# On-Demand Traffic Control

Generated by Doxygen 1.9.4

# Chapter 1

# Module Index

## 1.1 Modules

Here is a list of all modules:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# Chapter 3

# Module Documentation

## 3.1 Service layer

### Modules

- MCU Registers

### 3.1.1 Detailed Description

## 3.2 MCU Registers

### Modules

- Port A registers

### 3.2.1 Detailed Description

## 3.3 Port A registers

### Macros

- #define PORTA (∗((volatile uint8_t∗)0x3B))

  *Output register for port A.*
- #define DDRA (∗((volatile uint8_t∗)0x3A))

  *Direction register for port A.*
- #define PINA (∗((volatile uint8_t∗)0x39))

  *Input register for port A.*
- #define PORTB (∗((volatile uint8_t∗)0x38))

  *Output register for port B.*
- #define DDRB (∗((volatile uint8_t∗)0x37))

  *Direction register for port B.*
- #define PINB (∗((volatile uint8_t∗)0x36))

*Input register for port A.*
- #define PORTC (∗((volatile uint8_t∗)0x35))

  *Direction register for port C.*
- #define DDRC (∗((volatile uint8_t∗)0x34))

  *Direction register for port C.*
- #define PINC (∗((volatile uint8_t∗)0x33))

  *Input register for port C.*
- #define PORTD (∗((volatile uint8_t∗)0x32))

  *Direction register for port D.*
- #define DDRD (∗((volatile uint8_t∗)0x31))

  *Direction register for port D.*
- #define PIND (∗((volatile uint8_t∗)0x30))

  *Input register for port D.*

### 3.3.1 Detailed Description

**Note**

x may be (A,B,C, or D) and n from 0 to 7.

- Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. The DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.
- The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.
- If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running. \argIf PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an out put pin, the port pin is driven low (zero).

### 3.3.2 Macro Definition Documentation

#### 3.3.2.1 DDRA

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

Direction register for port A.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

#### 3.3.2.2 DDRB

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

Direction register for port B.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

#### 3.3.2.3 DDRC

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

Direction register for port C.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

#### 3.3.2.4 DDRD

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

Direction register for port D.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

#### 3.3.2.5 PINA

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

Input register for port A.

- This register stores the input values of port A.
- If the value is 1 then the applied voltage on this pin is high.
- If the value is 0 then the applied voltage on this pin is low.

#### 3.3.2.6 PINB

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

Input register for port A.

- This register stores the input values of port B.
- If the value is 1 then the applied voltage on this pin is high.
- If the value is 0 then the applied voltage on this pin is low.

#### 3.3.2.7 PINC

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

Input register for port C.

- This register stores the input values of port C.
- If the value is 1 then the applied voltage on this pin is high.
- If the value is 0 then the applied voltage on this pin is low.

#### 3.3.2.8 PIND

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

Input register for port D.

- This register stores the input values of port D.
- If the value is 1 then the applied voltage on this pin is high.
- If the value is 0 then the applied voltage on this pin is low.

#### 3.3.2.9 PORTA

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

Output register for port A.

- This register controls the output of the pin.
- Setting the bit in this register will make the pin high.
- Clearing the bit in this register will make the pin low
- If the pin is configured as output through DDRx and we write high to PORTx register this will activate internal pull up resistor (x may be A,B,C or D).

#### 3.3.2.10 PORTB

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

Output register for port B.

- This register controls the output of the pin.
- Setting the bit in this register will make the pin high.
- Clearing the bit in this register will make the pin low
- If the pin is configured as output through DDRx and we write high to PORTx register this will activate internal pull up resistor (x may be A,B,C or D).

#### 3.3.2.11 PORTC

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

Direction register for port C.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

#### 3.3.2.12 PORTD

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

Direction register for port D.

- This register controls the direction of the pin.
- Setting the bit in this register will make the pin output.
- Clearing the bit in this register will make the pin input

# Chapter 4

# File Documentation

## 4.1 App/app.c File Reference

## 4.2 App/app.h File Reference

## 4.3 app.h

Go to the documentation of this file.

```
/***************************************************************************************************************************************
/*                                                         Author   :  Ehab Omara
*/
/*                                                         Date     :  8/10/2022 12:03:55 PM
*/
/*                                                         File name:  app.h
*/
/***************************************************************************************************************************************

#ifndef APP_H_
#define APP_H_





#endif /* APP_H_ */
```

## 4.4 Debug/App/app.d File Reference

## 4.5 Debug/ECUAL/Button driver/Button.d File Reference

## 4.6 Debug/main.d File Reference

## 4.7 Debug/MCAL/Dio driver/DIO.d File Reference

## 4.8 ECUAL/Button driver/Button.c File Reference

```
#include "../../Service/ATmega32Port.h"
#include "Button.h"
```

## Functions

- EN_pinErro_t buttonInit (EN_pinNum_t buttonPin)

### 4.8.1 Function Documentation

#### 4.8.1.1 buttonInit()

```
EN_pinErro_t buttonInit (
            EN_pinNum_t buttonPin )
```

## 4.9 ECUAL/Button driver/Button.h File Reference

```
#include "../../MCAL/Dio driver/DIO.h"
#include "../../Service/ATmega32Port.h"
```

## Functions

- EN_pinErro_t buttonInit (EN_pinNum_t buttonPin)

### 4.9.1 Function Documentation

#### 4.9.1.1 buttonInit()

```
EN_pinErro_t buttonInit (
            EN_pinNum_t buttonPin )
```

## 4.10 Button.h

Go to the documentation of this file.

```
/*******************************************************************************************************************
/*                                                       Author    :  Ehab Omara
*/
/*                                                       Date      :  8/11/2022 8:24:25 PM
*/
/*                                                       File name:  Button.h
*/
/*******************************************************************************************************************

#ifndef BUTTON_H_
#define BUTTON_H_

#include "../../MCAL/Dio driver/DIO.h"
#include "../../Service/ATmega32Port.h"
/*
*buttonInit function
*Description:
*1.This function makes the button pin as Input.
*Input arguments:
*1.buttonPin it is the pin which the button is connected to,it may be (PA0 to PD7).
*Output arguments:
*1.no output arguments
*Return value:
*1.It return WRONG_PIN_NUM if the pinNum is wrong.
*2.It return OK if the pinNum and the pinDirection are correct.
*/
EN_pinErro_t buttonInit(EN_pinNum_t buttonPin);



#endif /* BUTTON_H_ */
```

## 4.11 main.c File Reference

`#include "./MCAL/Dio driver/DIO.h"`

### Functions

- int main (void)

### 4.11.1 Function Documentation

#### 4.11.1.1 main()

```
int main (
            void  )
```

## 4.12 MCAL/Dio driver/DIO.c File Reference

`#include "DIO.h"`

## Functions

- EN_pinErro_t DIO_pinInit (EN_pinNum_t pinNum, EN_pinDirection_t pinDirection)

    *Set the direction of the pin.*
- EN_pinErro_t DIO_pinWrite (EN_pinNum_t pinNum, EN_pinState_t pinState)

    *This function writes High or Low on the pin.*
- EN_pinErro_t DIO_pinRead (EN_pinNum_t pinNum, EN_pinState_t ∗pinState)

    *This function reads the state of the pin.*
- EN_pinErro_t DIO_pinToggle (EN_pinNum_t pinNum)

    *This function toggles the state of the pin.*

### 4.12.1 Function Documentation

#### 4.12.1.1 DIO_pinInit()

```
EN_pinErro_t DIO_pinInit (
            EN_pinNum_t pinNum,
            EN_pinDirection_t pinDirection )
```

Set the direction of the pin.

DIO_pinInit function

- This function makes pin input or output.
- it makes the pinNum Output by setting the pinNum in the DDRx (x:A,B,C or D) register.
- it makes the pinNum Input by clearing the pinNum in the DDRx (x:A,B,C or D) register.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| in | *pinDirection* | it represent the pin direction it may be (Input or Output). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *WRONG_PIN_DIR* | if the pinDirection is wrong. |
| *OK* | if the pinNum and the pinDirection are correct. |

#### 4.12.1.2 DIO_pinRead()

```
EN_pinErro_t DIO_pinRead (
              EN_pinNum_t pinNum,
              EN_pinState_t * pinState )
```

This function reads the state of the pin.

DIO_pinRead

- It reads the bit relative to the pinNum in the register PINx (A,B,C or D).

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| out | *pinState* | this is a pointer to store the state of the pin (High or Low). |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *OK* | if the pinNum is correct. |

#### 4.12.1.3 DIO_pinToggle()

```
EN_pinErro_t DIO_pinToggle (
              EN_pinNum_t pinNum )
```

This function toggles the state of the pin.

DIO_pinToggle

- if the current state of the pin is High it will make it Low.
- if the current state of the pin is Low it will make it High.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *OK* | if the pinNum is correct. |

#### 4.12.1.4 DIO_pinWrite()

```
EN_pinErro_t DIO_pinWrite (
            EN_pinNum_t pinNum,
            EN_pinState_t pinState )
```

This function writes High or Low on the pin.

DIO_pinWrite

- it writes High to the pinNum by setting the pinNum in the PORTx (x:A,B,C or D) register.
- it writes Low to the pinNum by clearing the pinNum in the PORTx (x:A,B,C or D) register.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (`PA0` to `PD7`). |
| in | *pinState* | it represent the pin state it may be (High or Low). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *WRONG_PIN_STATE* | if the pinState is wrong. |
| *OK* | if the pinNum and the pinState are correct. |

## 4.13 MCAL/Dio driver/DIO.h File Reference

```
#include "../../Service/ATmega32Port.h"
#include "../../Service/BitMath.h"
#include "../../Service/dataTypes.h"
#include "../../Service/RegisterFile.h"
```

### Functions

- EN_pinErro_t DIO_pinInit (EN_pinNum_t pinNum, EN_pinDirection_t pinDirection)

  *Set the direction of the pin.*
- EN_pinErro_t DIO_pinWrite (EN_pinNum_t pinNum, EN_pinState_t pinState)

  *This function writes High or Low on the pin.*
- EN_pinErro_t DIO_pinToggle (EN_pinNum_t pinNum)

  *This function toggles the state of the pin.*
- EN_pinErro_t DIO_pinRead (EN_pinNum_t pinNum, EN_pinState_t ∗pinState)

  *This function reads the state of the pin.*

### 4.13.1 Detailed Description

**Author**

: Ehab Omara

**Date**

: 8/10/2022 3:39:36 PM

### 4.13.2 Function Documentation

#### 4.13.2.1 DIO_pinInit()

```
EN_pinErro_t DIO_pinInit (
             EN_pinNum_t pinNum,
             EN_pinDirection_t pinDirection )
```

Set the direction of the pin.

DIO_pinInit function

- This function makes pin input or output.
- it makes the pinNum Output by setting the pinNum in the DDRx (x:A,B,C or D) register.
- it makes the pinNum Input by clearing the pinNum in the DDRx (x:A,B,C or D) register.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| in | *pinDirection* | it represent the pin direction it may be (Input or Output). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *WRONG_PIN_DIR* | if the pinDirection is wrong. |
| *OK* | if the pinNum and the pinDirection are correct. |

#### 4.13.2.2 DIO_pinRead()

```
EN_pinErro_t DIO_pinRead (
             EN_pinNum_t pinNum,
             EN_pinState_t * pinState )
```

This function reads the state of the pin.

DIO_pinRead

- It reads the bit relative to the pinNum in the register PINx (A,B,C or D).

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| out | *pinState* | this is a pointer to store the state of the pin (High or Low). |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *OK* | if the pinNum is correct. |

#### 4.13.2.3 DIO_pinToggle()

```
EN_pinErro_t DIO_pinToggle (
             EN_pinNum_t pinNum )
```

This function toggles the state of the pin.

DIO_pinToggle

- if the current state of the pin is High it will make it Low.
- if the current state of the pin is Low it will make it High.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *OK* | if the pinNum is correct. |

#### 4.13.2.4 DIO_pinWrite()

```
EN_pinErro_t DIO_pinWrite (
              EN_pinNum_t pinNum,
              EN_pinState_t pinState )
```

This function writes High or Low on the pin.

DIO_pinWrite

- it writes High to the pinNum by setting the pinNum in the PORTx (x:A,B,C or D) register.
- it writes Low to the pinNum by clearing the pinNum in the PORTx (x:A,B,C or D) register.

**Parameters**

| | | |
|---|---|---|
| in | *pinNum* | it represent the pin number (PA0 to PD7). |
| in | *pinState* | it represent the pin state it may be (High or Low). |
| out | *none* | no output arguments |

**Return values**

| | |
|---|---|
| *WRONG_PIN_NUM* | if the pinNum is wrong. |
| *WRONG_PIN_STATE* | if the pinState is wrong. |
| *OK* | if the pinNum and the pinState are correct. |

## 4.14 DIO.h

Go to the documentation of this file.

```
      /***********************************************************************************************************************
#ifndef DIO_H_
#define DIO_H_

#include "../../Service/ATmega32Port.h"
#include "../../Service/BitMath.h"
#include "../../Service/dataTypes.h"
#include "../../Service/RegisterFile.h"

EN_pinErro_t DIO_pinInit(EN_pinNum_t pinNum,EN_pinDirection_t pinDirection);
EN_pinErro_t DIO_pinWrite(EN_pinNum_t pinNum,EN_pinState_t pinState);
EN_pinErro_t DIO_pinToggle(EN_pinNum_t pinNum);
EN_pinErro_t DIO_pinRead(EN_pinNum_t pinNum,EN_pinState_t *pinState);



#endif /* DIO_H_ */
```

## 4.15 Service/ATmega32Port.h File Reference

### Macros

- #define PORTA_OFFSET 0
- #define PORTB_OFFSET 8
- #define PORTC_OFFSET 16
- #define PORTD_OFFSET 24

## Enumerations

- enum EN_pinNum_t {
  PA0 , PA1 , PA2 , PA3 ,
  PA4 , PA5 , PA6 , PA7 ,
  PB0 , PB1 , PB2 , PB3 ,
  PB4 , PB5 , PB6 , PB7 ,
  PC0 , PC1 , PC2 , PC3 ,
  PC4 , PC5 , PC6 , PC7 ,
  PD0 , PD1 , PD2 , PD3 ,
  PD4 , PD5 , PD6 , PD7 }
- enum EN_pinState_t { Low , High }
- enum EN_pinDirection_t { Input , Output }
- enum EN_pinErro_t { OK , WRONG_PIN_NUM , WRONG_PIN_DIR , WRONG_PIN_STATE }

## 4.15.1 Macro Definition Documentation

### 4.15.1.1 PORTA_OFFSET

```
#define PORTA_OFFSET 0
```

This macro defines the start of the PORTA pins

### 4.15.1.2 PORTB_OFFSET

```
#define PORTB_OFFSET 8
```

This macro defines the start of the PORTB pins

### 4.15.1.3 PORTC_OFFSET

```
#define PORTC_OFFSET 16
```

This macro defines the start of the PORTC pins

### 4.15.1.4 PORTD_OFFSET

```
#define PORTD_OFFSET 24
```

This macro defines the start of the PORTD pins

## 4.15.2 Enumeration Type Documentation

### 4.15.2.1 EN_pinDirection_t

```
enum EN_pinDirection_t
```

**Enumerator**

| | |
|---|---|
| Input | enum value for input direction |
| Output | enum value for output direction |

### 4.15.2.2 EN_pinErro_t

`enum EN_pinErro_t`

**Enumerator**

| | |
|---|---|
| OK | enum value that defines that the pin parameters are ok |
| WRONG_PIN_NUM | enum value that defines that the pin number is wrong |
| WRONG_PIN_DIR | enum value that defines that the pin direction is wrong |
| WRONG_PIN_STATE | enum value that defines that the pin state is wrong |

### 4.15.2.3 EN_pinNum_t

`enum EN_pinNum_t`

This enum contains the value for all pins of the MCU of the four ports (PORTA,PORTB,PORTC,PORTD)

**Enumerator**

| | |
|---|---|
| PA0 | enum value for PORTA pin 0 |
| PA1 | enum value for PORTA pin 1 |
| PA2 | enum value for PORTA pin 2 |
| PA3 | enum value for PORTA pin 3 |
| PA4 | enum value for PORTA pin 4 |
| PA5 | enum value for PORTA pin 5 |
| PA6 | enum value for PORTA pin 6 |
| PA7 | enum value for PORTA pin 7 |
| PB0 | enum value for PORTB pin 0 |
| PB1 | enum value for PORTB pin 1 |
| PB2 | enum value for PORTB pin 2 |
| PB3 | enum value for PORTB pin 3 |
| PB4 | enum value for PORTB pin 4 |
| PB5 | enum value for PORTB pin 5 |
| PB6 | enum value for PORTB pin 6 |
| PB7 | enum value for PORTB pin 7 |
| PC0 | enum value for PORTC pin 0 |
| PC1 | enum value for PORTC pin 1 |
| PC2 | enum value for PORTC pin 2 |

**Enumerator**

| | |
|---|---|
| PC3 | enum value for PORTC pin 3 |
| PC4 | enum value for PORTC pin 4 |
| PC5 | enum value for PORTC pin 5 |
| PC6 | enum value for PORTC pin 6 |
| PC7 | enum value for PORTC pin 7 |
| PD0 | enum value for PORTD pin 0 |
| PD1 | enum value for PORTD pin 1 |
| PD2 | enum value for PORTD pin 2 |
| PD3 | enum value for PORTD pin 3 |
| PD4 | enum value for PORTD pin 4 |
| PD5 | enum value for PORTD pin 5 |
| PD6 | enum value for PORTD pin 6 |
| PD7 | enum value for PORTD pin 7 |

#### 4.15.2.4 EN_pinState_t

```
enum EN_pinState_t
```

**Enumerator**

| | |
|---|---|
| Low | enum value for Low output |
| High | enum value for high output |

## 4.16 ATmega32Port.h

Go to the documentation of this file.

```
/***********************************************************************************************************************
/*                                                        Author    :  Ehab Omara
    */
/*                                                        Date      :  8/10/2022 3:49:55 PM
    */
/*                                                        File name:  ATmega32Port.h
    */
/***********************************************************************************************************************

#ifndef ATMEGA32PORT_H_
#define ATMEGA32PORT_H_

typedef enum
{
    /*PORTA pins*/
    PA0,
    PA1,
    PA2,
    PA3,
    PA4,
    PA5,
    PA6,
    PA7,
    /*PORTB pins*/
    PB0,
    PB1,
    PB2,
    PB3,
    PB4,
    PB5,
    PB6,
    PB7,
    /*PORTC pins*/
    PC0,
    PC1,
    PC2,
    PC3,
    PC4,
    PC5,
    PC6,
    PC7,
    /*PORTD pins*/
    PD0,
    PD1,
    PD2,
    PD3,
    PD4,
    PD5,
    PD6,
    PD7
}EN_pinNum_t;

#define PORTA_OFFSET    0
#define PORTB_OFFSET    8
#define PORTC_OFFSET    16
#define PORTD_OFFSET    24
typedef enum
{
    Low,
    High
}EN_pinState_t;
typedef enum
{
    Input,
    Output
}EN_pinDirection_t;
typedef enum
{
    OK,
    WRONG_PIN_NUM,
    WRONG_PIN_DIR,
    WRONG_PIN_STATE
}EN_pinErro_t;



#endif /* ATMEGA32PORT_H_ */
```

## 4.17 Service/BitMath.h File Reference

### Macros

- #define setBit(reg, bitNum) reg |= (1<<bitNum)
- #define clrBit(reg, bitNum) reg &= (∼(1<<bitNum))
- #define toggleBit(reg, bitNum) reg ^= (1<<bitNum)
- #define getBit(reg, bitNum) ((reg>>bitNum) & 0x01)

### 4.17.1 Macro Definition Documentation

#### 4.17.1.1 clrBit

```
#define clrBit(
                reg,
                bitNum ) reg &= (∼(1<<bitNum))
```

#### 4.17.1.2 getBit

```
#define getBit(
               reg,
               bitNum ) ((reg>>bitNum) & 0x01)
```

#### 4.17.1.3 setBit

```
#define setBit(
               reg,
               bitNum ) reg |= (1<<bitNum)
```

#### 4.17.1.4 toggleBit

```
#define toggleBit(
               reg,
               bitNum ) reg ^= (1<<bitNum)
```

## 4.18 BitMath.h

Go to the documentation of this file.

```
/***********************************************************************************************************************
/*                                                     Author   :  Ehab Omara
 */
/*                                                     Date     :  8/10/2022 12:46:40 PM
 */
/*                                                     File name:  BitMath.h
 */
/***********************************************************************************************************************

#ifndef BITMATH_H_
#define BITMATH_H_

/*
*setBit function
*1.this function takes register (reg) and bit number (bitNum).
*2.it make the required bit in the register High(1).
*/
#define setBit(reg,bitNum)  reg |= (1«bitNum)
/*
*clrBit function
*1.this function takes register (reg) and bit number (bitNum).
*2.it make the required bit in the register Low(0).
*/
#define clrBit(reg,bitNum)  reg &= (~(1«bitNum))
/*
*togBit function
*1.this function takes register (reg) and bit number (bitNum).
*2.it toggle the state of the required bit in the register.
*3.if the required bit is Low(0) it makes it High(1).
*4.if the required bit is High(1) it makes it Low(0).
*/
#define toggleBit(reg,bitNum)   reg ^= (1«bitNum)
/*
*getBit function
*1.this function takes register (reg) and bit number (bitNum).
*2.it returns the state of the required bit in the register.
*3.if the required bit is Low(0) it will return 0.
*4.if the required bit is High(1) it will return 1.
*/
#define getBit(reg,bitNum)              ((reg»bitNum)  &   0x01)



#endif /* BITMATH_H_ */
```

## 4.19 Service/dataTypes.h File Reference

### Typedefs

- typedef unsigned char uint8_t
- typedef signed char sint8_t
- typedef unsigned short int uint16_t
- typedef signed short int sint16_t
- typedef unsigned long int uint32_t
- typedef signed long int sint32_t
- typedef float float32_t
- typedef double float64_t
- typedef long double float128_t

### 4.19.1 Typedef Documentation

#### 4.19.1.1 float128_t

```
typedef long double float128_t
```

#### 4.19.1.2 float32_t

```
typedef float float32_t
```

#### 4.19.1.3 float64_t

```
typedef double float64_t
```

#### 4.19.1.4 sint16_t

```
typedef signed short int sint16_t
```

#### 4.19.1.5 sint32_t

```
typedef signed long int sint32_t
```

#### 4.19.1.6 sint8_t

```
typedef signed char sint8_t
```

#### 4.19.1.7 uint16_t

```
typedef unsigned short int uint16_t
```

#### 4.19.1.8 uint32_t

```
typedef unsigned long int uint32_t
```

#### 4.19.1.9 uint8_t

```
typedef unsigned char uint8_t
```

## 4.20 dataTypes.h

Go to the documentation of this file.

```
/**************************************************************************************************************************
/*                                                         Author   :  Ehab Omara
*/
/*                                                         Date     :  8/10/2022 12:06:28 PM
*/
/*                                                         File name:  dataTypes.h
*/
/**************************************************************************************************************************

#ifndef DATATYPES_H_
#define DATATYPES_H_

typedef unsigned char        uint8_t;
typedef signed char          sint8_t;
typedef unsigned short int   uint16_t;
typedef signed short int     sint16_t;
typedef unsigned long int    uint32_t;
typedef signed long int      sint32_t;
typedef float                float32_t;
typedef double               float64_t;
typedef long double          float128_t;



#endif /* DATATYPES_H_ */
```

## 4.21 Service/RegisterFile.h File Reference

### Macros

- #define PORTA (∗((volatile uint8_t∗)0x3B))

  *Output register for port A.*
- #define DDRA (∗((volatile uint8_t∗)0x3A))

  *Direction register for port A.*
- #define PINA (∗((volatile uint8_t∗)0x39))

  *Input register for port A.*
- #define PORTB (∗((volatile uint8_t∗)0x38))

  *Output register for port B.*
- #define DDRB (∗((volatile uint8_t∗)0x37))

  *Direction register for port B.*
- #define PINB (∗((volatile uint8_t∗)0x36))

  *Input register for port A.*
- #define PORTC (∗((volatile uint8_t∗)0x35))

  *Direction register for port C.*
- #define DDRC (∗((volatile uint8_t∗)0x34))

  *Direction register for port C.*
- #define PINC (∗((volatile uint8_t∗)0x33))

  *Input register for port C.*
- #define PORTD (∗((volatile uint8_t∗)0x32))

  *Direction register for port D.*
- #define DDRD (∗((volatile uint8_t∗)0x31))

  *Direction register for port D.*
- #define PIND (∗((volatile uint8_t∗)0x30))

  *Input register for port D.*

## 4.22 RegisterFile.h

Go to the documentation of this file.

```
/************************************************************************************************************
/*                                                        Author    :  Ehab Omara                           */
/*                                                        Date      :  8/10/2022 12:06:56 PM                */
/*                                                        File name:  RegisterFile.h                        */
/************************************************************************************************************

#ifndef REGISTERFILE_H_
#define REGISTERFILE_H_

/*
* if the DDRx is set to be output and we write High to the PORTx
* this will activate the internal Pull up resistor.
*/

/************************************************************* Port A registers
*************************************************************/

#define PORTA   (*((volatile uint8_t*)0x3B))  //1->high output              0->low output
#define DDRA    (*((volatile uint8_t*)0x3A))  //1->to make it output        0->to make it input
#define PINA    (*((volatile uint8_t*)0x39))  //this register to read a value from a pin



```

```
/************************************************************ Port B registers
   ************************************************************/

#define PORTB   (*((volatile uint8_t*)0x38))
#define DDRB    (*((volatile uint8_t*)0x37))
#define PINB    (*((volatile uint8_t*)0x36))

/************************************************************ Port C registers
   ************************************************************/
#define PORTC   (*((volatile uint8_t*)0x35))
#define DDRC    (*((volatile uint8_t*)0x34))
#define PINC    (*((volatile uint8_t*)0x33))
/************************************************************ Port D registers
   ************************************************************/

#define PORTD   (*((volatile uint8_t*)0x32))
#define DDRD    (*((volatile uint8_t*)0x31))
#define PIND    (*((volatile uint8_t*)0x30))
#endif /* REGISTERFILE_H_ */
```

# Index