

On-Demand Traffic Control

Generated by Doxygen 1.9.4

1 Module Index	1
1.1 Modules	1
2 File Index	3
2.1 File List	3
3 Module Documentation	5
3.1 Service layer	5
3.1.1 Detailed Description	5
3.2 MCU Registers	5
3.2.1 Detailed Description	5
3.3 Port A registers	5
3.3.1 Detailed Description	6
3.3.2 Macro Definition Documentation	6
3.3.2.1 DDRA	6
3.3.2.2 DDRB	7
3.3.2.3 DDRC	7
3.3.2.4 DDRD	7
3.3.2.5 PINA	7
3.3.2.6 PINB	8
3.3.2.7 PINC	8
3.3.2.8 PIND	8
3.3.2.9 PORTA	8
3.3.2.10 PORTB	9
3.3.2.11 PORTC	9
3.3.2.12 PORTD	9
4 File Documentation	11
4.1 App/app.c File Reference	11
4.2 App/app.h File Reference	11
4.3 app.h	11
4.4 Debug/App/app.d File Reference	11
4.5 Debug/ECUAL/Button driver/Button.d File Reference	11
4.6 Debug/main.d File Reference	11
4.7 Debug/MCAL/Dio driver/DIO.d File Reference	11
4.8 ECUAL/Button driver/Button.c File Reference	11
4.8.1 Function Documentation	12
4.8.1.1 buttonInit()	12
4.9 ECUAL/Button driver/Button.h File Reference	12
4.9.1 Function Documentation	12
4.9.1.1 buttonInit()	12
4.10 Button.h	13
4.11 main.c File Reference	13

4.11.1 Function Documentation	13
4.11.1.1 main()	13
4.12 MCAL/Dio driver/DIO.c File Reference	13
4.12.1 Function Documentation	14
4.12.1.1 DIO_pinInit()	14
4.12.1.2 DIO_pinRead()	14
4.12.1.3 DIO_pinToggle()	15
4.12.1.4 DIO_pinWrite()	16
4.13 MCAL/Dio driver/DIO.h File Reference	16
4.13.1 Detailed Description	17
4.13.2 Function Documentation	17
4.13.2.1 DIO_pinInit()	17
4.13.2.2 DIO_pinRead()	17
4.13.2.3 DIO_pinToggle()	18
4.13.2.4 DIO_pinWrite()	19
4.14 DIO.h	19
4.15 Service/ATmega32Port.h File Reference	19
4.15.1 Macro Definition Documentation	20
4.15.1.1 PORTA_OFFSET	20
4.15.1.2 PORTB_OFFSET	20
4.15.1.3 PORTC_OFFSET	20
4.15.1.4 PORTD_OFFSET	20
4.15.2 Enumeration Type Documentation	20
4.15.2.1 EN_pinDirection_t	20
4.15.2.2 EN_pinError_t	21
4.15.2.3 EN_pinNum_t	21
4.15.2.4 EN_pinState_t	22
4.16 ATmega32Port.h	22
4.17 Service/BitMath.h File Reference	23
4.17.1 Macro Definition Documentation	23
4.17.1.1 clrBit	23
4.17.1.2 getBit	24
4.17.1.3 setBit	24
4.17.1.4 toggleBit	24
4.18 BitMath.h	24
4.19 Service/dataTypes.h File Reference	25
4.19.1 Typedef Documentation	25
4.19.1.1 float128_t	25
4.19.1.2 float32_t	25
4.19.1.3 float64_t	25
4.19.1.4 sint16_t	25
4.19.1.5 sint32_t	25

4.19.1.6 sint8_t	26
4.19.1.7 uint16_t	26
4.19.1.8 uint32_t	26
4.19.1.9 uint8_t	26
4.20 dataTypes.h	26
4.21 Service/RegisterFile.h File Reference	27
4.22 RegisterFile.h	27
Index	29

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Service layer	5
MCU Registers	5
Port A registers	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

main.c	13
App/ app.c	11
App/ app.h	11
Debug/ main.d	11
Debug/App/ app.d	11
Debug/ECUAL/Button driver/ Button.d	11
Debug/MCAL/Dio driver/ DIO.d	11
ECUAL/Button driver/ Button.c	11
ECUAL/Button driver/ Button.h	12
MCAL/Dio driver/ DIO.c	13
MCAL/Dio driver/ DIO.h	16
Service/ ATmega32Port.h	19
Service/ BitMath.h	23
Service/ dataTypes.h	25
Service/ RegisterFile.h	27

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.](#)

```
1
2  /* ***** Author : Ehab Omara ***** */
3  /* ***** Date : 8/10/2022 12:03:55 PM ***** */
4  /* ***** File name: app.h ***** */
5  /* ***** */
6
7  #ifndef APP_H_
8  #define APP_H_
9
10
11
12
13
14 #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.](#)

```

1
2  /* ***** Author : Ehab Omara ***** */
3  /* ***** Date : 8/11/2022 8:24:25 PM ***** */
4  /* ***** File name: Button.h ***** */
5  /* ***** */
6
7  #ifndef BUTTON_H_
8  #define BUTTON_H_
9
10 #include "../MCAL/Dio driver/DIO.h"
11 #include "../Service/ATmega32Port.h"
12 /*
13 *buttonInit function
14 *Description:
15 *1.This function makes the button pin as Input.
16 *Input arguments:
17 *1.buttonPin it is the pin which the button is connected to,it may be (PA0 to PD7).
18 *Output arguments:
19 *1.no output arguments
20 *Return value:
21 *1.It return WRONG_PIN_NUM if the pinNum is wrong.
22 *2.It return OK if the pinNum and the pinDirection are correct.
23 */
24 EN_pinErro_t buttonInit(EN_pinNum_t buttonPin);
25
26
27
28 #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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
in	<i>pinDirection</i>	it represent the pin direction it may be (Input or Output).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>WRONG_PIN_DIR</i>	if the pinDirection is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
out	<i>pinState</i>	this is a pointer to store the state of the pin (High or Low).

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
in	<i>pinState</i>	it represent the pin state it may be (High or Low).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>WRONG_PIN_STATE</i>	if the pinState is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
in	<i>pinDirection</i>	it represent the pin direction it may be (Input or Output).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>WRONG_PIN_DIR</i>	if the pinDirection is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
out	<i>pinState</i>	this is a pointer to store the state of the pin (High or Low).

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>OK</i>	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	<i>pinNum</i>	it represent the pin number (PA0 to PD7).
in	<i>pinState</i>	it represent the pin state it may be (High or Low).
out	<i>none</i>	no output arguments

Return values

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>WRONG_PIN_STATE</i>	if the pinState is wrong.
<i>OK</i>	if the pinNum and the pinState are correct.

4.14 DIO.h

[Go to the documentation of this file.](#)

```

1
2  /*****
3  #ifndef DIO_H_
4  #define DIO_H_
5
6  #include "../Service/ATmega32Port.h"
7  #include "../Service/BitMath.h"
8  #include "../Service/dataTypes.h"
9  #include "../Service/RegisterFile.h"
10
11 EN_pinErro_t DIO_pinInit(EN_pinNum_t pinNum, EN_pinDirection_t pinDirection);
12 EN_pinErro_t DIO_pinWrite(EN_pinNum_t pinNum, EN_pinState_t pinState);
13 EN_pinErro_t DIO_pinToggle(EN_pinNum_t pinNum);
14 EN_pinErro_t DIO_pinRead(EN_pinNum_t pinNum, EN_pinState_t *pinState);
15
16 #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.](#)

```

1
2  /* ***** Author : Ehab Omara ***** */
3  /* ***** Date : 8/10/2022 3:49:55 PM ***** */
4  /* ***** File name: ATmega32Port.h ***** */
5
6  /* ***** */
7  #ifndef ATMEGA32PORT_H_
8  #define ATMEGA32PORT_H_
9
10 typedef enum
11 {
12     /*PORTA pins*/
13     PA0,
14     PA1,
15     PA2,
16     PA3,
17     PA4,
18     PA5,
19     PA6,
20     PA7,
21     /*PORTB pins*/
22     PB0,

```

```

28     PB1,
29     PB2,
30     PB3,
31     PB4,
32     PB5,
33     PB6,
34     PB7,
35     /*PORTC pins*/
36     PC0,
37     PC1,
38     PC2,
39     PC3,
40     PC4,
41     PC5,
42     PC6,
43     PC7,
44     /*PORTD pins*/
45     PD0,
46     PD1,
47     PD2,
48     PD3,
49     PD4,
50     PD5,
51     PD6,
52     PD7
53 }EN_pinNum_t;
54
55 #define PORTA_OFFSET    0
56 #define PORTB_OFFSET    8
57 #define PORTC_OFFSET    16
58 #define PORTD_OFFSET    24
59 typedef enum
60 {
61     Low,
62     High
63 }EN_pinState_t;
64 typedef enum
65 {
66     Input,
67     Output
68 }EN_pinDirection_t;
69 typedef enum
70 {
71     OK,
72     WRONG_PIN_NUM,
73     WRONG_PIN_DIR,
74     WRONG_PIN_STATE
75 }EN_pinErro_t;
76
77
78
79
80 #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.](#)

```
1
2 /*
3  /*
4  /*
5
6
7 #ifndef BITMATH_H_
8 #define BITMATH_H_
9
10 /*
11 *setBit function
12 *1.this function takes register (reg) and bit number (bitNum).
13 *2.it make the required bit in the register High(1).
14 */
15 #define setBit(reg,bitNum)  reg |= (1<<bitNum)
16 /*
17 *clrBit function
18 *1.this function takes register (reg) and bit number (bitNum).
19 *2.it make the required bit in the register Low(0).
20 */
21 #define clrBit(reg,bitNum)  reg &= (~(1<<bitNum))
22 /*
23 *togBit function
24 *1.this function takes register (reg) and bit number (bitNum).
25 *2.it toggle the state of the required bit in the register.
26 *3.if the required bit is Low(0) it makes it High(1).
27 *4.if the required bit is High(1) it makes it Low(0).
28 */
29 #define toggleBit(reg,bitNum)  reg ^= (1<<bitNum)
30 /*
31 *getBit function
32 *1.this function takes register (reg) and bit number (bitNum).
33 *2.it returns the state of the required bit in the register.
34 *3.if the required bit is Low(0) it will return 0.
35 *4.if the required bit is High(1) it will return 1.
36 */
37 #define getBit(reg,bitNum)      ((reg>>bitNum) & 0x01)
38
39
40
41 #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.](#)

```
1
2  /* ***** Author : Ehab Omara ***** */
3  /* ***** Date : 8/10/2022 12:06:28 PM ***** */
4  /* ***** File name: dataTypes.h ***** */
5  /* ***** */
6
7  #ifndef DATATYPES_H_
8  #define DATATYPES_H_
9
10 typedef unsigned char    uint8_t;
11 typedef signed char      sint8_t;
12 typedef unsigned short int uint16_t;
13 typedef signed short int sint16_t;
14 typedef unsigned long int uint32_t;
15 typedef signed long int  sint32_t;
16 typedef float            float32_t;
17 typedef double           float64_t;
18 typedef long double      float128_t;
19
20
21
22 #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.](#)

```

1
2  /* ***** Author : Ehab Omara *****
3  /*
4  /* ***** Date : 8/10/2022 12:06:56 PM *****
5  /*
6  /* ***** File name: RegisterFile.h *****
7
8  #ifndef REGISTERFILE_H_
9  #define REGISTERFILE_H_
10
11 /*
12 * if the DDRx is set to be output and we write High to the PORTx
13 * this will activate the internal Pull up resistor.
14 */
15
16 /* ***** Port A registers *****
17 *****
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53 #define PORTA  (*((volatile uint8_t*)0x3B)) //1->high output      0->low output
54 #define DDRA   (*((volatile uint8_t*)0x3A)) //1->to make it output 0->to make it input
55 #define PINA   (*((volatile uint8_t*)0x39)) //this register to read a value from a pin
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

```
73 /***** Port B registers
    *****/
74
83 #define PORTB    (*((volatile uint8_t*)0x38))
91 #define DDRB     (*((volatile uint8_t*)0x37))
99 #define PINB     (*((volatile uint8_t*)0x36))
100
101 /***** Port C registers
    *****/
109 #define PORTC    (*((volatile uint8_t*)0x35))
117 #define DDRC     (*((volatile uint8_t*)0x34))
125 #define PINC     (*((volatile uint8_t*)0x33))
126 /***** Port D registers
    *****/
127
135 #define PORTD    (*((volatile uint8_t*)0x32))
143 #define DDRD     (*((volatile uint8_t*)0x31))
151 #define PIND     (*((volatile uint8_t*)0x30))
155 #endif /* REGISTERFILE_H_ */
```

Index

App/app.c, [11](#)
App/app.h, [11](#)
ATmega32Port.h
 EN_pinDirection_t, [20](#)
 EN_pinErro_t, [21](#)
 EN_pinNum_t, [21](#)
 EN_pinState_t, [22](#)
 High, [22](#)
 Input, [21](#)
 Low, [22](#)
 OK, [21](#)
 Output, [21](#)
 PA0, [21](#)
 PA1, [21](#)
 PA2, [21](#)
 PA3, [21](#)
 PA4, [21](#)
 PA5, [21](#)
 PA6, [21](#)
 PA7, [21](#)
 PB0, [21](#)
 PB1, [21](#)
 PB2, [21](#)
 PB3, [21](#)
 PB4, [21](#)
 PB5, [21](#)
 PB6, [21](#)
 PB7, [21](#)
 PC0, [21](#)
 PC1, [21](#)
 PC2, [21](#)
 PC3, [22](#)
 PC4, [22](#)
 PC5, [22](#)
 PC6, [22](#)
 PC7, [22](#)
 PD0, [22](#)
 PD1, [22](#)
 PD2, [22](#)
 PD3, [22](#)
 PD4, [22](#)
 PD5, [22](#)
 PD6, [22](#)
 PD7, [22](#)
 PORTA_OFFSET, [20](#)
 PORTB_OFFSET, [20](#)
 PORTC_OFFSET, [20](#)
 PORTD_OFFSET, [20](#)
 WRONG_PIN_DIR, [21](#)

 WRONG_PIN_NUM, [21](#)
 WRONG_PIN_STATE, [21](#)

BitMath.h
 clrBit, [23](#)
 getBit, [23](#)
 setBit, [24](#)
 toggleBit, [24](#)
Button.c
 buttonInit, [12](#)
Button.h
 buttonInit, [12](#)
buttonInit
 Button.c, [12](#)
 Button.h, [12](#)

clrBit
 BitMath.h, [23](#)

dataTypes.h
 float128_t, [25](#)
 float32_t, [25](#)
 float64_t, [25](#)
 sint16_t, [25](#)
 sint32_t, [25](#)
 sint8_t, [25](#)
 uint16_t, [26](#)
 uint32_t, [26](#)
 uint8_t, [26](#)
DDRA
 Port A registers, [6](#)
DDRB
 Port A registers, [6](#)
DDRC
 Port A registers, [7](#)
DDRD
 Port A registers, [7](#)
Debug/App/app.d, [11](#)
Debug/ECUAL/Button driver/Button.d, [11](#)
Debug/main.d, [11](#)
Debug/MCAL/Dio driver/DIO.d, [11](#)
DIO.c
 DIO_pinInit, [14](#)
 DIO_pinRead, [14](#)
 DIO_pinToggle, [15](#)
 DIO_pinWrite, [15](#)
DIO.h
 DIO_pinInit, [17](#)
 DIO_pinRead, [17](#)
 DIO_pinToggle, [18](#)

DIO_pinWrite, [18](#)
 DIO_pinInit
 DIO.c, [14](#)
 DIO.h, [17](#)
 DIO_pinRead
 DIO.c, [14](#)
 DIO.h, [17](#)
 DIO_pinToggle
 DIO.c, [15](#)
 DIO.h, [18](#)
 DIO_pinWrite
 DIO.c, [15](#)
 DIO.h, [18](#)

 ECUAL/Button driver/Button.c, [11](#)
 ECUAL/Button driver/Button.h, [12](#), [13](#)
 EN_pinDirection_t
 ATmega32Port.h, [20](#)
 EN_pinErro_t
 ATmega32Port.h, [21](#)
 EN_pinNum_t
 ATmega32Port.h, [21](#)
 EN_pinState_t
 ATmega32Port.h, [22](#)

 float128_t
 dataTypes.h, [25](#)
 float32_t
 dataTypes.h, [25](#)
 float64_t
 dataTypes.h, [25](#)

 getBit
 BitMath.h, [23](#)

 High
 ATmega32Port.h, [22](#)

 Input
 ATmega32Port.h, [21](#)

 Low
 ATmega32Port.h, [22](#)

 main
 main.c, [13](#)
 main.c, [13](#)
 main, [13](#)
 MCAL/Dio driver/DIO.c, [13](#)
 MCAL/Dio driver/DIO.h, [16](#), [19](#)
 MCU Registers, [5](#)

 OK
 ATmega32Port.h, [21](#)
 Output
 ATmega32Port.h, [21](#)

 PA0
 ATmega32Port.h, [21](#)
 PA1
 ATmega32Port.h, [21](#)
 PA2
 ATmega32Port.h, [21](#)
 PA3
 ATmega32Port.h, [21](#)
 PA4
 ATmega32Port.h, [21](#)
 PA5
 ATmega32Port.h, [21](#)
 PA6
 ATmega32Port.h, [21](#)
 PA7
 ATmega32Port.h, [21](#)
 PB0
 ATmega32Port.h, [21](#)
 PB1
 ATmega32Port.h, [21](#)
 PB2
 ATmega32Port.h, [21](#)
 PB3
 ATmega32Port.h, [21](#)
 PB4
 ATmega32Port.h, [21](#)
 PB5
 ATmega32Port.h, [21](#)
 PB6
 ATmega32Port.h, [21](#)
 PB7
 ATmega32Port.h, [21](#)
 PC0
 ATmega32Port.h, [21](#)
 PC1
 ATmega32Port.h, [21](#)
 PC2
 ATmega32Port.h, [21](#)
 PC3
 ATmega32Port.h, [22](#)
 PC4
 ATmega32Port.h, [22](#)
 PC5
 ATmega32Port.h, [22](#)
 PC6
 ATmega32Port.h, [22](#)
 PC7
 ATmega32Port.h, [22](#)
 PD0
 ATmega32Port.h, [22](#)
 PD1
 ATmega32Port.h, [22](#)
 PD2
 ATmega32Port.h, [22](#)
 PD3
 ATmega32Port.h, [22](#)
 PD4
 ATmega32Port.h, [22](#)
 PD5
 ATmega32Port.h, [22](#)
 PD6

- ATmega32Port.h, [22](#)
- PD7
 - ATmega32Port.h, [22](#)
- PINA
 - Port A registers, [7](#)
- PINB
 - Port A registers, [7](#)
- PINC
 - Port A registers, [8](#)
- PIND
 - Port A registers, [8](#)
- Port A registers, [5](#)
 - DDRA, [6](#)
 - DDRB, [6](#)
 - DDRC, [7](#)
 - DDRD, [7](#)
 - PINA, [7](#)
 - PINB, [7](#)
 - PINC, [8](#)
 - PIND, [8](#)
 - PORTA, [8](#)
 - PORTB, [8](#)
 - PORTC, [9](#)
 - PORTD, [9](#)
- PORTA
 - Port A registers, [8](#)
- PORTA_OFFSET
 - ATmega32Port.h, [20](#)
- PORTB
 - Port A registers, [8](#)
- PORTB_OFFSET
 - ATmega32Port.h, [20](#)
- PORTC
 - Port A registers, [9](#)
- PORTC_OFFSET
 - ATmega32Port.h, [20](#)
- PORTD
 - Port A registers, [9](#)
- PORTD_OFFSET
 - ATmega32Port.h, [20](#)
- Service layer, [5](#)
- Service/ATmega32Port.h, [19](#), [22](#)
- Service/BitMath.h, [23](#), [24](#)
- Service/dataTypes.h, [25](#), [26](#)
- Service/RegisterFile.h, [27](#)
- setBit
 - BitMath.h, [24](#)
- sint16_t
 - dataTypes.h, [25](#)
- sint32_t
 - dataTypes.h, [25](#)
- sint8_t
 - dataTypes.h, [25](#)
- toggleBit
 - BitMath.h, [24](#)
- uint16_t
 - dataTypes.h, [26](#)
- uint32_t
 - dataTypes.h, [26](#)
- uint8_t
 - dataTypes.h, [26](#)
- WRONG_PIN_DIR
 - ATmega32Port.h, [21](#)
- WRONG_PIN_NUM
 - ATmega32Port.h, [21](#)
- WRONG_PIN_STATE
 - ATmega32Port.h, [21](#)