∗(volatile uint8_t∗) 0x5B)
- #define GIFR (∗(volatile uint8_t∗) 0x5A)
- #define ISC00 0
- #define ISC01 1
- #define INT0 6
- #define PIN0 0
- #define PIN1 1
- #define PIN2 2
- #define PIN3 3
- #define PIN4 4
- #define PIN5 5
- #define PIN6 6
- #define PIN7 7
- #define bitset(byte, nbit) ((byte) |= (1<<(nbit)))
- #define bitclear(byte, nbit) ((byte) &= ∼(1<<(nbit)))
- #define bitflip(byte, nbit) ((byte) ^= (1<<(nbit)))
- #define bitRead(byte, nbit) (((byte) & (1<<(nbit))) >> nbit)

3.19.1 Macro Definition Documentation

3.19.1.1 bitclear

```
#define bitclear(
              byte,
              nbit ) ((byte) &= ∼(1<<(nbit)))
```

3.19.1.2 bitflip

```
#define bitflip(
              byte,
              nbit ) ((byte) ^= (1<<(nbit)))
```

3.19.1.3 bitRead

```
#define bitRead(
              byte,
              nbit ) (((byte) & (1<<(nbit))) >> nbit)
```

3.19.1.4 bitset

```
#define bitset(
              byte,
              nbit ) ((byte) |= (1<<(nbit)))
```

3.19.1.5 DDRA

```
#define DDRA (*(volatile uint8_t*)0x3A)
```

3.19.1.6 DDRB

```
#define DDRB (*(volatile uint8_t*) 0x37)
```

3.19.1.7 DDRC

```
#define DDRC (*(volatile uint8_t*)0x34)
```

3.19.1.8 DDRD

```
#define DDRD (*(volatile uint8_t*) 0x31)
```

3.19.1.9 GICR

```
#define GICR (*(volatile uint8_t*) 0x5B)
```

3.19.1.10 GIFR

```
#define GIFR (*(volatile uint8_t*) 0x5A)
```

3.19.1.11 INT0

```
#define INT0 6
```

3.19.1.12 ISC00

```
#define ISC00 0
```

3.19.1.13 ISC01

```
#define ISC01 1
```

3.19.1.14 MCUCR

```
#define MCUCR (*(volatile uint8_t*) 0x55)
```

3.19.1.15 MCUCSR

```
#define MCUCSR (*(volatile uint8_t*) 0x54)
```

3.19.1.16 OCR0

```
#define OCR0 (*(volatile uint8_t*) 0x5C)
```

3.19.1.17 OCR2

```
#define OCR2 (*(volatile uint8_t*) 0x43)
```

3.19.1.18 PIN0

```
#define PIN0 0
```

3.19.1.19 PIN1

```
#define PIN1 1
```

3.19.1.20 PIN2

```
#define PIN2 2
```

3.19.1.21 PIN3

```
#define PIN3 3
```

3.19.1.22 PIN4

```
#define PIN4 4
```

3.19.1.23 PIN5

```
#define PIN5 5
```

3.19.1.24 PIN6

```
#define PIN6 6
```

3.19.1.25 PIN7

```
#define PIN7 7
```

3.19.1.26 PINA

```
#define PINA (*(volatile uint8_t*) 0x39)
```

3.19.1.27 PINB

```
#define PINB (*(volatile uint8_t*) 0x36)
```

3.19.1.28 PINC

```
#define PINC (*(volatile uint8_t*) 0x33)
```

3.19.1.29 PIND

```
#define PIND (*(volatile uint8_t*) 0x30)
```

3.19.1.30 PORTA

```
#define PORTA (*(volatile uint8_t*) 0x3B)
```

3.19.1.31 PORTB

```
#define PORTB (*(volatile uint8_t*) 0x38)
```

3.19.1.32 PORTC

```
#define PORTC (*(volatile uint8_t*) 0x35)
```

3.19.1.33 PORTD

```
#define PORTD (*(volatile uint8_t*) 0x32)
```

3.19.1.34 SREG

```
#define SREG (*(volatile uint8_t*) 0x5F)
```

3.19.1.35 TCCR0

```
#define TCCR0 (*(volatile uint8_t*) 0x53)
```

3.19.1.36 TCCR2

```
#define TCCR2 (*(volatile uint8_t*) 0x45)
```

3.19.1.37 TCNT0

```
#define TCNT0 (*(volatile uint8_t*) 0x52)
```

3.19.1.38 TCNT2

```
#define TCNT2 (*(volatile uint8_t*) 0x44)
```

3.19.1.39 TIFR

```
#define TIFR (*(volatile uint8_t*) 0x58)
```

3.20 Registers.h

Go to the documentation of this file.

```
1 /*
2 * Registers.h
3 *
4 * Created:  9/6/2022 9:50:40 PM
5 *  Author:  Moataz
6 */
7 /*****************************************************************/
8 /* All MicroController Registers for ATMEGA32A
9 */
10 /*****************************************************************/
11
12
13
14 #ifndef REGISTERS_H_
15 #define REGISTERS_H_
16
17 //include
18 #include "Types.h"
19
20
21 //Definitions
22 /*****************************************************************/
23 /* DIO Registers for ATMEGA32A
24 */
25 /*****************************************************************/
26
```

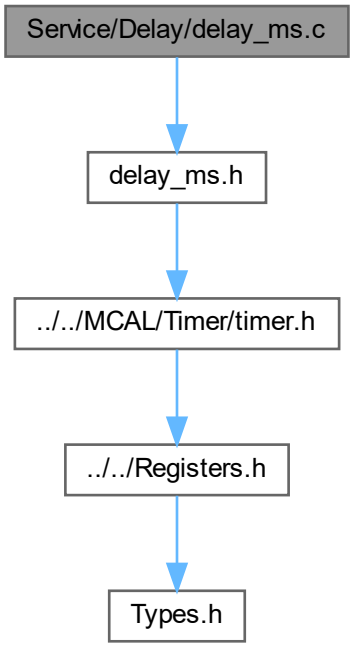
```
27 // PORTA register
28 #define PORTA (*(volatile uint8_t*) 0x3B)
29 #define DDRA (*(volatile uint8_t*)0x3A)
30 #define PINA (*(volatile uint8_t*) 0x39)
31
32 // PORTB register
33 #define PORTB (*(volatile uint8_t*) 0x38)
34 #define DDRB (*(volatile uint8_t*) 0x37)
35 #define PINB (*(volatile uint8_t*) 0x36)
36
37 // PORTC register
38 #define PORTC (*(volatile uint8_t*) 0x35)
39 #define DDRC (*(volatile uint8_t*)0x34)
40 #define PINC (*(volatile uint8_t*) 0x33)
41
42 // PORTD register
43 #define PORTD (*(volatile uint8_t*) 0x32)
44 #define DDRD (*(volatile uint8_t*) 0x31)
45 #define PIND (*(volatile uint8_t*) 0x30)
46
47 /*****************************************************************/
48 /* Timer Registers for ATMEGA32A
49 */
50 /*****************************************************************/
51
52 //Timer 0 registers
53 #define TCCR0 (*(volatile uint8_t*) 0x53)
54 #define TCNT0 (*(volatile uint8_t*) 0x52)
55 #define OCR0 (*(volatile uint8_t*) 0x5C)
56
57 // Timer 2 registers
58 #define TCCR2 (*(volatile uint8_t*) 0x45)
59 #define TCNT2 (*(volatile uint8_t*) 0x44)
60 #define OCR2 (*(volatile uint8_t*) 0x43)
61
62
63 #define TIFR (*(volatile uint8_t*) 0x58)
64 /*****************************************************************/
65 /* Status Register for ATMEGA32A
66 */
67 /*****************************************************************/
68
69 #define SREG (*(volatile uint8_t*) 0x5F)
70
71 /*****************************************************************/
72 /* External Interrupt Registers for ATMEGA32A
73 */
74 /*****************************************************************/
75 #define MCUCR (*(volatile uint8_t*) 0x55)
76 #define MCUCSR (*(volatile uint8_t*) 0x54)
77 #define GICR (*(volatile uint8_t*) 0x5B)
78 #define GIFR (*(volatile uint8_t*) 0x5A)
79
80 #define ISC00   0
81 #define ISC01   1
82 #define INT0    6
83
84 /*****************************************************************/
85 /* Definition of the bits of the registers
86 */
87 /*****************************************************************/
88
89 #define PIN0 0
90 #define PIN1 1
91 #define PIN2 2
92 #define PIN3 3
93 #define PIN4 4
94 #define PIN5 5
95 #define PIN6 6
96 #define PIN7 7
97
98 //function like macros for register and bit accessing
99 #define bitset(byte,nbit)   ((byte) |=  (1<<(nbit)))
100 #define bitclear(byte,nbit) ((byte) &= ~(1<<(nbit)))
101 #define bitflip(byte,nbit)  ((byte) ^=  (1<<(nbit)))
102 #define bitRead(byte,nbit) (((byte) &   (1<<(nbit))) >> nbit)
103
104
105 #endif /* REGISTERS_H_ */
```

3.21 Service/Delay/delay_ms.c File Reference

Implements Delays to be used in the program using the Timer driver.

```
#include "delay_ms.h"
```

Include dependency graph for delay_ms.c:



Functions

- Status delay_init_0 (Timer_Mode mode, COM_Mode comMode)

  Initialize timer 0 with the provided Timer mode and compare match output mode.
- Status delay_ms_0 (volatile uint16_t delay_ms)

  perform a delay with the required time passed in delay_ms

3.21.1 Detailed Description

Implements Delays to be used in the program using the Timer driver.

Author

Moataz Khaled

Version

0.1

Date

2022-09-12

Copyright

Copyright (c) 2022

3.21.2 Function Documentation

3.21.2.1 delay_init_0()

```
Status delay_init_0 (
            Timer_Mode mode,
            COM_Mode comMode )
```

Initialize timer 0 with the provided Timer mode and compare match output mode.

calls the timer0_Init() to initialize the timer with the provided Timer mode and compare match output mode

Parameters

mode	determine which mode will be used for counting and overflow of the interrupt
comMode	determine the compare match output mode

See also

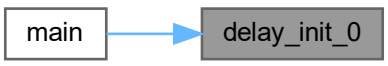
timer0_Init()

Returns

Status Ok for no error occurs, Not_Ok for error during intialization

Here is the call graph for this function:

Here is the caller graph for this function:



3.21.2.2 delay_ms_0()

```
Status delay_ms_0 (
            volatile uint16_t delay_ms )
```

perform a delay with the required time passed in delay_ms

according to the presecalor used, the function will calculate the number of overflows and the OCR and performs a loop the timer 0 reach the OCR the exact number of overflows

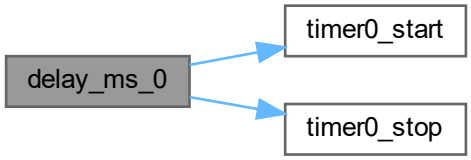
Parameters

delay_ms	The amount of delay in milli second

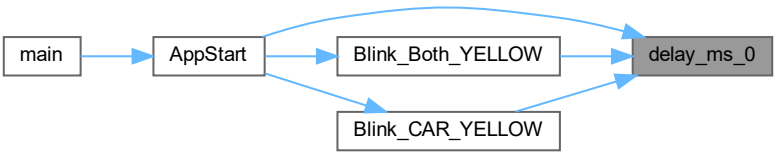
Returns

Status Not_Ok if the component failed to perform the delay otherwise will return OK.

Here is the call graph for this function:



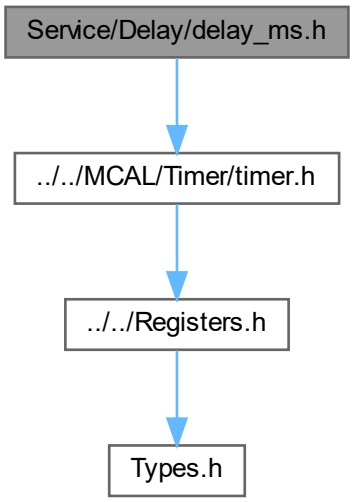
Here is the caller graph for this function:



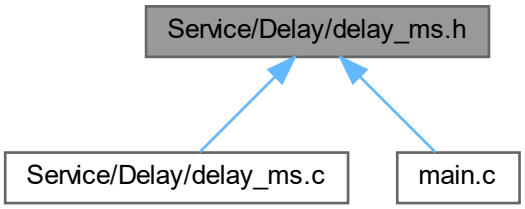
3.22 Service/Delay/delay_ms.h File Reference

```
#include "../../MCAL/Timer/timer.h"
```

Include dependency graph for delay_ms.h:



This graph shows which files directly or indirectly include this file:



Functions

- Status delay_init_0 (Timer_Mode mode, COM_Mode comMode)

  Initialize timer 0 with the provided Timer mode and compare match output mode.
- Status delay_ms_0 (uint16_t delay_ms)

  perform a delay with the required time passed in delay_ms
- Status Stop_delay ()

3.22.1 Function Documentation

3.22.1.1 delay_init_0()

```
Status delay_init_0 (
            Timer_Mode mode,
            COM_Mode comMode )
```

Initialize timer 0 with the provided Timer mode and compare match output mode.

calls the timer0_Init() to initialize the timer with the provided Timer mode and compare match output mode

Parameters

mode	determine which mode will be used for counting and overflow of the interrupt
comMode	determine the compare match output mode

See also

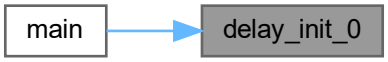
timer0_Init()

Returns

Status Ok for no error occurs, Not_Ok for error during intialization

Here is the call graph for this function:

Here is the caller graph for this function:



3.22.1.2 delay_ms_0()

```
Status delay_ms_0 (
            volatile uint16_t delay_ms )
```

perform a delay with the required time passed in delay_ms

according to the presecalor used, the function will calculate the number of overflows and the OCR and performs a loop the timer 0 reach the OCR the exact number of overflows

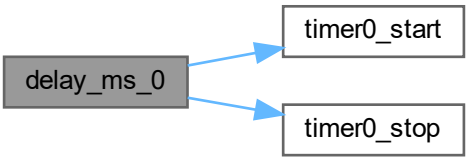
Parameters

delay_ms	The amount of delay in milli second

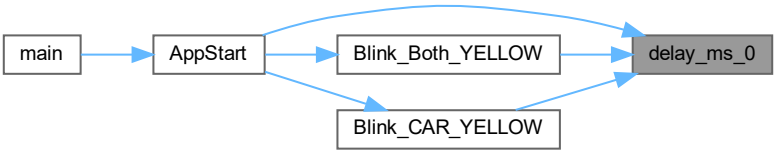
Returns

Status Not_Ok if the component failed to perform the delay otherwise will return OK.

Here is the call graph for this function:



Here is the caller graph for this function:



3.22.1.3 Stop_delay()

```
Status Stop_delay ( )
```

3.23 delay_ms.h

Go to the documentation of this file.

```
1 /*
2 * delay.h
3 *
4 * Created:  9/10/2022 10:30:37 PM
5 *  Author:  Moataz
6 */
```

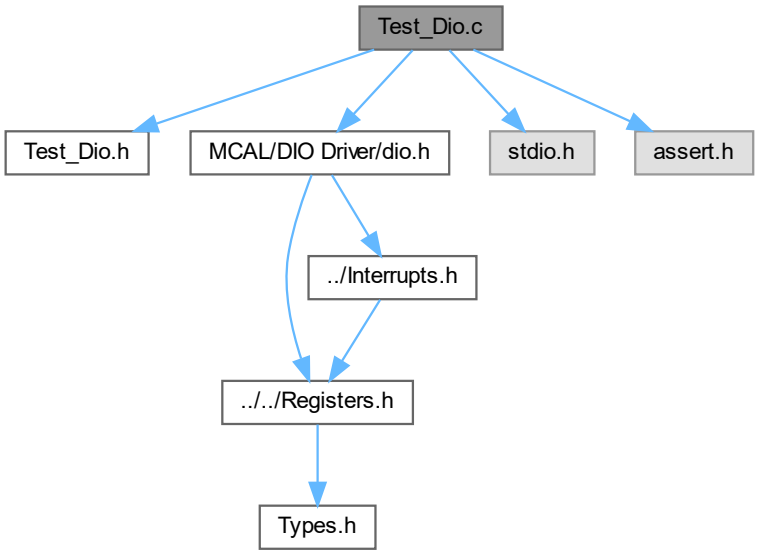
```
7
8
9 #ifndef DELAY_H_
10 #define DELAY_H_
11
12
13 //includes
14 #include "../../MCAL/Timer/timer.h"
15
16 //Definitions
17
18 //Functions prototype
19 Status delay_init_0(Timer_Mode mode,COM_Mode comMode);
20
21
22
23 Status delay_ms_0(uint16_t delay_ms);
24
25
26 Status Stop_delay();
27
28 #endif /* DELAY_H_ */
```

3.24 Test_Dio.c File Reference

file contain the tests for DIO Driver

```
#include "Test_Dio.h"
#include "MCAL/DIO Driver/dio.h"
#include <stdio.h>
#include <assert.h>
```

Include dependency graph for Test_Dio.c:



Functions

- void TRAFF_DIO_001 ()

- void TRAFF_DIO_002 ()
- void TRAFF_DIO_003 ()
- void TRAFF_DIO_004 ()
- void TRAFF_DIO_005 ()
- void TRAFF_DIO_006 ()
- void TRAFF_DIO_007 ()
- void TRAFF_DIO_008 ()
- void TRAFF_DIO_009 ()
- void TRAFF_DIO_010 ()
- void TRAFF_DIO_011 ()
- void TRAFF_DIO_012 ()
- void TRAFF_DIO_013 ()
- void TRAFF_DIO_014 ()
- void TRAFF_DIO_015 ()
- void TRAFF_DIO_016 ()
- void TRAFF_DIO_017 ()
- void TRAFF_DIO_018 ()
- void TRAFF_DIO_019 ()
- void TRAFF_DIO_020 ()
- void TRAFF_DIO_021 ()
- void TRAFF_DIO_022 ()
- void TRAFF_DIO_023 ()
- void TRAFF_DIO_024 ()
- void TRAFF_DIO_025 ()
- void TRAFF_DIO_026 ()
- void TRAFF_DIO_027 ()
- void TRAFF_DIO_028 ()
- void TRAFF_DIO_029 ()
- void TRAFF_DIO_030 ()
- void TRAFF_DIO_031 ()
- void TRAFF_DIO_032 ()
- void TRAFF_DIO_033 ()
- void TRAFF_DIO_034 ()
- void TRAFF_DIO_035 ()

3.24.1 Detailed Description

file contain the tests for DIO Driver

Author

Moataz Khaled

Version

0.1

Date

2022-09-13

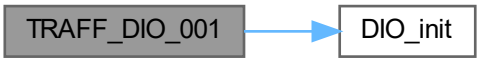
Copyright

3.24.2 Function Documentation

3.24.2.1 TRAFF_DIO_001()

```
void TRAFF_DIO_001 ( )
```

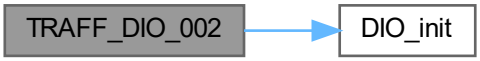
Here is the call graph for this function:



3.24.2.2 TRAFF_DIO_002()

```
void TRAFF_DIO_002 ( )
```

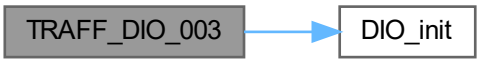
Here is the call graph for this function:



3.24.2.3 TRAFF_DIO_003()

```
void TRAFF_DIO_003 ( )
```

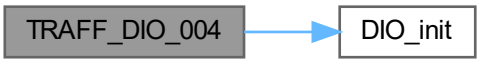
Here is the call graph for this function:



3.24.2.4 TRAFF_DIO_004()

```
void TRAFF_DIO_004 ( )
```

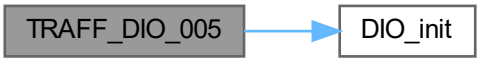
Here is the call graph for this function:



3.24.2.5 TRAFF_DIO_005()

```
void TRAFF_DIO_005 ( )
```

Here is the call graph for this function:



3.24.2.6 TRAFF_DIO_006()

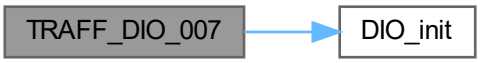
```
void TRAFF_DIO_006 ( )
```

Here is the call graph for this function:

3.24.2.7 TRAFF_DIO_007()

```
void TRAFF_DIO_007 ( )
```

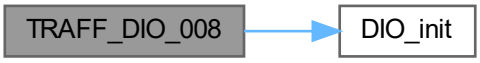
Here is the call graph for this function:



3.24.2.8 TRAFF_DIO_008()

```
void TRAFF_DIO_008 ( )
```

Here is the call graph for this function:



3.24.2.9 TRAFF_DIO_009()

```
void TRAFF_DIO_009 ( )
```

Here is the call graph for this function:

3.24.2.10 TRAFF_DIO_010()

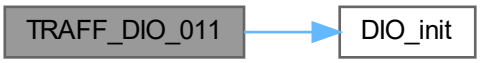
```
void TRAFF_DIO_010 ( )
```

Here is the call graph for this function:

3.24.2.11 TRAFF_DIO_011()

```
void TRAFF_DIO_011 ( )
```

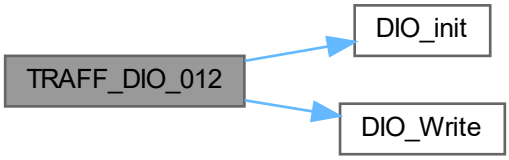
Here is the call graph for this function:



3.24.2.12 TRAFF_DIO_012()

```
void TRAFF_DIO_012 ( )
```

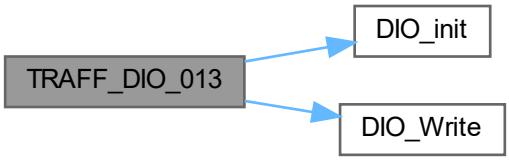
Here is the call graph for this function:



3.24.2.13 TRAFF_DIO_013()

```
void TRAFF_DIO_013 ( )
```

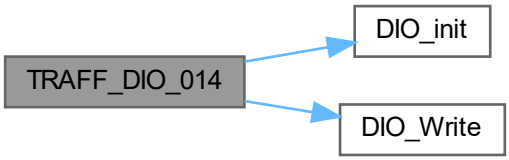
Here is the call graph for this function:



3.24.2.14 TRAFF_DIO_014()

```
void TRAFF_DIO_014 ( )
```

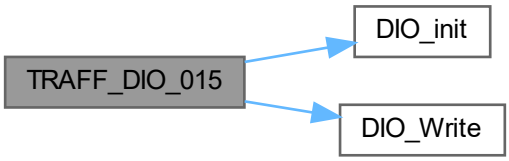
Here is the call graph for this function:



3.24.2.15 TRAFF_DIO_015()

```
void TRAFF_DIO_015 ( )
```

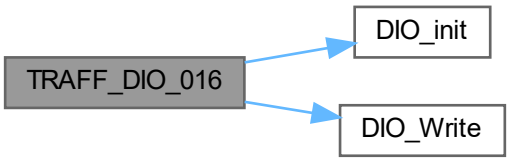
Here is the call graph for this function:



3.24.2.16 TRAFF_DIO_016()

```
void TRAFF_DIO_016 ( )
```

Here is the call graph for this function:



3.24.2.17 TRAFF_DIO_017()

```
void TRAFF_DIO_017 ( )
```

Here is the call graph for this function:

3.24.2.18 TRAFF_DIO_018()

```
void TRAFF_DIO_018 ( )
```

Here is the call graph for this function:

3.24.2.19 TRAFF_DIO_019()

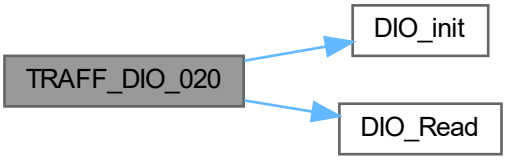
```
void TRAFF_DIO_019 ( )
```

Here is the call graph for this function:

3.24.2.20 TRAFF_DIO_020()

```
void TRAFF_DIO_020 ( )
```

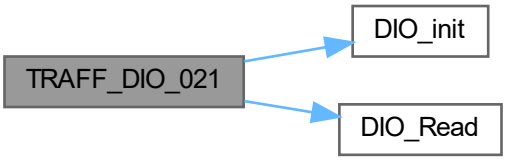
Here is the call graph for this function:



3.24.2.21 TRAFF_DIO_021()

```
void TRAFF_DIO_021 ( )
```

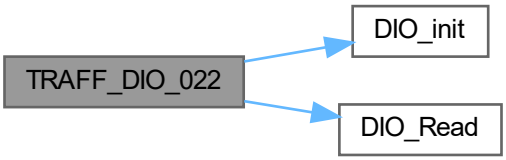
Here is the call graph for this function:



3.24.2.22 TRAFF_DIO_022()

```
void TRAFF_DIO_022 ( )
```

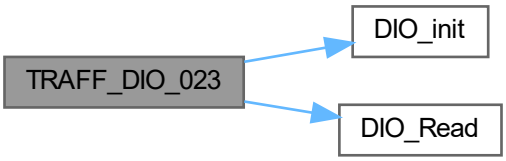
Here is the call graph for this function:



3.24.2.23 TRAFF_DIO_023()

```
void TRAFF_DIO_023 ( )
```

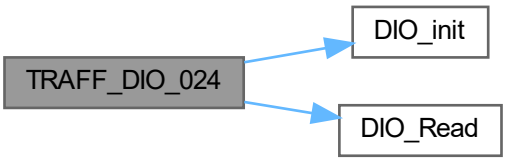
Here is the call graph for this function:



3.24.2.24 TRAFF_DIO_024()

```
void TRAFF_DIO_024 ( )
```

Here is the call graph for this function:



3.24.2.25 TRAFF_DIO_025()

```
void TRAFF_DIO_025 ( )
```

Here is the call graph for this function:

3.24.2.26 TRAFF_DIO_026()

```
void TRAFF_DIO_026 ( )
```

Here is the call graph for this function:

3.24.2.27 TRAFF_DIO_027()

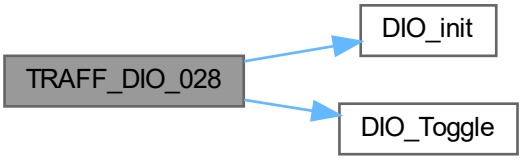
```
void TRAFF_DIO_027 ( )
```

Here is the call graph for this function:

3.24.2.28 TRAFF_DIO_028()

```
void TRAFF_DIO_028 ( )
```

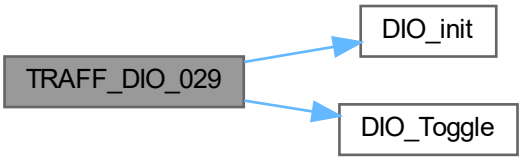
Here is the call graph for this function:



3.24.2.29 TRAFF_DIO_029()

```
void TRAFF_DIO_029 ( )
```

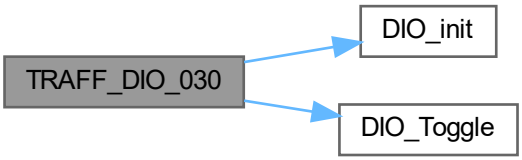
Here is the call graph for this function:



3.24.2.30 TRAFF_DIO_030()

```
void TRAFF_DIO_030 ( )
```

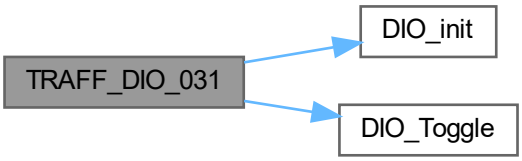
Here is the call graph for this function:



3.24.2.31 TRAFF_DIO_031()

```
void TRAFF_DIO_031 ( )
```

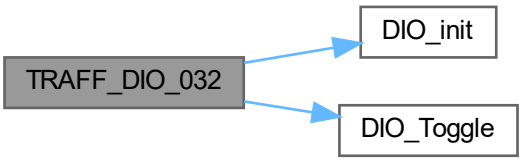
Here is the call graph for this function:



3.24.2.32 TRAFF_DIO_032()

```
void TRAFF_DIO_032 ( )
```

Here is the call graph for this function:



3.24.2.33 TRAFF_DIO_033()

```
void TRAFF_DIO_033 ( )
```

Here is the call graph for this function:

3.24.2.34 TRAFF_DIO_034()

```
void TRAFF_DIO_034 ( )
```

Here is the call graph for this function:

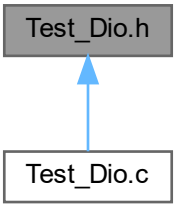
3.24.2.35 TRAFF_DIO_035()

```
void TRAFF_DIO_035 ( )
```

Here is the call graph for this function:

3.25 Test_Dio.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define PORT_E 'E'

Functions

- void TRAFF_DIO_001 ()
- void TRAFF_DIO_002 ()
- void TRAFF_DIO_003 ()
- void TRAFF_DIO_004 ()
- void TRAFF_DIO_005 ()
- void TRAFF_DIO_006 ()
- void TRAFF_DIO_007 ()
- void TRAFF_DIO_008 ()
- void TRAFF_DIO_009 ()
- void TRAFF_DIO_010 ()
- void TRAFF_DIO_011 ()
- void TRAFF_DIO_012 ()
- void TRAFF_DIO_013 ()
- void TRAFF_DIO_014 ()
- void TRAFF_DIO_015 ()
- void TRAFF_DIO_016 ()
- void TRAFF_DIO_017 ()
- void TRAFF_DIO_018 ()
- void TRAFF_DIO_019 ()
- void TRAFF_DIO_020 ()
- void TRAFF_DIO_021 ()
- void TRAFF_DIO_022 ()
- void TRAFF_DIO_023 ()
- void TRAFF_DIO_024 ()
- void TRAFF_DIO_025 ()
- void TRAFF_DIO_026 ()
- void TRAFF_DIO_027 ()
- void TRAFF_DIO_028 ()
- void TRAFF_DIO_029 ()
- void TRAFF_DIO_030 ()
- void TRAFF_DIO_031 ()
- void TRAFF_DIO_032 ()
- void TRAFF_DIO_033 ()
- void TRAFF_DIO_034 ()
- void TRAFF_DIO_035 ()

3.25.1 Macro Definition Documentation

3.25.1.1 PORT_E

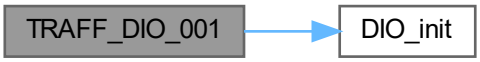
```
#define PORT_E 'E'
```

3.25.2 Function Documentation

3.25.2.1 TRAFF_DIO_001()

```
void TRAFF_DIO_001 ( )
```

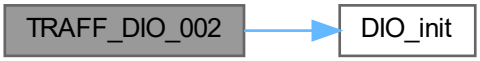
Here is the call graph for this function:



3.25.2.2 TRAFF_DIO_002()

```
void TRAFF_DIO_002 ( )
```

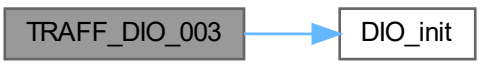
Here is the call graph for this function:



3.25.2.3 TRAFF_DIO_003()

```
void TRAFF_DIO_003 ( )
```

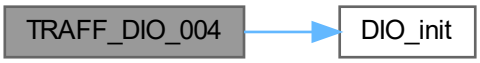
Here is the call graph for this function:



3.25.2.4 TRAFF_DIO_004()

```
void TRAFF_DIO_004 ( )
```

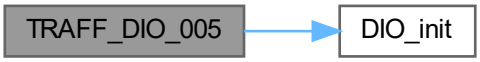
Here is the call graph for this function:



3.25.2.5 TRAFF_DIO_005()

```
void TRAFF_DIO_005 ( )
```

Here is the call graph for this function:



3.25.2.6 TRAFF_DIO_006()

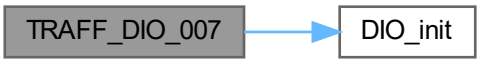
```
void TRAFF_DIO_006 ( )
```

Here is the call graph for this function:

3.25.2.7 TRAFF_DIO_007()

```
void TRAFF_DIO_007 ( )
```

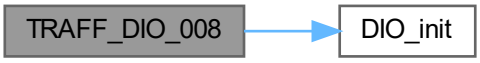
Here is the call graph for this function:



3.25.2.8 TRAFF_DIO_008()

```
void TRAFF_DIO_008 ( )
```

Here is the call graph for this function:



3.25.2.9 TRAFF_DIO_009()

```
void TRAFF_DIO_009 ( )
```

Here is the call graph for this function:

3.25.2.10 TRAFF_DIO_010()

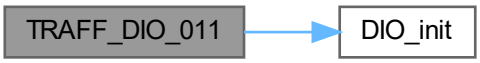
```
void TRAFF_DIO_010 ( )
```

Here is the call graph for this function:

3.25.2.11 TRAFF_DIO_011()

```
void TRAFF_DIO_011 ( )
```

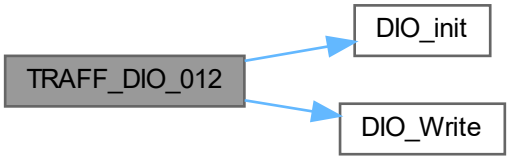
Here is the call graph for this function:



3.25.2.12 TRAFF_DIO_012()

```
void TRAFF_DIO_012 ( )
```

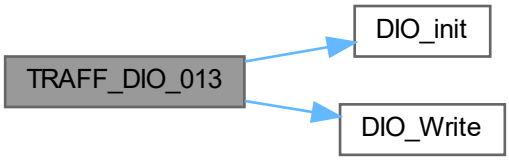
Here is the call graph for this function:



3.25.2.13 TRAFF_DIO_013()

```
void TRAFF_DIO_013 ( )
```

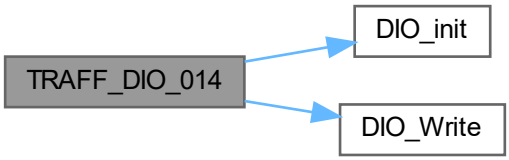
Here is the call graph for this function:



3.25.2.14 TRAFF_DIO_014()

```
void TRAFF_DIO_014 ( )
```

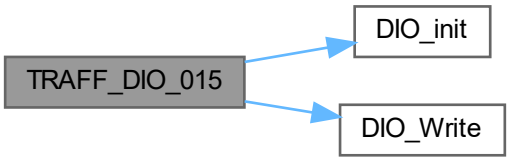
Here is the call graph for this function:



3.25.2.15 TRAFF_DIO_015()

```
void TRAFF_DIO_015 ( )
```

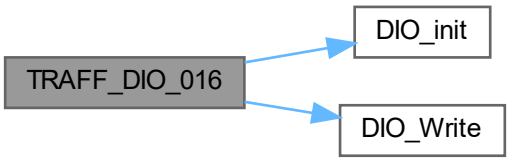
Here is the call graph for this function:



3.25.2.16 TRAFF_DIO_016()

```
void TRAFF_DIO_016 ( )
```

Here is the call graph for this function:



3.25.2.17 TRAFF_DIO_017()

```
void TRAFF_DIO_017 ( )
```

Here is the call graph for this function:

3.25.2.18 TRAFF_DIO_018()

```
void TRAFF_DIO_018 ( )
```

Here is the call graph for this function:

3.25.2.19 TRAFF_DIO_019()

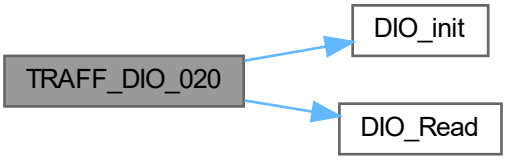
```
void TRAFF_DIO_019 ( )
```

Here is the call graph for this function:

3.25.2.20 TRAFF_DIO_020()

```
void TRAFF_DIO_020 ( )
```

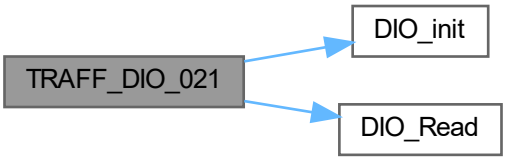
Here is the call graph for this function:



3.25.2.21 TRAFF_DIO_021()

```
void TRAFF_DIO_021 ( )
```

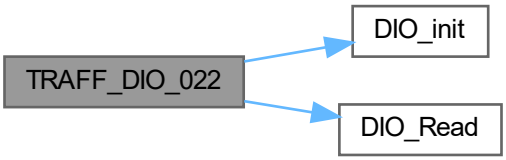
Here is the call graph for this function:



3.25.2.22 TRAFF_DIO_022()

```
void TRAFF_DIO_022 ( )
```

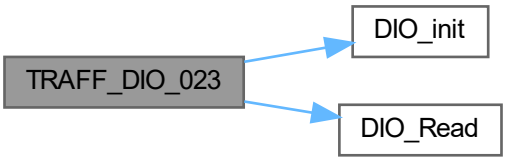
Here is the call graph for this function:



3.25.2.23 TRAFF_DIO_023()

```
void TRAFF_DIO_023 ( )
```

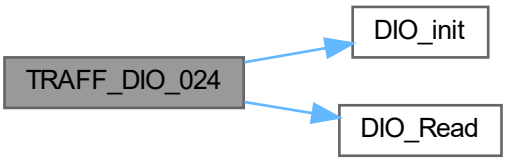
Here is the call graph for this function:



3.25.2.24 TRAFF_DIO_024()

```
void TRAFF_DIO_024 ( )
```

Here is the call graph for this function:



3.25.2.25 TRAFF_DIO_025()

```
void TRAFF_DIO_025 ( )
```

Here is the call graph for this function:

3.25.2.26 TRAFF_DIO_026()

```
void TRAFF_DIO_026 ( )
```

Here is the call graph for this function:

3.25.2.27 TRAFF_DIO_027()

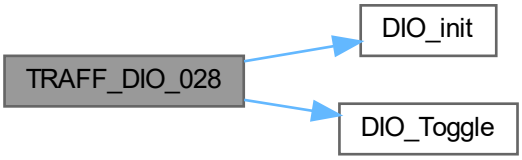
```
void TRAFF_DIO_027 ( )
```

Here is the call graph for this function:

3.25.2.28 TRAFF_DIO_028()

```
void TRAFF_DIO_028 ( )
```

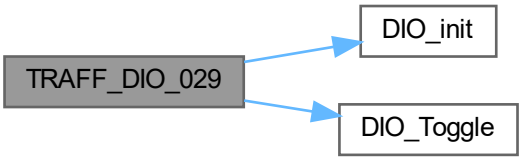
Here is the call graph for this function:



3.25.2.29 TRAFF_DIO_029()

```
void TRAFF_DIO_029 ( )
```

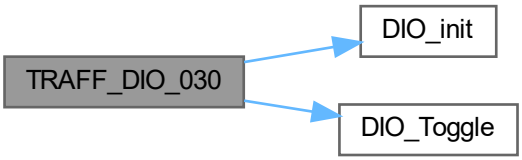
Here is the call graph for this function:



3.25.2.30 TRAFF_DIO_030()

```
void TRAFF_DIO_030 ( )
```

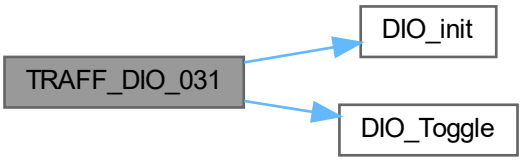
Here is the call graph for this function:



3.25.2.31 TRAFF_DIO_031()

```
void TRAFF_DIO_031 ( )
```

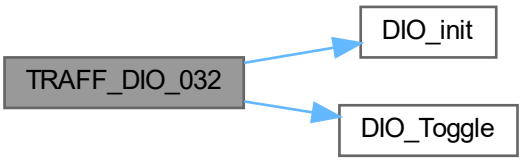
Here is the call graph for this function:



3.25.2.32 TRAFF_DIO_032()

```
void TRAFF_DIO_032 ( )
```

Here is the call graph for this function:



3.25.2.33 TRAFF_DIO_033()

```
void TRAFF_DIO_033 ( )
```

Here is the call graph for this function:

3.25.2.34 TRAFF_DIO_034()

```
void TRAFF_DIO_034 ( )
```

Here is the call graph for this function:

3.25.2.35 TRAFF_DIO_035()

```
void TRAFF_DIO_035 ( )
```

Here is the call graph for this function:

3.26 Test_Dio.h

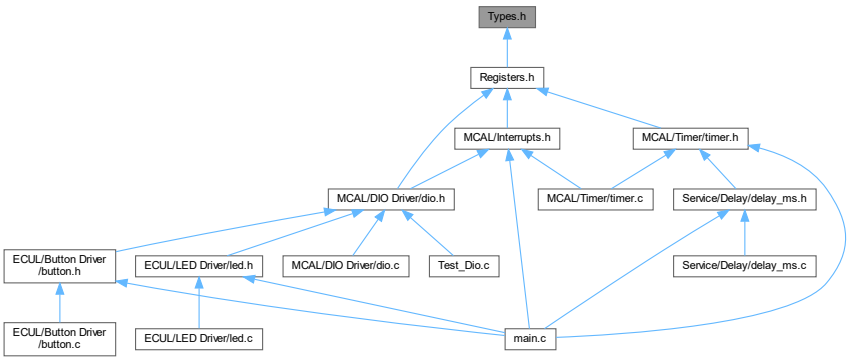
Go to the documentation of this file.

```
1
2
3
4 #ifndef TEST_DIO_H_
5 #define TEST_DIO_H_
6
7
8 #define  PORT_E 'E'
9
10 void TRAFF_DIO_001();
11 void TRAFF_DIO_002();
12 void TRAFF_DIO_003();
13 void TRAFF_DIO_004();
14 void TRAFF_DIO_005();
15 void TRAFF_DIO_006();
16 void TRAFF_DIO_007();
17 void TRAFF_DIO_008();
18 void TRAFF_DIO_009();
19 void TRAFF_DIO_010();
20 void TRAFF_DIO_011();
21 void TRAFF_DIO_012();
22 void TRAFF_DIO_013();
```

```
23 void TRAFF_DIO_014();
24 void TRAFF_DIO_015();
25 void TRAFF_DIO_016();
26 void TRAFF_DIO_017();
27 void TRAFF_DIO_018();
28 void TRAFF_DIO_019();
29 void TRAFF_DIO_020();
30 void TRAFF_DIO_021();
31 void TRAFF_DIO_022();
32 void TRAFF_DIO_023();
33 void TRAFF_DIO_024();
34 void TRAFF_DIO_025();
35 void TRAFF_DIO_026();
36 void TRAFF_DIO_027();
37 void TRAFF_DIO_028();
38 void TRAFF_DIO_029();
39 void TRAFF_DIO_030();
40 void TRAFF_DIO_031();
41 void TRAFF_DIO_032();
42 void TRAFF_DIO_033();
43 void TRAFF_DIO_034();
44 void TRAFF_DIO_035();
45
46 #endif /* INCFILE1_H_ */
```

3.27 Types.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define NULL 0 /∗∗ Define NULL in the program∗/
- #define TRUE 0xAA

  Define the true Value.
- #define FALSE 0x55

Typedefs

- typedef enum Status Status

  Enum showing the state of the function.
- typedef unsigned char uint8_t
- typedef unsigned int uint16_t

Enumerations

- enum Status { Ok , Not_ok }
  Enum showing the state of the function.

3.27.1 Macro Definition Documentation

3.27.1.1 FALSE

```
#define FALSE 0x55
```

3.27.1.2 NULL

```
#define NULL 0 /** Define NULL in the program*/
```

3.27.1.3 TRUE

```
#define TRUE 0xAA
```

Define the true Value.

3.27.2 Typedef Documentation

3.27.2.1 Status

```
typedef enum Status Status
```

Enum showing the state of the function.

3.27.2.2 uint16_t

```
typedef unsigned int uint16_t
```

3.27.2.3 uint8_t

```
typedef unsigned char uint8_t
```

3.27.3 Enumeration Type Documentation

3.27.3.1 Status

```
enum Status
```

Enum showing the state of the function.

Enumerator

Ok	
Not_ok	

3.28 Types.h

Go to the documentation of this file.

```
1 /*
2 * Types.h
3 *
4 * Created:  9/6/2022 10:19:47 PM
5 *  Author:  Moataz
6 */
7
8
9 #ifndef TYPES_H_
10 #define TYPES_H_
11
12
13 #define NULL 0
16 #define TRUE 0xAA
17 //*Define the False value*/
18 #define FALSE 0x55
19
24 typedef enum Status{
25     Ok,
26     Not_ok
27     }Status;
28
29
30 typedef unsigned char uint8_t;
31 typedef unsigned int uint16_t;
32
33
34 #endif /* TYPES_H_ */
```

Index