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 ECUAL layer	5
	3.1.1 Detailed Description	5
	3.2 Button driver	5
	3.2.1 Detailed Description	5
	3.2.2 Function Documentation	5
	3.2.2.1 buttonInit()	5
	3.2.2.2 buttonRead()	6
	3.3 LED driver	6
	3.3.1 Detailed Description	7
	3.3.2 Function Documentation	7
	3.3.2.1 ledInit()	7
	3.3.2.2 ledOff()	7
	3.3.2.3 ledOn()	8
	3.3.2.4 ledToggle()	8
	3.4 MCAL layer	9
	3.4.1 Detailed Description	9
	3.5 DIO driver	9
	3.5.1 Detailed Description	10
	3.5.2 Function Documentation	10
	3.5.2.1 DIO_pinInit()	10
	3.5.2.2 DIO_pinRead()	10
	3.5.2.3 DIO_pinToggle()	11
	3.5.2.4 DIO_pinWrite()	11
	3.6 MCU ports	12
	3.6.1 Detailed Description	12
	3.6.2 Macro Definition Documentation	13
	3.6.2.1 PORTA OFFSET	13
	3.6.2.2 PORTB_OFFSET	13
	3.6.2.3 PORTC OFFSET	13
	3.6.2.4 PORTD_OFFSET	13
	3.6.3 Enumeration Type Documentation	13
	3.6.3.1 EN_pinDirection_t	13
	3.6.3.2 EN_pinErro_t	14
	3.6.3.3 EN_pinNum_t	14
	3.6.3.4 EN_pinState_t	15
	3.7 Bit math	15

3.7.1 Detailed Description	. 16
3.7.2 Macro Definition Documentation	. 16
3.7.2.1 clrBit	. 16
3.7.2.2 getBit	. 16
3.7.2.3 setBit	. 17
3.7.2.4 toggleBit	. 17
3.8 Definition of data types	. 18
3.8.1 Detailed Description	. 18
3.8.2 Typedef Documentation	. 18
3.8.2.1 float128_t	. 18
3.8.2.2 float32_t	. 18
3.8.2.3 float64_t	. 19
3.8.2.4 sint16_t	. 19
3.8.2.5 sint32_t	. 19
3.8.2.6 sint8_t	. 19
3.8.2.7 uint16_t	. 19
3.8.2.8 uint32_t	. 20
3.8.2.9 uint8_t	. 20
3.9 Service layer	. 20
3.9.1 Detailed Description	. 20
3.10 MCU Registers	. 20
3.10.1 Detailed Description	. 20
3.11 I/O registers	. 21
3.11.1 Detailed Description	. 21
3.12 Port A registers	. 21
3.12.1 Detailed Description	. 21
3.12.2 Macro Definition Documentation	. 21
3.12.2.1 DDRA	. 22
3.12.2.2 PINA	. 22
3.12.2.3 PORTA	. 22
3.13 Port B registers	. 23
3.13.1 Detailed Description	. 23
3.13.2 Macro Definition Documentation	. 23
3.13.2.1 DDRB	. 23
3.13.2.2 PINB	. 23
3.13.2.3 PORTB	. 24
3.14 Port C registers	. 24
3.14.1 Detailed Description	
3.14.2 Macro Definition Documentation	
3.14.2.1 DDRC	
3.14.2.2 PINC	
3.14.2.3 PORTC	

	3.15 Port D registers	25
	3.15.1 Detailed Description	25
	3.15.2 Macro Definition Documentation	25
	3.15.2.1 DDRD	26
	3.15.2.2 PIND	26
	3.15.2.3 PORTD	26
_		
4	File Documentation	27
	4.1 App/app.c File Reference	
	4.2 app.c	
	4.3 App/app.h File Reference	
	4.4 app.h	
	4.5 Debug/App/app.d File Reference	28
	4.6 app.d	28
	4.7 Debug/ECUAL/Button driver/Button.d File Reference	28
	4.8 Button.d	
	4.9 Debug/ECUAL/LED driver/LED.d File Reference	28
	4.10 LED.d	28
	4.11 Debug/main.d File Reference	29
	4.12 main.d	29
	4.13 Debug/MCAL/Dio driver/DIO.d File Reference	29
	4.14 DIO.d	29
	4.15 ECUAL/Button driver/Button.c File Reference	29
	4.16 Button.c	30
	4.17 ECUAL/Button driver/Button.h File Reference	30
	4.18 Button.h	30
	4.19 ECUAL/LED driver/LED.c File Reference	31
	4.20 LED.c	31
	4.21 ECUAL/LED driver/LED.h File Reference	31
	4.22 LED.h	32
	4.23 main.c File Reference	32
	4.23.1 Function Documentation	32
	4.23.1.1 main()	33
	4.24 main.c	33
	4.25 MCAL/Dio driver/DIO.c File Reference	33
	4.26 DIO.c	34
	4.27 MCAL/Dio driver/DIO.h File Reference	36
	4.27.1 Detailed Description	37
	4.28 DIO.h	37
	4.29 Service/ATmega32Port.h File Reference	37
		_
	4.30 ATmega32Port.h	38
	4.31 Service/BitMath.h File Reference	39

In	ndex	43
	4.36 RegisterFile.h	41
	4.35 Service/RegisterFile.h File Reference	40
	4.34 dataTypes.h	40
	4.33 Service/dataTypes.h File Reference	39
	4.32 BitMath.h	39

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

ECUAL layer	5
Button driver	. 5
LED driver	. 6
MCAL layer	9
DIO driver	. 9
Service layer	20
MCU ports	. 12
Bit math	. 15
Definition of data types	
MCU Registers	. 20
I/O registers	. 21
Port A registers	. 21
Port B registers	. 23
Port C registers	. 24
Port D registers	25

2 Module Index

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

main.c
App/app.c
App/app.h
Debug/main.d
Debug/App/app.d
Debug/ECUAL/Button driver/Button.d
Debug/ECUAL/LED driver/LED.d
Debug/MCAL/Dio driver/DIO.d
ECUAL/Button driver/Button.c
ECUAL/Button driver/Button.h
ECUAL/LED driver/LED.c
ECUAL/LED driver/LED.h
MCAL/Dio driver/DIO.c
MCAL/Dio driver/DIO.h
Service/ATmega32Port.h
Service/BitMath.h
Service/dataTypes.h
Service/RegisterFile.h 4

File Index

Chapter 3

Module Documentation

3.1 ECUAL layer

Modules

- · Button driver
- · LED driver

3.1.1 Detailed Description

This layer contains all the drivers for the external devices that connected to the MCU.

3.2 Button driver

Functions

- EN_pinErro_t buttonInit (EN_pinNum_t buttonPin) initialize the button pin.
- EN_pinErro_t buttonRead (EN_pinNum_t buttonPin, EN_pinState_t *pinState) reads the value of the button.

3.2.1 Detailed Description

This driver contains all the function that controls the buttons connected to the MCU.

3.2.2 Function Documentation

3.2.2.1 buttonInit()

initialize the button pin.

buttonInit function:

• This function makes the button pin as Input.

Parameters

in	buttonPin	it is the pin which the button is connected to,it may be (PA0 to PD7).	
out	none	no output arguments	

Return values

WRONG_PIN_NUM	if the pinNum is wrong.
OK	if the pinNum is correct.

Definition at line 11 of file Button.c.

3.2.2.2 buttonRead()

reads the value of the button.

buttonRead function:

- It reads the value of the connected pin to the button.
- It store the value in the pinState pointer.

Parameters

in	buttonPin	it is the pin which the button is connected to,it may be (PA0 to PD7).
out	pinState	the function store the value of the button in that pointer.

Return values

WRONG_PIN_NUM	if the pinNum is wrong.
OK	if the pinNum is correct.

Definition at line 16 of file Button.c.

3.3 LED driver

Functions

• EN_pinErro_t ledInit (EN_pinNum_t ledPin)

3.3 LED driver 7

```
initialize the led pin.
```

• EN_pinErro_t ledOn (EN_pinNum_t ledPin)

turn the led on.

• EN_pinErro_t ledOff (EN_pinNum_t ledPin)

turn the led off.

• EN_pinNum_t ledToggle (EN_pinNum_t ledPin)

toggle the led state.

3.3.1 Detailed Description

This driver contains all the function that controls the LEDs connected to the MCU.

3.3.2 Function Documentation

3.3.2.1 ledlnit()

initialize the led pin.

ledInit function:

• This function initialize the led pin as output.

Parameters

in	<i>ledPin</i>	it is the pin which the led is connected to,it may be (PA0 to PD7).
out	none	no output arguments

Return values

WRONG_PIN_NUM	if the pinNum is wrong.	
OK	if the pinNum is correct.	

Definition at line 10 of file LED.c.

3.3.2.2 ledOff()

turn the led off.

ledOff function:

• This function turns the led off by writing low to the pin.

Parameters

	in	ledPin	it is the pin which the led is connected to,it may be (PA0 to PD7).
Ī	out	none	no output arguments

Return values

WRONG_PIN_NUM	if the pinNum is wrong.
OK	if the pinNum is correct.

Definition at line 20 of file LED.c.

3.3.2.3 ledOn()

turn the led on.

ledOn function:

• This function turns the led on by writing high to the pin.

Parameters

in ledPin it is the pin which the out none no output arguments		ledPin	it is the pin which the led is connected to,it may be (PA0 to PD7).
		none	no output arguments

Return values

	WRONG_PIN_NUM	if the pinNum is wrong.	
ſ	OK	if the pinNum is correct.	

Definition at line 15 of file LED.c.

3.3.2.4 ledToggle()

```
EN_pinNum_t ledToggle (
```

3.4 MCAL layer 9

```
EN_pinNum_t ledPin )
```

toggle the led state.

ledToggle function:

- · This function toggle the led state.
- · It makes the led on if the led was off.
- It makes the led off if the led was on.

Parameters

	in	ledPin	it is the pin which the led is connected to,it may be (PA0 to PD7).
Ī	out none no output arguments		no output arguments

Return values

WRONG_PIN_NUM	if the pinNum is wrong.
OK	if the pinNum is correct.

Definition at line 25 of file LED.c.

3.4 MCAL layer

Modules

DIO driver

3.4.1 Detailed Description

This layer contains all the driver related to the MCU.

3.5 DIO driver

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.

3.5.1 Detailed Description

This contains all the function needed to configure and manipulate the MCU ports.

3.5.2 Function Documentation

3.5.2.1 DIO_pinInit()

Set the direction of the pin.

DIO_pinInit

- · 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.

Definition at line 12 of file DIO.c.

3.5.2.2 DIO_pinRead()

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).

3.5 DIO driver

Parameters

	in	pinNum	it represent the pin number (PA0 to PD7).
ſ	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.

Definition at line 166 of file DIO.c.

3.5.2.3 DIO_pinToggle()

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.

Definition at line 198 of file DIO.c.

3.5.2.4 DIO_pinWrite()

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.

Definition at line 90 of file DIO.c.

3.6 MCU ports

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 }
```

3.6.1 Detailed Description

This contains all the definition for MCU pins, input and output pins values and pins errors.

3.6 MCU ports 13

3.6.2 Macro Definition Documentation

3.6.2.1 PORTA_OFFSET

```
#define PORTA_OFFSET 0
```

This macro defines the start of the PORTA pins

Definition at line 62 of file ATmega32Port.h.

3.6.2.2 PORTB_OFFSET

```
#define PORTB_OFFSET 8
```

This macro defines the start of the PORTB pins

Definition at line 63 of file ATmega32Port.h.

3.6.2.3 PORTC_OFFSET

```
#define PORTC_OFFSET 16
```

This macro defines the start of the PORTC pins

Definition at line 64 of file ATmega32Port.h.

3.6.2.4 PORTD_OFFSET

```
#define PORTD_OFFSET 24
```

This macro defines the start of the PORTD pins

Definition at line 65 of file ATmega32Port.h.

3.6.3 Enumeration Type Documentation

3.6.3.1 EN_pinDirection_t

enum EN_pinDirection_t

Enumerator

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

Definition at line 72 of file ATmega32Port.h.

3.6.3.2 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

Definition at line 77 of file ATmega32Port.h.

3.6.3.3 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

3.7 Bit math

Enumerator

PC0	enum value for PORTC pin 0
PC1	enum value for PORTC pin 1
PC2	enum value for PORTC pin 2
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

Definition at line 22 of file ATmega32Port.h.

3.6.3.4 EN_pinState_t

enum EN_pinState_t

Enumerator

Low	enum value for Low output
High	enum value for high output

Definition at line 67 of file ATmega32Port.h.

3.7 Bit math

Macros

- #define setBit(reg, bitNum) reg |= (1<<bitNum)
 this Macro writes 1 to the bit.
- #define clrBit(reg, bitNum) reg &= (~(1<<bitNum))
 this Macro clear the bit.
- #define toggleBit(reg, bitNum) reg $^-$ = (1<<bitNum) This Macro toggle the bit logic.
- #define getBit(reg, bitNum) ((reg>>bitNum) & 0x01)
 This Macro read this bit value.

3.7.1 Detailed Description

Author: Ehab Omara

Date: 8/10/2022 12:46:40 PM

File name: BitMath.h

This contains all the bit math macros that manipulates the registers values.

3.7.2 Macro Definition Documentation

3.7.2.1 clrBit

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

this Macro clear the bit.

clrBit function

- this function takes register (reg) and bit number (bitNum).
- it make the required bit in the register Low(0).

Parameters

in	reg	this is register that needed to be changed.
in	bitNum	this is bit number that needed to be written to 0 in the register.

Definition at line 37 of file BitMath.h.

3.7.2.2 getBit

This Macro read this bit value.

getBit function

- this function takes register (reg) and bit number (bitNum).
- it returns the state of the required bit in the register.
- if the required bit is Low(0) it will return 0.
- if the required bit is High(1) it will return 1.

3.7 Bit math

Parameters

in	reg	This is register where it reads the value from it.
in	bitNum	This is the bit number that needed to be read.

Definition at line 62 of file BitMath.h.

3.7.2.3 setBit

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

this Macro writes 1 to the bit.

setBit function

- this function takes register (reg) and bit number (bitNum).
- it make the required bit in the register High(1).

Parameters

in	reg	this is register that needed to be changed.
in	bitNum	this is bit number that needed to be written to 1 in the register.

Definition at line 26 of file BitMath.h.

3.7.2.4 toggleBit

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

This Macro toggle the bit logic.

#togBit function

- this function takes register (reg) and bit number (bitNum).
- it toggle the state of the required bit in the register.
- if the required bit is Low(0) it makes it High(1).
- if the required bit is High(1) it makes it Low(0).

Parameters

in	reg	this is register that needed to be changed.
in	bitNum	this is bit number that needed to be changed in the register.

Definition at line 50 of file BitMath.h.

3.8 Definition of data types

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

3.8.1 Detailed Description

This file contains all the data types definitions that needed in this project.

3.8.2 Typedef Documentation

3.8.2.1 float128_t

```
typedef long double float128_t
```

This is define a memory size of 16 byte float

Definition at line 23 of file dataTypes.h.

3.8.2.2 float32_t

```
typedef float float32_t
```

This is define a memory size of 4 byte float

Definition at line 21 of file dataTypes.h.

3.8.2.3 float64_t

```
typedef double float64_t
```

This is define a memory size of 8 byte float

Definition at line 22 of file dataTypes.h.

3.8.2.4 sint16_t

```
typedef signed short int sint16_t
```

This is define a memory size of 2 byte signed

Definition at line 18 of file dataTypes.h.

3.8.2.5 sint32_t

```
typedef signed long int sint32_t
```

This is define a memory size of 4 byte signed

Definition at line 20 of file dataTypes.h.

3.8.2.6 sint8_t

```
typedef signed char sint8_t
```

This is define a memory size of 1 byte signed

Definition at line 16 of file dataTypes.h.

3.8.2.7 uint16_t

```
{\tt typedef \ unsigned \ short \ int \ uint16\_t}
```

This is define a memory size of 2 byte

Definition at line 17 of file dataTypes.h.

3.8.2.8 uint32_t

```
typedef unsigned long int uint32_t
```

This is define a memory size of 4 byte

Definition at line 19 of file dataTypes.h.

3.8.2.9 uint8_t

```
typedef unsigned char uint8_t
```

This is define a memory size of 1 byte

Definition at line 15 of file dataTypes.h.

3.9 Service layer

Modules

- MCU ports
- Bit math
- Definition of data types
- MCU Registers

3.9.1 Detailed Description

This layer contains all the common services that the other layers need like data types, MCU registers, bit math and MCU ports.

3.10 MCU Registers

Modules

• I/O registers

3.10.1 Detailed Description

This contains all the MCU registers definition and description for each register.

3.11 I/O registers 21

3.11 I/O registers

Modules

- Port A registers
- · Port B registers
- · Port C registers
- · Port D registers

3.11.1 Detailed Description

This contains all I/O registers that controls the functionality of the MCU ports.

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. \arglf PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven low (one). If PORTxn is written logic zero when the pin is configured as an out put pin, the port pin is driven low (zero).

3.12 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.

3.12.1 Detailed Description

3.12.2 Macro Definition Documentation

3.12.2.1 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

Definition at line 67 of file RegisterFile.h.

3.12.2.2 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.

Definition at line 75 of file RegisterFile.h.

3.12.2.3 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).

Definition at line 59 of file RegisterFile.h.

3.13 Port B registers 23

3.13 Port B registers

Macros

```
    #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.
```

3.13.1 Detailed Description

3.13.2 Macro Definition Documentation

3.13.2.1 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

Definition at line 101 of file RegisterFile.h.

3.13.2.2 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.

Definition at line 109 of file RegisterFile.h.

3.13.2.3 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).

Definition at line 93 of file RegisterFile.h.

3.14 Port C registers

Macros

```
    #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.

3.14.1 Detailed Description

3.14.2 Macro Definition Documentation

3.14.2.1 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

Definition at line 132 of file RegisterFile.h.

3.15 Port D registers 25

3.14.2.2 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.

Definition at line 140 of file RegisterFile.h.

3.14.2.3 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

Definition at line 124 of file RegisterFile.h.

3.15 Port D registers

Macros

```
• #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.

3.15.1 Detailed Description

3.15.2 Macro Definition Documentation

3.15.2.1 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

Definition at line 163 of file RegisterFile.h.

3.15.2.2 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.

Definition at line 171 of file RegisterFile.h.

3.15.2.3 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

Definition at line 155 of file RegisterFile.h.

Chapter 4

File Documentation

4.1 App/app.c File Reference

4.2 app.c

4.3 App/app.h File Reference

4.4 app.h

00014 #endif /* APP_H_ */

00012

28 File Documentation

4.5 Debug/App/app.d File Reference

4.6 app.d

```
Go to the documentation of this file.
00001 App/app.d App/app.o: ../App/app.d
```

4.7 Debug/ECUAL/Button driver/Button.d File Reference

4.8 Button.d

```
Go to the documentation of this file.
00001 ECUAL/Button driver/Button.d ECUAL/Button driver/Button.o: \
00002 ../ECUAL/Button\ driver/Button.c ../ECUAL/Button\ driver/Button.h \ 00003 ../ECUAL/Button\ driver/../../Service/ATmega32Port.h \
00004 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/DIO.h \
00005 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h \
00006 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h \
00007 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h
80000
       ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h
00009
00010 ../ECUAL/Button\ driver/Button.h:
00012 ../ECUAL/Button\ driver/../../Service/ATmega32Port.h:
00013
00014 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/DIO.h:
00015
00016 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h:
00018 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h:
00019
00020 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/.../../Service/dataTypes.h:
00021
```

4.9 Debug/ECUAL/LED driver/LED.d File Reference

00022 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h:

4.10 LED.d

Go to the documentation of this file.

```
00001 ECUAL/LED driver/LED.d ECUAL/LED driver/LED.o: ../ECUAL/LED driver/LED.c \
00002 ../ECUAL/LED\ driver/LED.h \
00003 ../ECUAL/LED\ driver/../../Service/ATmega32Port.h \
00004 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/DIO.h \
00005 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h \ 00006 ../ECUAL/LED\ driver/.../MCAL/Dio\ driver/.../.Service/BitMath.h \
00007 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h \ 00008 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h
00010 ../ECUAL/LED\ driver/LED.h:
00011
00012 ../ECUAL/LED\ driver/../../Service/ATmega32Port.h:
00013
00014 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/DIO.h:
00015
00016 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h:
00017
00018 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h:
00019
00020 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h:
00022 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h:
```

4.11 Debug/main.d File Reference

4.12 main.d

```
Go to the documentation of this file.

00001 main.d main.o: ././main.c .//./MCAL/Dio\ driver/DIO.h \
00002 ./././MCAL/Dio\ driver/.././Service/ATmega32Port.h \
00003 ./././MCAL/Dio\ driver/.././Service/BitMath.h \
00004 ././.MCAL/Dio\ driver/.././Service/dataTypes.h \
00005 ./././MCAL/Dio\ driver/.././Service/RegisterFile.h
```

```
00006

00007 .././MCAL/Dio\ driver/DIO.h:

00008

00009 .././MCAL/Dio\ driver/../../Service/ATmega32Port.h:

00010

00011 .././MCAL/Dio\ driver/../../Service/BitMath.h:

00012

00013 .././MCAL/Dio\ driver/../../Service/dataTypes.h:

00014
```

00015 .././MCAL/Dio\ driver/../../Service/RegisterFile.h:

4.13 Debug/MCAL/Dio driver/DIO.d File Reference

4.14 DIO.d

Go to the documentation of this file.

```
00001 MCAL/Dio driver/DIO.d MCAL/Dio driver/DIO.c: ../MCAL/Dio\ driver/DIO.c \
00002 ../MCAL/Dio\ driver/DIO.h \
00003 ../MCAL/Dio\ driver/../../Service/ATmega32Port.h \
00004 ../MCAL/Dio\ driver/../../Service/BitMath.h \
00005 ../MCAL/Dio\ driver/../../Service/dataTypes.h \
00006 ../MCAL/Dio\ driver/../../Service/RegisterFile.h
00007
00008 ../MCAL/Dio\ driver/DIO.h:
00009
00010 ../MCAL/Dio\ driver/../../Service/ATmega32Port.h:
00011
00012 ../MCAL/Dio\ driver/../../Service/BitMath.h:
00013
00014 ../MCAL/Dio\ driver/../../Service/dataTypes.h:
00015
00016 ../MCAL/Dio\ driver/../../Service/RegisterFile.h:
```

4.15 ECUAL/Button driver/Button.c File Reference

```
#include "Button.h"
```

Functions

EN_pinErro_t buttonInit (EN_pinNum_t buttonPin)
 initialize the button pin.

• EN_pinErro_t buttonRead (EN_pinNum_t buttonPin, EN_pinState_t *pinState)

reads the value of the button.

30 File Documentation

4.16 Button.c

```
Go to the documentation of this file.
```

```
00001
00002
00003 /*
                                                                   Author : Ehab Omara
00004
                                                                   Date : 8/11/2022 8:25:13 PM
00005 /*
                                                                   File name: Button.c
00006
00007
00008
00009 #include "Button.h"
00010
00011 EN_pinErro_t buttonInit(EN_pinNum_t buttonPin)
00013
          return DIO_pinInit(buttonPin,Input);
00014 }
\texttt{00016} \ \texttt{EN\_pinErro\_t} \ \texttt{buttonRead} \\ (\texttt{EN\_pinNum\_t} \ \texttt{buttonPin,EN\_pinState\_t} \ \star \\ \texttt{pinState})
00017 {
00018
          return DIO_pinRead(buttonPin,pinState);
00019 }
```

4.17 ECUAL/Button driver/Button.h File Reference

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

Functions

- EN_pinErro_t buttonInit (EN_pinNum_t buttonPin)
 initialize the button pin.
- EN_pinErro_t buttonRead (EN_pinNum_t buttonPin, EN_pinState_t *pinState) reads the value of the button.

4.18 Button.h

Go to the documentation of this file.

```
00001
00002 /*
                                                                 Author : Ehab Omara
00003
                                                                  Date : 8/11/2022 8:24:25 PM
00004 /*
                                                                 File name: Button.h
00005
00006
00007 #ifndef BUTTON_H_
00008 #define BUTTON_H_
00009
00010 #include "../../Service/ATmega32Port.h"
00011 #include "../../MCAL/Dio driver/DIO.h
00012
00013
00036 EN_pinErro_t buttonInit(EN_pinNum_t buttonPin);
00037
00050 EN_pinErro_t buttonRead(EN_pinNum_t buttonPin,EN_pinState_t *pinState);
00052 #endif /* BUTTON_H_ */
```

4.19 ECUAL/LED driver/LED.c File Reference

```
#include "LED.h"
```

Functions

- EN_pinErro_t ledInit (EN_pinNum_t ledPin)
 initialize the led pin.
- EN_pinErro_t ledOn (EN_pinNum_t ledPin) turn the led on.
- EN_pinErro_t ledOff (EN_pinNum_t ledPin)
- EN_pinNum_t ledToggle (EN_pinNum_t ledPin) toggle the led state.

4.20 LED.c

```
Go to the documentation of this file.
```

turn the led off.

```
00001
00002 /*
                                            Author : Ehab Omara
                                            Date : 8/12/2022 9:42:19 PM
00003 /*
00004 /*
                                            File name: LED.c
00005
00006 #include "LED.h"
00007
80000
00009
00010 EN_pinErro_t ledInit(EN_pinNum_t ledPin)
00011 {
00012
      return DIO_pinInit(ledPin,Output);
00013 }
00015 EN_pinErro_t ledOn(EN_pinNum_t ledPin)
00016 {
      return DIO_pinWrite(ledPin, High);
00018 }
00020 EN_pinErro_t ledOff(EN_pinNum_t ledPin)
00021 {
00022
      return DIO_pinWrite(ledPin,Low);
00023 }
00025 EN_pinNum_t ledToggle(EN_pinNum_t ledPin)
00026 {
00027
      return DIO_pinToggle(ledPin);
00028 }
```

4.21 ECUAL/LED driver/LED.h File Reference

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

Functions

```
    EN_pinErro_t ledInit (EN_pinNum_t ledPin)
        initialize the led pin.
    EN_pinErro_t ledOn (EN_pinNum_t ledPin)
        turn the led on.
    EN_pinErro_t ledOff (EN_pinNum_t ledPin)
        turn the led off.
    EN_pinNum_t ledToggle (EN_pinNum_t ledPin)
```

toggle the led state.

4.22 LED.h

```
Go to the documentation of this file.
```

```
00001
                    ******************
00002
                                                                Author : Ehab Omara
                                                                      : 8/12/2022 9:42:50 PM
00003 /*
                                                                Date
00004 /*
                                                                File name: LED.h
00006
00007 #ifndef LED_H_
00008 #define LED_H_
00009
00010 #include "../../Service/ATmega32Port.h"
00011 #include "../../MCAL/Dio driver/DIO.h"
00012
00032 EN_pinErro_t ledInit(EN_pinNum_t ledPin);
00033 /**********
00046 EN_pinErro_t ledOn(EN_pinNum_t ledPin);
00060 EN_pinErro_t ledOff(EN_pinNum_t ledPin);
00076 EN_pinNum_t ledToggle(EN_pinNum_t ledPin);
00081 #endif /* LED_H_ */
```

4.23 main.c File Reference

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

Functions

• int main (void)

4.23.1 Function Documentation

4.24 main.c 33

4.23.1.1 main()

```
int main (
     void )
```

Definition at line 9 of file main.c.

4.24 main.c

Go to the documentation of this file.

```
00002 /*
                                                                  Author : Ehab Omara
00003 /*
                                                                         : 8/10/2022 12:00:19 PM
                                                                  Date
00004 /*
00005
00006
00007 #include "./MCAL/Dio driver/DIO.h"
80000
00009 int main(void)
00010 {
         DIO_pinInit (PAO,Output);
00011
00012
         while (1)
00014
             DIO_pinWrite(PA0,Low);
00015
00016
          return 0;
00017 }
00018
```

4.25 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.

4.26 DIO.c

Go to the documentation of this file.

```
00001
                             *******************
00002 /*
                                                                   Author : Ehab Omara
00003
                                                                           : 8/10/2022 3:39:46 PM
                                                                    Date
00004 /*
                                                                    File name: DIO.c
00005
00006
00007 #include "DIO.h"
00008
00009
00010
00011
00012 EN_pinErro_t DIO_pinInit(EN_pinNum_t pinNum, EN_pinDirection_t pinDirection)
00013 {
00014
          EN_pinErro_t error = OK;
          //check if the pin is located in port A
if (pinNum <= PA7)</pre>
00015
00016
00017
00018
              if (pinDirection == Output)
00019
              {
00020
                  setBit(DDRA,pinNum);
00021
00022
              else if (pinDirection == Input)
00023
              {
00024
                  clrBit (DDRA, pinNum);
00025
00026
00027
              {
                  error = WRONG PIN DIR:
00028
00029
00030
00031
          //check if the pin is located in port B
00032
          else if (pinNum <= PB7)</pre>
00033
              pinNum-=PORTB_OFFSET;
00034
              if (pinDirection == Output)
{
00035
00036
00037
                  setBit(DDRB,pinNum);
00038
00039
              else if (pinDirection == Input)
00040
              {
00041
                  clrBit (DDRB,pinNum);
00042
00043
              else
00044
              {
00045
                  error = WRONG_PIN_DIR;
00046
              }
00047
          //check if the pin is located in port C
else if (pinNum <= PC7)</pre>
00048
00049
00050
00051
              pinNum-=PORTC_OFFSET;
00052
              if (pinDirection == Output)
00053
00054
                  setBit(DDRC,pinNum);
00055
00056
              else if (pinDirection == Input)
00057
              {
00058
                  clrBit(DDRC,pinNum);
00059
00060
              else
00061
              {
00062
                  error = WRONG_PIN_DIR;
00063
00064
          //check if the pin is located in port D
00065
          else if (pinNum <= PD7)
00066
00067
00068
              pinNum-=PORTD_OFFSET;
00069
              if (pinDirection == Output)
00070
              {
00071
                  setBit(DDRD,pinNum);
00072
00073
              else if (pinDirection == Input)
00074
00075
                  clrBit (DDRD,pinNum);
00076
00077
              else
```

4.26 DIO.c 35

```
{
00079
                   error = WRONG_PIN_DIR;
00080
00081
           //{\rm if} the pinNum is wrong
00082
00083
          else
00085
              error = WRONG_PIN_NUM;
00086
00087
           return error;
00088 }
00089
00090 EN_pinErro_t DIO_pinWrite(EN_pinNum_t pinNum,EN_pinState_t pinState)
00091 {
00092
           EN_pinErro_t error = OK;
          //check if the pin is located in port A
if (pinNum <= PA7)</pre>
00093
00094
00095
00096
               if (pinState == High)
00097
               {
00098
                   setBit(PORTA, pinNum);
00099
00100
               else if (pinState == Low)
00101
              {
00102
                   clrBit (PORTA, pinNum);
00103
00104
               else
00105
               {
00106
                   error = WRONG PIN STATE:
00107
00108
00109
           //check if the pin is located in port {\tt B}
00110
           else if (pinNum <= PB7)</pre>
00111
               pinNum-=PORTB_OFFSET;
00112
               if (pinState == High)
{
00113
00114
00115
                   setBit(PORTB,pinNum);
00116
00117
               else if (pinState == Low)
00118
              {
00119
                   clrBit (PORTB, pinNum);
00120
00121
               else
00122
               {
00123
                   error = WRONG_PIN_STATE;
00124
               }
00125
00126
           //check if the pin is located in port C
           else if (pinNum <= PC7)</pre>
00127
00128
00129
               if (pinState == High)
00130
               {
                   setBit (PORTC, pinNum);
00131
00132
               else if (pinState == Low)
00134
              {
00135
                   clrBit (PORTC, pinNum);
00136
              }
00137
               else
00138
               {
00139
                   error = WRONG_PIN_STATE;
00140
00141
00142
           //check if the pin is located in port {\tt D}
           else if (pinNum <= PD7)</pre>
00143
00144
00145
               if (pinState == High)
00146
              {
00147
                   setBit(PORTD,pinNum);
00148
00149
               else if (pinState == Low)
00150
                   clrBit (PORTD, pinNum);
00151
00152
00153
00154
00155
                   error = WRONG_PIN_STATE;
               }
00156
00157
00158
           //if the pinNum is wrong
00159
           else
00160
00161
               error = WRONG_PIN_NUM;
00162
00163
           return error:
```

```
00164 }
00165
00166 EN_pinErro_t DIO_pinRead(EN_pinNum_t pinNum,EN_pinState_t *pinState)
00167 {
00168
          EN pinErro t error = OK:
          //check if the pin is located in port A
00169
00170
           if (pinNum <= PA7)</pre>
00171
00172
               *pinState = getBit(PINA,pinNum);
00173
          //check if the pin is located in port B else if (pinNum <= PB7)
00174
00175
00176
00177
               pinNum-=PORTB_OFFSET;
00178
               *pinState = getBit(PINB,pinNum);
00179
00180
          //check if the pin is located in port C
          else if (pinNum <= PC7)</pre>
00181
00182
          {
00183
               *pinState = getBit(PINC,pinNum);
00184
          //check if the pin is located in port {\tt D}
00185
00186
          else if (pinNum <= PD7)</pre>
00187
00188
               *pinState = getBit(PIND,pinNum);
00189
00190
          //if the pinNum is wrong
00191
          else
00192
          {
00193
               error = WRONG PIN NUM:
00194
00195
          return error;
00196 }
00197
00198 EN_pinErro_t DIO_pinToggle(EN_pinNum_t pinNum)
00199 {
00200
          EN_pinErro_t error = OK;
00201
          //check if the pin is located in port A
00202
          if (pinNum <= PA7)</pre>
00203
          {
              toggleBit (PORTA, pinNum);
00204
00205
00206
          //check if the pin is located in port B
00207
          else if (pinNum <= PB7)</pre>
00208
              pinNum-=PORTB_OFFSET;
toggleBit(PORTB,pinNum);
00209
00210
00211
00212
          //check if the pin is located in port C
00213
          else if (pinNum <= PC7)
00214
00215
              toggleBit (PORTC, pinNum);
00216
00217
          //check if the pin is located in port D
00218
          else if (pinNum <= PD7)
00219
          {
00220
              toggleBit (PORTD, pinNum);
00221
          //if the pinNum is wrong
00222
00223
          else
00224
          {
00225
              error = WRONG_PIN_NUM;
00226
          return error;
00227
00228 }
00229
```

4.27 MCAL/Dio driver/DIO.h File Reference

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

4.28 DIO.h 37

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.

4.27.1 Detailed Description

Author

: Ehab Omara

Date

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

Definition in file DIO.h.

4.28 DIO.h

Go to the documentation of this file.

4.29 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 }
```

4.30 ATmega32Port.h

Go to the documentation of this file.

00062 #define PORTA_OFFSET

```
00001
         ************************************
00002
                                                     Author : Ehab Omara
00003 /*
                                                      Date : 8/10/2022 3:49:55 PM
00004 /*
                                                      File name: ATmega32Port.h
00005
     00006
00007 #ifndef ATMEGA32PORT_H_
00008 #define ATMEGA32PORT_H_
00022 typedef enum
00023 {
00024
        /*PORTA pins*/
00025
        PAO,
00026
        PA1,
00027
        PA2,
00028
        PA3,
00029
        PA4,
00030
        PA5,
        PA6,
00031
        PA7,
00033
        /*PORTB pins*/
00034
        PBO,
00035
        PB1,
00036
        PB2,
00037
        PB3,
00038
        PB4,
00039
        PB5,
00040
        PB6,
00041
        PB7
        /*PORTC pins*/
00042
        PCO,
00043
00044
        PC1,
00045
        PC2,
00046
        PC3,
00047
        PC4,
00048
00049
        PC5,
        PC6,
00050
        PC7
00051
        /*PORTD pins*/
00052
        PD0,
00053
        PD1,
00054
        PD2,
        PD3,
00055
00056
        PD4,
00057
        PD5,
00058
        PD6,
00059
        PD7
00060 }EN_pinNum_t;
00061
```

```
00063 #define PORTB_OFFSET
00064 #define PORTC_OFFSET
00065 #define PORTD_OFFSET
00067 typedef enum
00068 {
          Low,
00069
00070
         High
00071 }EN_pinState_t;
00072 typedef enum
00073 {
00074
          Input,
00075
          Output
00076 }EN_pinDirection_t;
00077 typedef enum
00078 {
00079
         WRONG_PIN_NUM,
00080
        WRONG_PIN_DIR,
WRONG_PIN_STATE
00081
00082
00083 }EN_pinErro_t;
00087 #endif /* ATMEGA32PORT_H_ */
```

4.31 Service/BitMath.h File Reference

Macros

```
    #define setBit(reg, bitNum) reg |= (1<<bitNum)
        this Macro writes 1 to the bit.</li>
    #define clrBit(reg, bitNum) reg &= (~(1<<bitNum))
        this Macro clear the bit.</li>
    #define toggleBit(reg, bitNum) reg ^= (1<<bitNum)
        This Macro toggle the bit logic.</li>
    #define getBit(reg, bitNum) ((reg>>bitNum) & 0x01)
```

This Macro read this bit value.

4.32 BitMath.h

Go to the documentation of this file.

4.33 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

4.34 dataTypes.h

```
Go to the documentation of this file.
```

```
00002
                                                                   Author : Ehab Omara
00003
                                                                          : 8/10/2022 12:06:28 PM
00004
                                                                   File name: dataTypes.h
00005
00006
00007 #ifndef DATATYPES_H_
00008 #define DATATYPES_H_
                                  uint8 t:
00015 typedef unsigned char
00016 typedef signed char
                                   sint8_t;
00017 typedef unsigned short int uint16_t;
00018 typedef signed short int
                                 uint32_t;
00019 typedef unsigned long int
00020 typedef signed long int
00021 typedef float
                                  sint32_t;
                                  float32_t;
00022 typedef double
                                   float64 t;
                                  float128_t;
00023 typedef long double
00027 #endif /* DATATYPES_H_ */
```

4.35 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.

4.36 RegisterFile.h

4.36 RegisterFile.h

Go to the documentation of this file.

```
************************************
00002 /*
                                                       Author : Ehab Omara
00003 /*
                                                       Date : 8/10/2022 12:06:56 PM
00004 /*
                                                       File name: RegisterFile.h
00005
00007 #ifndef REGISTERFILE_H_
00008 #define REGISTERFILE_H_
00009
00010 /*
00011 \star if the DDRx is set to be output and we write High to the PORTx 00012 \star this will activate the internal Pull up resistor.
00013 */
00014
00045 /******* Port A registers
     *******************
00059 #define PORTA (*((volatile uint8_t*)0x3B)) //1->high output 0->low output 00067 #define DDRA (*((volatile uint8_t*)0x3A)) //1->to make it output 0->to make it 00075 #define PINA (*((volatile uint8_t*)0x39)) //this register to read a value from a pin
                                                                    0->low output
                                                                    0->to make it input
*******************

      00124 #define PORTC
      (*((volatile uint8_t*)0x35))

      00132 #define DDRC
      (*((volatile uint8_t*)0x34))

      00140 #define PINC
      (*((volatile uint8_t*)0x33))

00142 /****** Port D registers
00175 #endif /* REGISTERFILE_H_ */
```

Index

App/app.c, 27	DIO driver, 10
App/app.h, 27	DIO_pinToggle DIO driver, 11
Bit math, 15	DIO_pinWrite
clrBit, 16	DIO driver, 11
getBit, 16	Dio diivoi, Ti
setBit, 17	ECUAL layer, 5
toggleBit, 17	ECUAL/Button driver/Button.c, 29, 30
Button driver, 5	ECUAL/Button driver/Button.h, 30
buttonInit, 5	ECUAL/LED driver/LED.c, 31
buttonRead, 6	ECUAL/LED driver/LED.h, 31, 32
buttonInit	EN_pinDirection_t
Button driver, 5	MCU ports, 13
buttonRead	EN_pinErro_t
Button driver, 6	MCU ports, 14
	EN_pinNum_t
clrBit	MCU ports, 14
Bit math, 16	EN_pinState_t
DDRA	MCU ports, 15
Port A registers, 21	float128 t
DDRB	Definition of data types, 18
Port B registers, 23	float32 t
DDRC	Definition of data types, 18
Port C registers, 24	float64_t
DDRD	Definition of data types, 18
Port D registers, 25	Dominion of data types, 10
Debug/App/app.d, 28	getBit
Debug/ECUAL/Button driver/Button.d, 28	Bit math, 16
Debug/ECUAL/LED driver/LED.d, 28	
Debug/main.d, 29	High
Debug/MCAL/Dio driver/DIO.d, 29	MCU ports, 15
Definition of data types, 18	I/O va sistava O1
float128_t, 18	I/O registers, 21
float32_t, 18	Input MCII porto 14
float64_t, 18	MCU ports, 14
sint16_t, 19	LED driver, 6
sint32_t, 19	ledInit, 7
sint8_t, 19	ledOff, 7
uint16_t, 19	ledOn, 8
uint32_t, 19	ledToggle, 8
uint8_t, 20	ledInit
DIO driver, 9	LED driver, 7
DIO_pinInit, 10	ledOff
DIO_pinRead, 10	LED driver, 7
DIO_pinToggle, 11	ledOn
DIO_pinWrite, 11	LED driver, 8
DIO_pinInit	ledToggle
DIO driver, 10	LED driver, 8
DIO_pinRead	Low

44 INDEX

MOU.	01/
MCU ports, 15	OK MCI I nowto 14
main	MCU ports, 14
main.c, 32	Output MCU ports, 14
main.c, 32	MOO ports, 14
main, 32	PA0
MCAL layer, 9	MCU ports, 14
MCAL/Dio driver/DIO.c, 33, 34	PA1
MCAL/Dio driver/DIO.h, 36, 37	MCU ports, 14
MCU ports, 12	PA2
EN_pinDirection_t, 13	MCU ports, 14
EN_pinErro_t, 14	PA3
EN_pinNum_t, 14	MCU ports, 14
EN_pinState_t, 15	PA4
High, 15	MCU ports, 14
Input, 14	PA5
Low, 15	MCU ports, 14
OK, 14	PA6
Output, 14	MCU ports, 14
PA0, 14	PA7
PA1, 14	MCU ports, 14
PA2, 14	PB0
PA3, 14 PA4, 14	MCU ports, 14
PA4, 14 PA5, 14	PB1
PA6, 14	MCU ports, 14 PB2
PA7, 14	MCU ports, 14
PB0, 14	PB3
PB1, 14	MCU ports, 14
PB2, 14	PB4
PB3, 14	MCU ports, 14
PB4, 14	PB5
PB5, 14	MCU ports, 14
PB6, 14	PB6
PB7, 14	MCU ports, 14
PC0, 15	PB7
PC1, 15	MCU ports, 14
PC2, 15	PC0
PC3, 15	MCU ports, 15
PC4, 15	PC1
PC5, 15	MCU ports, 15
PC6, 15	PC2
PC7, 15	MCU ports, 15
PD0, 15	PC3
PD1, 15 PD2, 15	MCU ports, 15 PC4
PD3, 15	MCU ports, 15
PD4, 15	PC5
PD5, 15	MCU ports, 15
PD6, 15	PC6
PD7, 15	MCU ports, 15
PORTA OFFSET, 13	PC7
PORTB OFFSET, 13	MCU ports, 15
PORTC_OFFSET, 13	PD0
PORTD_OFFSET, 13	MCU ports, 15
WRONG_PIN_DIR, 14	PD1
WRONG_PIN_NUM, 14	MCU ports, 15
WRONG_PIN_STATE, 14	PD2
MCU Registers, 20	MCU ports, 15

INDEX 45

PD3	Definition of data types, 19
MCU ports, 15	sint32_t
PD4	Definition of data types, 19
MCU ports, 15	sint8_t
PD5	Definition of data types, 19
MCU ports, 15	
PD6	toggleBit
MCU ports, 15	Bit math, 17
PD7	
MCU ports, 15	uint16_t
PINA	Definition of data types, 19
Port A registers, 22	uint32_t
PINB	Definition of data types, 19
Port B registers, 23	uint8_t
PINC	Definition of data types, 20
Port C registers, 24	WEONG DIN DID
PIND	WRONG_PIN_DIR
Port D registers, 26	MCU ports, 14
Port A registers, 21	WRONG_PIN_NUM
DDRA, 21	MCU ports, 14
PINA, 22	WRONG_PIN_STATE MCU ports, 14
PORTA, 22	MCO ports, 14
Port B registers, 23	
DDRB, 23	
PINB, 23	
PORTB, 23	
Port C registers, 24	
DDRC, 24	
PINC, 24	
PORTC, 25	
Port D registers, 25	
DDRD, 25	
PIND, 26	
PORTD, 26	
PORTA	
Port A registers, 22	
PORTA_OFFSET	
MCU ports, 13	
PORTB	
Port B registers, 23	
PORTB_OFFSET	
MCU ports, 13	
PORTC	
Port C registers, 25	
PORTC_OFFSET	
MCU ports, 13	
PORTD	
Port D registers, 26	
PORTD_OFFSET	
MCU ports, 13	
Service layer, 20	
Service/ATmega32Port.h, 37, 38	
Service/BitMath.h, 39	
Service/dataTypes.h, 39, 40	
Service/RegisterFile.h, 40, 41	
setBit	
Bit math, 17	
sint16 t	