

## On-Demand Traffic Control

Generated by Doxygen 1.9.4



<b>1 Module Index</b>	<b>1</b>
1.1 Modules	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Module Documentation</b>	<b>5</b>
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 Interrupts driver	12
3.6.1 Detailed Description	12
3.7 ATMEGA32 interrupts definitions	12
3.7.1 Detailed Description	13
3.7.2 Macro Definition Documentation	13
3.7.2.1 ADC	13
3.7.2.2 ANA_COMP	13
3.7.2.3 cli	14
3.7.2.4 EE_RDY	14
3.7.2.5 EXT_INT0	14
3.7.2.6 EXT_INT1	14
3.7.2.7 EXT_INT2	15
3.7.2.8 ISR	15

3.7.2.9 sei	15
3.7.2.10 SPI_STC	16
3.7.2.11 SPM_RDY	16
3.7.2.12 TIM0_COMP	16
3.7.2.13 TIM0_OVF	16
3.7.2.14 TIM1_CAPT	16
3.7.2.15 TIM1_COMPA	17
3.7.2.16 TIM1_COMPB	17
3.7.2.17 TIM1_OVF	17
3.7.2.18 TIM2_COMP	17
3.7.2.19 TIM2_OVF	17
3.7.2.20 TWI	18
3.7.2.21 USART_RXC	18
3.7.2.22 USART_TXC	18
3.7.2.23 USART_UDRE	18
3.8 MCU ports	18
3.8.1 Detailed Description	19
3.8.2 Macro Definition Documentation	19
3.8.2.1 PORTA_OFFSET	19
3.8.2.2 PORTB_OFFSET	19
3.8.2.3 PORTC_OFFSET	19
3.8.2.4 PORTD_OFFSET	20
3.8.3 Enumeration Type Documentation	20
3.8.3.1 EN_pinDirection_t	20
3.8.3.2 EN_pinErro_t	20
3.8.3.3 EN_pinNum_t	20
3.8.3.4 EN_pinState_t	21
3.9 Bit math	22
3.9.1 Detailed Description	22
3.9.2 Macro Definition Documentation	22
3.9.2.1 clrBit	22
3.9.2.2 getBit	23
3.9.2.3 setBit	23
3.9.2.4 toggleBit	24
3.10 Definition of data types	24
3.10.1 Detailed Description	24
3.10.2 Typedef Documentation	25
3.10.2.1 float128_t	25
3.10.2.2 float32_t	25
3.10.2.3 float64_t	25
3.10.2.4 sint16_t	25
3.10.2.5 sint32_t	25

3.10.2.6 sint8_t . . . . .	26
3.10.2.7 uint16_tt . . . . .	26
3.10.2.8 uint32_t . . . . .	26
3.10.2.9 uint8_t . . . . .	26
3.11 Service layer . . . . .	26
3.11.1 Detailed Description . . . . .	27
3.12 MCU Registers . . . . .	27
3.12.1 Detailed Description . . . . .	27
3.13 I/O registers . . . . .	27
3.13.1 Detailed Description . . . . .	27
3.14 Port A registers . . . . .	28
3.14.1 Detailed Description . . . . .	28
3.14.2 Macro Definition Documentation . . . . .	28
3.14.2.1 DDRA . . . . .	28
3.14.2.2 PINA . . . . .	28
3.14.2.3 PORTA . . . . .	29
3.15 Port B registers . . . . .	29
3.15.1 Detailed Description . . . . .	29
3.15.2 Macro Definition Documentation . . . . .	29
3.15.2.1 DDRB . . . . .	29
3.15.2.2 PINB . . . . .	30
3.15.2.3 PORTB . . . . .	30
3.16 Port C registers . . . . .	30
3.16.1 Detailed Description . . . . .	30
3.16.2 Macro Definition Documentation . . . . .	30
3.16.2.1 DDRC . . . . .	31
3.16.2.2 PINC . . . . .	31
3.16.2.3 PORTC . . . . .	31
3.17 Port D registers . . . . .	32
3.17.1 Detailed Description . . . . .	32
3.17.2 Macro Definition Documentation . . . . .	32
3.17.2.1 DDRD . . . . .	32
3.17.2.2 PIND . . . . .	32
3.17.2.3 PORTD . . . . .	33
3.18 Interrupt registers . . . . .	33
3.18.1 Detailed Description . . . . .	33
3.18.2 Macro Definition Documentation . . . . .	33
3.18.2.1 GICR . . . . .	33
3.18.2.2 GIFR . . . . .	34
3.18.2.3 MCUCR . . . . .	34
3.18.2.4 MCUCSR . . . . .	35

<b>4 File Documentation</b>	<b>37</b>
4.1 App/app.c File Reference	37
4.2 app.c	37
4.3 App/app.h File Reference	37
4.4 app.h	37
4.5 Debug/App/app.d File Reference	38
4.6 app.d	38
4.7 Debug/ECUAL/Button driver/Button.d File Reference	38
4.8 Button.d	38
4.9 Debug/ECUAL/LED driver/LED.d File Reference	38
4.10 LED.d	38
4.11 Debug/main.d File Reference	39
4.12 main.d	39
4.13 Debug/MCAL/Dio driver/DIO.d File Reference	39
4.14 DIO.d	39
4.15 ECUAL/Button driver/Button.c File Reference	40
4.16 Button.c	40
4.17 ECUAL/Button driver/Button.h File Reference	40
4.18 Button.h	41
4.19 ECUAL/LED driver/LED.c File Reference	41
4.20 LED.c	41
4.21 ECUAL/LED driver/LED.h File Reference	42
4.22 LED.h	42
4.23 main.c File Reference	43
4.23.1 Function Documentation	43
4.23.1.1 main()	43
4.24 main.c	43
4.25 MCAL/Dio driver/DIO.c File Reference	43
4.26 DIO.c	44
4.27 MCAL/Dio driver/DIO.h File Reference	47
4.27.1 Detailed Description	47
4.28 DIO.h	47
4.29 MCAL/Ext interrupt driver/Ext interrupt.c File Reference	48
4.30 Ext interrupt.c	48
4.31 MCAL/Ext interrupt driver/Ext interrupt.h File Reference	48
4.31.1 Macro Definition Documentation	48
4.31.1.1 EXT	48
4.32 Ext interrupt.h	48
4.33 MCAL/Interrupt/Interrupt.h File Reference	49
4.34 Interrupt.h	49
4.35 Service/ATmega32Port.h File Reference	50
4.36 ATmega32Port.h	50

---

4.37 Service/BitMath.h File Reference . . . . .	51
4.38 BitMath.h . . . . .	51
4.39 Service/dataTypes.h File Reference . . . . .	52
4.40 dataType.h . . . . .	52
4.41 Service/RegisterFile.h File Reference . . . . .	52
4.42 RegisterFile.h . . . . .	53
<b>Index</b>	<b>55</b>





# 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
Interrupts driver . . . . .	12
ATMEGA32 interrupts definitions . . . . .	12
Service layer . . . . .	26
MCU ports . . . . .	18
Bit math . . . . .	22
Definition of data types . . . . .	24
MCU Registers . . . . .	27
I/O registers . . . . .	27
Port A registers . . . . .	28
Port B registers . . . . .	29
Port C registers . . . . .	30
Port D registers . . . . .	32
Interrupt registers . . . . .	33



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">main.c</a>	43
App/ <a href="#">app.c</a>	37
App/ <a href="#">app.h</a>	37
Debug/ <a href="#">main.d</a>	39
Debug/App/ <a href="#">app.d</a>	38
Debug/ECUAL/Button driver/ <a href="#">Button.d</a>	38
Debug/ECUAL/LED driver/ <a href="#">LED.d</a>	38
Debug/MCAL/Dio driver/ <a href="#">DIO.d</a>	39
ECUAL/Button driver/ <a href="#">Button.c</a>	40
ECUAL/Button driver/ <a href="#">Button.h</a>	40
ECUAL/LED driver/ <a href="#">LED.c</a>	41
ECUAL/LED driver/ <a href="#">LED.h</a>	42
MCAL/Dio driver/ <a href="#">DIO.c</a>	43
MCAL/Dio driver/ <a href="#">DIO.h</a>	47
MCAL/Ext interrupt driver/ <a href="#">Ext interrupt.c</a>	48
MCAL/Ext interrupt driver/ <a href="#">Ext interrupt.h</a>	48
MCAL/Interrupt/ <a href="#">Interrupt.h</a>	49
Service/ <a href="#">ATmega32Port.h</a>	50
Service/ <a href="#">BitMath.h</a>	51
Service/ <a href="#">dataTypes.h</a>	52
Service/ <a href="#">RegisterFile.h</a>	52



## 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()

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

initialize the button pin.

[buttonInit](#) function:

- This function makes the button pin as Input.

**Parameters**

in	<i>buttonPin</i>	it is the pin which the button is connected to,it may be (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.

Definition at line 11 of file [Button.c](#).

**3.2.2.2 buttonRead()**

```
EN_pinErro_t buttonRead (
    EN_pinNum_t buttonPin,
    EN_pinState_t * pinState )
```

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

**Return values**

<i>WRONG_PIN_NUM</i>	if the pinNum is wrong.
<i>OK</i>	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)

- 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 ledInit()

```
EN_pinErro_t ledInit (
    EN_pinNum_t ledPin )
```

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	<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.

Definition at line 10 of file [LED.c](#).

#### 3.3.2.2 ledOff()

```
EN_pinErro_t ledOff (
    EN_pinNum_t ledPin )
```

turn the led off.

**ledOff** function:

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

#### Parameters

in	<i>ledPin</i>	it is the pin which the led is connected to,it may be (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.

Definition at line 20 of file [LED.c](#).

### 3.3.2.3 ledOn()

```
EN_pinErro_t ledOn (
    EN_pinNum_t ledPin )
```

turn the led on.

**ledOn** function:

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

#### Parameters

in	<i>ledPin</i>	it is the pin which the led is connected to,it may be (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.

Definition at line 15 of file [LED.c](#).

### 3.3.2.4 ledToggle()

```
EN_pinNum_t ledToggle (
```



```
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	<i>ledPin</i>	it is the pin which the led is connected to,it may be (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.

Definition at line 25 of file [LED.c](#).

## 3.4 MCAL layer

### Modules

- [DIO driver](#)
- [Interrupts 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()

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

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	<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.

Definition at line 12 of file [DIO.c](#).

#### 3.5.2.2 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.

Definition at line 166 of file [DIO.c](#).

### 3.5.2.3 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.

Definition at line 198 of file [DIO.c](#).

### 3.5.2.4 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.

Definition at line 90 of file [DIO.c](#).

## 3.6 Interrupts driver

### Modules

- [ATMEGA32 interrupts definitions](#)  
*Interrupts request handlers.*

### 3.6.1 Detailed Description

## 3.7 ATMEGA32 interrupts definitions

Interrupts request handlers.

### Macros

- #define [sei\(\)](#) \_\_asm\_\_ \_\_volatile\_\_ ("sei" ::: "memory")
- #define [cli\(\)](#) \_\_asm\_\_ \_\_volatile\_\_ ("cli" ::: "memory")
- #define [EXT\\_INT0](#) \_\_vector\_1
- #define [EXT\\_INT1](#) \_\_vector\_2
- #define [EXT\\_INT2](#) \_\_vector\_3
- #define [TIM2\\_COMP](#) \_\_vector\_4
- #define [TIM2\\_OVF](#) \_\_vector\_5
- #define [TIM1\\_CAPT](#) \_\_vector\_6
- #define [TIM1\\_COMPA](#) \_\_vector\_7
- #define [TIM1\\_COMPB](#) \_\_vector\_8

- `#define TIM1_OVF __vector_9`
- `#define TIM0_COMP __vector_10`
- `#define TIM0_OVF __vector_11`
- `#define SPI_STC __vector_12`
- `#define USART_RXC __vector_13`
- `#define USART_UDRE __vector_14`
- `#define USART_TXC __vector_15`
- `#define ADC __vector_16`
- `#define EE_RDY __vector_17`
- `#define ANA_COMP __vector_18`
- `#define TWI __vector_19`
- `#define SPM_RDY __vector_20`
- `#define ISR(INT_VECT)`

*interrupt service routine Macro.*

### 3.7.1 Detailed Description

Interrupts request handlers.

#### This section contains:

- Macros for Interrupts request handlers in ATmega32.
- Macros for enabling and disabling global interrupt.
- ISR Macro which defines interrupt service routine function.

### 3.7.2 Macro Definition Documentation

#### 3.7.2.1 ADC

```
#define ADC __vector_16
```

This Macro defines ADC Conversion Complete Handler

Definition at line 63 of file [Interrupt.h](#).

#### 3.7.2.2 ANA\_COMP

```
#define ANA_COMP __vector_18
```

This Macro defines Analog Comparator Handler

Definition at line 65 of file [Interrupt.h](#).

### 3.7.2.3 cli

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

- Disables all interrupts by clearing the global interrupt mask.
- This function actually compiles into a single line of assembly, so there is no function call overhead.
- However, the macro also implies a **memory barrier** which can cause additional loss of optimization.

Definition at line 46 of file [Interrupt.h](#).

### 3.7.2.4 EE\_RDY

```
#define EE_RDY __vector_17
```

This Macro defines EEPROM Ready Handler

Definition at line 64 of file [Interrupt.h](#).

### 3.7.2.5 EXT\_INT0

```
#define EXT_INT0 __vector_1
```

This Macro defines IRQ0 Handler

Definition at line 48 of file [Interrupt.h](#).

### 3.7.2.6 EXT\_INT1

```
#define EXT_INT1 __vector_2
```

This Macro defines IRQ1 Handler

Definition at line 49 of file [Interrupt.h](#).

### 3.7.2.7 EXT\_INT2

```
#define EXT_INT2 __vector_3
```

This Macro defines IRQ2 Handler

Definition at line 50 of file [Interrupt.h](#).

### 3.7.2.8 ISR

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

**Value:**

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

interrupt service routine Macro.

- Introduces an interrupt handler function (interrupt service routine) that runs with global interrupts initially disabled by default with no attributes specified.

**Precondition**

`vector` must be one of the interrupt vector names that are valid for the particular MCU type.

Definition at line 78 of file [Interrupt.h](#).

### 3.7.2.9 sei

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

- Disables all interrupts by clearing the global interrupt mask.
- This function actually compiles into a single line of assembly, so there is no function call overhead.
- However, the macro also implies a **memory barrier** which can cause additional loss of optimization.

Definition at line 35 of file [Interrupt.h](#).

### 3.7.2.10 SPI\_STC

```
#define SPI_STC __vector_12
```

This Macro defines SPI Transfer Complete Handler

Definition at line 59 of file [Interrupt.h](#).

### 3.7.2.11 SPM\_RDY

```
#define SPM_RDY __vector_20
```

This Macro defines Store Program Memory Ready Handler

Definition at line 67 of file [Interrupt.h](#).

### 3.7.2.12 TIM0\_COMP

```
#define TIM0_COMP __vector_10
```

This Macro defines Timer0 Compare Handler

Definition at line 57 of file [Interrupt.h](#).

### 3.7.2.13 TIM0\_OVF

```
#define TIM0_OVF __vector_11
```

This Macro defines Timer0 Overflow Handler

Definition at line 58 of file [Interrupt.h](#).

### 3.7.2.14 TIM1\_CAPT

```
#define TIM1_CAPT __vector_6
```

This Macro defines Timer1 Capture Handler

Definition at line 53 of file [Interrupt.h](#).



#### 3.7.2.15 TIM1\_COMPA

```
#define TIM1_COMPA __vector_7
```

This Macro defines Timer1 CompareA Handler

Definition at line 54 of file [Interrupt.h](#).

#### 3.7.2.16 TIM1\_COMPB

```
#define TIM1_COMPB __vector_8
```

This Macro defines Timer1 CompareB Handler

Definition at line 55 of file [Interrupt.h](#).

#### 3.7.2.17 TIM1\_OVF

```
#define TIM1_OVF __vector_9
```

This Macro defines Timer1 Overflow Handler

Definition at line 56 of file [Interrupt.h](#).

#### 3.7.2.18 TIM2\_COMP

```
#define TIM2_COMP __vector_4
```

This Macro defines Timer2 Compare Handler

Definition at line 51 of file [Interrupt.h](#).

#### 3.7.2.19 TIM2\_OVF

```
#define TIM2_OVF __vector_5
```

This Macro defines Timer2 Overflow Handler

Definition at line 52 of file [Interrupt.h](#).

### 3.7.2.20 TWI

```
#define TWI __vector_19
```

This Macro defines Two-wire Serial Interface Handler

Definition at line 66 of file [Interrupt.h](#).

### 3.7.2.21 USART\_RXC

```
#define USART_RXC __vector_13
```

This Macro defines USART RX Complete Handler

Definition at line 60 of file [Interrupt.h](#).

### 3.7.2.22 USART\_TXC

```
#define USART_TXC __vector_15
```

This Macro defines USART TX Complete Handler

Definition at line 62 of file [Interrupt.h](#).

### 3.7.2.23 USART\_UDRE

```
#define USART_UDRE __vector_14
```

This Macro defines UDR Empty Handler

Definition at line 61 of file [Interrupt.h](#).

## 3.8 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` }

### 3.8.1 Detailed Description

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

### 3.8.2 Macro Definition Documentation

#### 3.8.2.1 PORTA\_OFFSET

```
#define PORTA_OFFSET 0
```

This macro defines the start of the PORTA pins

Definition at line 62 of file [ATmega32Port.h](#).

#### 3.8.2.2 PORTB\_OFFSET

```
#define PORTB_OFFSET 8
```

This macro defines the start of the PORTB pins

Definition at line 63 of file [ATmega32Port.h](#).

#### 3.8.2.3 PORTC\_OFFSET

```
#define PORTC_OFFSET 16
```

This macro defines the start of the PORTC pins

Definition at line 64 of file [ATmega32Port.h](#).

### 3.8.2.4 PORTD\_OFFSET

```
#define PORTD_OFFSET 24
```

This macro defines the start of the PORTD pins

Definition at line 65 of file [ATmega32Port.h](#).

## 3.8.3 Enumeration Type Documentation

### 3.8.3.1 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](#).

### 3.8.3.2 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](#).

### 3.8.3.3 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
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](#).

### 3.8.3.4 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](#).

## 3.9 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.*

### 3.9.1 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.

### 3.9.2 Macro Definition Documentation

#### 3.9.2.1 clrBit

```
#define clrBit(  
    reg,  
    bitNum ) reg &= (~(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	<i>reg</i>	this is register that needed to be changed.
in	<i>bitNum</i>	this is bit number that needed to be written to 0 in the register.

Definition at line 37 of file [BitMath.h](#).

### 3.9.2.2 getBit

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

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.

#### Parameters

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

Definition at line 62 of file [BitMath.h](#).

### 3.9.2.3 setBit

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

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

Definition at line 26 of file [BitMath.h](#).

### 3.9.2.4 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	<i>reg</i>	this is register that needed to be changed.
in	<i>bitNum</i>	this is bit number that needed to be changed in the register.

Definition at line 50 of file [BitMath.h](#).

## 3.10 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](#)

### 3.10.1 Detailed Description

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



## 3.10.2 Typedef Documentation

### 3.10.2.1 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](#).

### 3.10.2.2 float32\_t

```
typedef float float32_t
```

This is define a memory size of 4 byte float

Definition at line 21 of file [dataTypes.h](#).

### 3.10.2.3 float64\_t

```
typedef double float64_t
```

This is define a memory size of 8 byte float

Definition at line 22 of file [dataTypes.h](#).

### 3.10.2.4 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](#).

### 3.10.2.5 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](#).

### 3.10.2.6 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](#).

### 3.10.2.7 uint16\_t

```
typedef unsigned short int uint16_t
```

This is define a memory size of 2 byte

Definition at line 17 of file [dataTypes.h](#).

### 3.10.2.8 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](#).

### 3.10.2.9 uint8\_t

```
typedef unsigned char uint8_t
```

This is define a memory size of 1 byte

Definition at line 15 of file [dataTypes.h](#).

## 3.11 Service layer

### Modules

- [MCU ports](#)
- [Bit math](#)
- [Definition of data types](#)
- [MCU Registers](#)

### 3.11.1 Detailed Description

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

## 3.12 MCU Registers

### Modules

- [I/O registers](#)
- [Interrupt registers](#)

### 3.12.1 Detailed Description

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

## 3.13 I/O registers

### Modules

- [Port A registers](#)
- [Port B registers](#)
- [Port C registers](#)
- [Port D registers](#)

### 3.13.1 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. If 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 output pin, the port pin is driven low (zero).

## 3.14 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.*

### 3.14.1 Detailed Description

### 3.14.2 Macro Definition Documentation

#### 3.14.2.1 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](#).

#### 3.14.2.2 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](#).

### 3.14.2.3 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](#).

## 3.15 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.*

### 3.15.1 Detailed Description

### 3.15.2 Macro Definition Documentation

#### 3.15.2.1 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](#).

### 3.15.2.2 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](#).

### 3.15.2.3 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](#).

## 3.16 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.*

### 3.16.1 Detailed Description

### 3.16.2 Macro Definition Documentation

### 3.16.2.1 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](#).

### 3.16.2.2 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](#).

### 3.16.2.3 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](#).

## 3.17 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.*

### 3.17.1 Detailed Description

### 3.17.2 Macro Definition Documentation

#### 3.17.2.1 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](#).

#### 3.17.2.2 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](#).



### 3.17.2.3 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](#).

## 3.18 Interrupt registers

### Macros

- `#define GICR (*((volatile uint8_t*)0x5B))`  
*General Interrupt Control Register.*
- `#define GIFR (*((volatile uint8_t*)0x5A))`  
*General Interrupt Flag Register.*
- `#define MCUOCR (*((volatile uint8_t*)0x55))`  
*MCU Control Register.*
- `#define MCUCSR (*((volatile uint8_t*)0x54))`  
*MCU Control and Status Register.*

### 3.18.1 Detailed Description

### 3.18.2 Macro Definition Documentation

#### 3.18.2.1 GICR

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

General Interrupt Control Register.

Bit	7	6	5	4	3	2	1	0	
	<b>INT1</b>	<b>INT0</b>	<b>INT2</b>	–	–	–	<b>IVSEL</b>	<b>IVCE</b>	<b>GICR</b>
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 - INT1: External Interrupt Request 1 Enable
- Bit 6 - INT0: External Interrupt Request 0 Enable
- Bit 5 - INT2: External Interrupt Request 2 Enable

Definition at line 188 of file [RegisterFile.h](#).

### 3.18.2.2 GIFR

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

General Interrupt Flag Register.

Bit	7	6	5	4	3	2	1	0	
	<b>INTF1</b>	<b>INTF0</b>	<b>INTF2</b>	–	–	–	–	–	<b>GIFR</b>
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 - INTF1: External Interrupt Flag 1
- Bit 6 - INTF0: External Interrupt Flag 0
- Bit 5 - INTF2: External Interrupt Flag 2

Definition at line 199 of file [RegisterFile.h](#).

### 3.18.2.3 MCUCR

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

MCU Control Register.

Bit	7	6	5	4	3	2	1	0	
	<b>SE</b>	<b>SM2</b>	<b>SM1</b>	<b>SM0</b>	<b>ISC11</b>	<b>ISC10</b>	<b>ISC01</b>	<b>ISC00</b>	<b>MCUCR</b>
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 3, 2 - ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0.
- Interrupt 0 and interrupt 1 Sense Control.

ISCx1	ISCx0	Description
0	0	The low level of INTx generates an interrupt request.
0	1	Any logical change on INTx generates an interrupt request.
1	0	The falling edge of INTx generates an interrupt request.
1	1	The rising edge of INTx generates an interrupt request.

Note

x may be 0 or 1.

Definition at line 216 of file [RegisterFile.h](#).

## 3.18.2.4 MCUCSR

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

MCU Control and Status Register.

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0						

- Bit 6 - ISC2: Interrupt Sense Control 2

ISC2	Description
0	The falling edge on INT2 activates the interrupt request.
1	The rising edge on INT2 activates the interrupt request.

•

Definition at line 230 of file [RegisterFile.h](#).



## File Documentation

## 4.2 app.c

```
00001 /*****
00002 */
00003 */
00004 */
00005 */
```

#### 4.4 app.h

```
00001  /******
00002  /*                                     Author   :  Ehab Omara
00003  /*                                     Date      :  8/10/2022 12:03:55 PM
00004  /*                                     File name:  app.h
00005  /******
00006
00007  #ifndef APP_H_
00008  #define APP_H_
00009
00010
00011
00012
00013
00014  #endif /* APP_H_ */
```

## 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.c
```

## 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 \
00008 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h
00009
00010 ../ECUAL/Button\ driver/Button.h:
00011
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:
00017
00018 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h:
00019
00020 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h:
00021
00022 ../ECUAL/Button\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h:
```

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

### 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
00009
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:
00021
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 ../ECUAL/LED\ driver/LED.h \
00002 ../ECUAL/LED\ driver/../../Service/ATmega32Port.h \
00003 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/DIO.h \
00004 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h \
00005 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h \
00006 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h \
00007 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h \
00008 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util\delay.h
00009 \
00009 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\inttypes.h \
00010 c:\program\ files\
00010 (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\lib\gcc\avr\5.4.0\include\stdint.h \
00011 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\stdint.h \
00012 c:\program\ files\
00012 (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util\delay_basic.h \
00013 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\math.h
00014
00015 ../ECUAL/LED\ driver/LED.h:
00016
00017 ../ECUAL/LED\ driver/../../Service/ATmega32Port.h:
00018
00019 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/DIO.h:
00020
00021 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/ATmega32Port.h:
00022
00023 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/BitMath.h:
00024
00025 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h:
00026
00027 ../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/RegisterFile.h:
00028
00029 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util\delay.h:
00030
00031 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\inttypes.h:
00032
00033 c:\program\ files\
00033 (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\lib\gcc\avr\5.4.0\include\stdint.h:
00034
00035 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\stdint.h:
00036
00037 c:\program\ files\
00037 (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\util\delay_basic.h:
00038
00039 c:\program\ files\ (x86)\atmel\studio\7.0\toolchain\avr8\avr8-gnu-toolchain\avr\include\math.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.o: ../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.*

## 4.16 Button.c

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

```
00001
00002
00003 /******
00004  */
00005  */
00006  */
00007  */
00008  */
00009 #include "Button.h"
00010
00011 EN_pinErro_t buttonInit (EN_pinNum_t buttonPin)
00012 {
00013     return DIO_pinInit (buttonPin, Input);
00014 }
00015 /******
00016 EN_pinErro_t buttonRead (EN_pinNum_t buttonPin, EN_pinState_t *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.*







## 4.23 main.c File Reference

```
#include "../ECUAL/LED driver/LED.h"
#include <util/delay.h>
```

### Functions

- int [main](#) (void)

#### 4.23.1 Function Documentation

##### 4.23.1.1 main()

```
int main (
    void )
```

Definition at line 10 of file [main.c](#).

## 4.24 main.c

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

```
00001
00002  /* *****
00003  /*                                     Author   :   Ehab Omara
00004  /*                                     Date      :   8/10/2022 12:00:19 PM
00005  /*                                     File name:   main.c
00006  /* *****
00007  #include "../ECUAL/LED driver/LED.h"
00008  #include <util/delay.h>
00009
00010  int main(void)
00011  {
00012      ledInit(PA0);
00013      while (1)
00014      {
00015          ledToggle(PA0);
00016          _delay_ms(200);
00017      }
00018      return 0;
00019  }
00020
00021
```

## 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  /*                                     Date      :   8/10/2022 3:39:46 PM
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;
00015     //check if the pin is located in port A
00016     if (pinNum <= PA7)
00017     {
00018         if (pinDirection == Output)
00019         {
00020             setBit(DDRA,pinNum);
00021         }
00022         else if (pinDirection == Input)
00023         {
00024             clrBit(DDRA,pinNum);
00025         }
00026         else
00027         {
00028             error = WRONG_PIN_DIR;
00029         }
00030     }
00031     //check if the pin is located in port B
00032     else if (pinNum <= PB7)
00033     {
00034         pinNum-=PORTB_OFFSET;
00035         if (pinDirection == Output)
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     }
00048     //check if the pin is located in port C
00049     else if (pinNum <= PC7)
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 }
00065 //check if the pin is located in port D
00066 else if (pinNum <= PD7)
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
00078     {
00079         error = WRONG_PIN_DIR;
00080     }
00081 }
00082 //if the pinNum is wrong
00083 else
00084 {
00085     error = WRONG_PIN_NUM;
00086 }
00087 return error;
00088 }
00089
/*****
00090 EN_pinError_t DIO_pinWrite(EN_pinNum_t pinNum, EN_pinState_t pinState)
00091 {
00092     EN_pinError_t error = OK;
00093     //check if the pin is located in port A
00094     if (pinNum <= PA7)
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 B
00110     else if (pinNum <= PB7)
00111     {
00112         pinNum-=PORTB_OFFSET;
00113         if (pinState == High)
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
00127     else if (pinNum <= PC7)
00128     {
00129         if (pinState == High)
00130         {
00131             setBit(PORTC, pinNum);
00132         }
00133         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 D

```

```

00143     else if (pinNum <= PD7)
00144     {
00145         if (pinState == High)
00146         {
00147             setBit(PORTD,pinNum);
00148         }
00149         else if (pinState == Low)
00150         {
00151             clrBit(PORTD,pinNum);
00152         }
00153         else
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;
00169     //check if the pin is located in port A
00170     if (pinNum <= PA7)
00171     {
00172         *pinState = getBit(PINA,pinNum);
00173     }
00174     //check if the pin is located in port B
00175     else if (pinNum <= PB7)
00176     {
00177         pinNum-=PORTB_OFFSET;
00178         *pinState = getBit(PINB,pinNum);
00179     }
00180     //check if the pin is located in port C
00181     else if (pinNum <= PC7)
00182     {
00183         *pinState = getBit(PINC,pinNum);
00184     }
00185     //check if the pin is located in port D
00186     else if (pinNum <= PD7)
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)
00203     {
00204         toggleBit(PORTA,pinNum);
00205     }
00206     //check if the pin is located in port B
00207     else if (pinNum <= PB7)
00208     {
00209         pinNum-=PORTB_OFFSET;
00210         toggleBit(PORTB,pinNum);
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     }
00222     //if the pinNum is wrong
00223     else
00224     {
00225         error = WRONG_PIN_NUM;
00226     }
00227     return error;

```

```
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"
```

### 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.](#)

```
00001
```

```
/* ****
```

```
00007 #ifndef DIO_H_
00008 #define DIO_H_
00009
00010 #include "../Service/ATmega32Port.h"
00011 #include "../Service/BitMath.h"
00012 #include "../Service/dataTypes.h"
00013 #include "../Service/RegisterFile.h"
00040 EN_pinErro_t DIO_pinInit(EN_pinNum_t pinNum, EN_pinDirection_t pinDirection);
00058 EN_pinErro_t DIO_pinWrite(EN_pinNum_t pinNum, EN_pinState_t pinState);
00072 EN_pinErro_t DIO_pinToggle(EN_pinNum_t pinNum);
00086 EN_pinErro_t DIO_pinRead(EN_pinNum_t pinNum, EN_pinState_t *pinState);
00090 #endif /* DIO_H_ */
```

## 4.29 MCAL/Ext interrupt driver/Ext interrupt.c File Reference

### 4.30 Ext interrupt.c

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

```
00001  /*****
00002  /*                                     Author   :   Ehab Omara
00003  /*                                     Date      :   8/13/2022 4:40:08 AM
00004  /*                                     File name:   Ext interrupt.c
00005  /*
00006  *****/
```

## 4.31 MCAL/Ext interrupt driver/Ext interrupt.h File Reference

### Macros

- `#define` [EXT](#) INTERRUPT\_H\_

#### 4.31.1 Macro Definition Documentation

##### 4.31.1.1 EXT

```
#define EXT INTERRUPT_H_
```

Definition at line 8 of file [Ext interrupt.h](#).

### 4.32 Ext interrupt.h

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

```
00001  /*****
00002  /*                                     Author   :   Ehab Omara
00003  /*                                     Date      :   8/13/2022 4:39:49 AM
00004  /*                                     File name:   Ext interrupt.h
00005  /*
00006  *****/
00007  #ifndef EXT INTERRUPT_H_
00008  #define EXT INTERRUPT_H_
00009
00010
00011
00012
00013
00014  #endif /* EXT INTERRUPT_H_ */
```



## 4.33 MCAL/Interrupt/Interrupt.h File Reference

### Macros

- `#define sei() __asm__ __volatile__ ("sei" ::: "memory")`
- `#define cli() __asm__ __volatile__ ("cli" ::: "memory")`
- `#define EXT_INT0 __vector_1`
- `#define EXT_INT1 __vector_2`
- `#define EXT_INT2 __vector_3`
- `#define TIM2_COMP __vector_4`
- `#define TIM2_OVF __vector_5`
- `#define TIM1_CAPT __vector_6`
- `#define TIM1_COMPA __vector_7`
- `#define TIM1_COMPB __vector_8`
- `#define TIM1_OVF __vector_9`
- `#define TIM0_COMP __vector_10`
- `#define TIM0_OVF __vector_11`
- `#define SPI_STC __vector_12`
- `#define USART_RXC __vector_13`
- `#define USART_UDRE __vector_14`
- `#define USART_TXC __vector_15`
- `#define ADC __vector_16`
- `#define EE_RDY __vector_17`
- `#define ANA_COMP __vector_18`
- `#define TWI __vector_19`
- `#define SPM_RDY __vector_20`
- `#define ISR(INT_VECT)`

*interrupt service routine Macro.*

## 4.34 Interrupt.h

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

```
00001 /*****
00002  */
00003  */
00004  */
00005  */
00006 /*****
00007 #ifndef INTERRUPT_H_
00008 #define INTERRUPT_H_
00035 # define sei() __asm__ __volatile__ ("sei" ::: "memory")
00036
00046 # define cli() __asm__ __volatile__ ("cli" ::: "memory")
00047
00048 #define EXT_INT0 __vector_1
00049 #define EXT_INT1 __vector_2
00050 #define EXT_INT2 __vector_3
00051 #define TIM2_COMP __vector_4
00052 #define TIM2_OVF __vector_5
00053 #define TIM1_CAPT __vector_6
00054 #define TIM1_COMPA __vector_7
00055 #define TIM1_COMPB __vector_8
00056 #define TIM1_OVF __vector_9
00057 #define TIM0_COMP __vector_10
00058 #define TIM0_OVF __vector_11
00059 #define SPI_STC __vector_12
00060 #define USART_RXC __vector_13
00061 #define USART_UDRE __vector_14
00062 #define USART_TXC __vector_15
```

```

00063 #define ADC                __vector_16
00064 #define EE_RDY              __vector_17
00065 #define ANA_COMP            __vector_18
00066 #define TWI                  __vector_19
00067 #define SPM_RDY              __vector_20
00078 #define ISR(INT_VECT)      void INT_VECT(void) __attribute__((signal,used));\
00079 void INT_VECT(void)
00082 #endif /* INTERRUPT_H_ */

```

## 4.35 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](#) }

## 4.36 ATmega32Port.h

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

```

00001
00002 /*
00003 */
00004 /*
00005 */
00006
00007
00008
00009
00010
00022 typedef enum
00023 {
00024     /*PORTA pins*/
00025     PA0,
00026     PA1,
00027     PA2,
00028     PA3,
00029     PA4,
00030     PA5,
00031     PA6,
00032     PA7,

```

```

00033      /*PORTB pins*/
00034      PB0,
00035      PB1,
00036      PB2,
00037      PB3,
00038      PB4,
00039      PB5,
00040      PB6,
00041      PB7,
00042      /*PORTC pins*/
00043      PC0,
00044      PC1,
00045      PC2,
00046      PC3,
00047      PC4,
00048      PC5,
00049      PC6,
00050      PC7,
00051      /*PORTD pins*/
00052      PD0,
00053      PD1,
00054      PD2,
00055      PD3,
00056      PD4,
00057      PD5,
00058      PD6,
00059      PD7
00060 }EN_pinNum_t;
00061
00062 #define PORTA_OFFSET    0
00063 #define PORTB_OFFSET    8
00064 #define PORTC_OFFSET    16
00065 #define PORTD_OFFSET    24
00066 typedef enum
00067 {
00068     Low,
00069     High
00070 }EN_pinState_t;
00071 typedef enum
00072 {
00073     Input,
00074     Output
00075 }EN_pinDirection_t;
00076 typedef enum
00077 {
00078     OK,
00079     WRONG_PIN_NUM,
00080     WRONG_PIN_DIR,
00081     WRONG_PIN_STATE
00082 }EN_pinError_t;
00083 #endif /* ATMEGA32PORT_H_ */

```

## 4.37 Service/BitMath.h File Reference

### 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.*

## 4.38 BitMath.h

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

```

00001
00002 /*****
00007 #ifndef BITMATH_H_
00008 #define BITMATH_H_
00009
00026 #define setBit(reg,bitNum)  reg |= (1<<bitNum)
00037 #define clrBit(reg,bitNum)  reg &= (~(1<<bitNum))
00050 #define toggleBit(reg,bitNum)  reg ^= (1<<bitNum)
00062 #define getBit(reg,bitNum)      ((reg>>bitNum) & 0x01)
00066 #endif /* BITMATH_H_ */

```

## 4.39 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](#)

## 4.40 dataTypes.h

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

```

00001
00002 /*****
00003 */
00004 */
00005
00006 /*****
00007 #ifndef DATATYPES_H_
00008 #define DATATYPES_H_
00015 typedef unsigned char      uint8_t;
00016 typedef signed char         sint8_t;
00017 typedef unsigned short int  uint16_t;
00018 typedef signed short int    sint16_t;
00019 typedef unsigned long int   uint32_t;
00020 typedef signed long int     sint32_t;
00021 typedef float               float32_t;
00022 typedef double              float64_t;
00023 typedef long double         float128_t;
00027 #endif /* DATATYPES_H_ */

```

## 4.41 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.
- #define GICR (\*((volatile uint8\_t\*)0x5B))  
General Interrupt Control Register.
- #define GIFR (\*((volatile uint8\_t\*)0x5A))  
General Interrupt Flag Register.
- #define MCUCR (\*((volatile uint8\_t\*)0x55))  
MCU Control Register.
- #define MCUCSR (\*((volatile uint8\_t\*)0x54))  
MCU Control and Status Register.

## 4.42 RegisterFile.h

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

```

00001
00002 /*
00003 /*
00004 /*
00005 /*
00006
00007 #ifndef REGISTERFILE_H_
00008 #define REGISTERFILE_H_
00009
00010 /*
00011 * if the DDRx is set to be output and we write High to the PORTx
00012 * 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 input
00075 #define PINA  (*((volatile uint8_t*)0x39)) //this register to read a value from a pin
00079 /***** Port B registers
00093 #define PORTB (*((volatile uint8_t*)0x38))
00101 #define DDRB  (*((volatile uint8_t*)0x37))
00109 #define PINB  (*((volatile uint8_t*)0x36))

```

```
00111 /***** Port C registers
00112 *****/
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
00143 *****/
00155 #define PORTD    (*((volatile uint8_t*)0x32))
00163 #define DDRD     (*((volatile uint8_t*)0x31))
00171 #define PIND     (*((volatile uint8_t*)0x30))
00173 /***** Interrupts registers
00174 *****/
00188 #define GICR     (*((volatile uint8_t*)0x5B))
00189
00199 #define GIFR     (*((volatile uint8_t*)0x5A))
00216 #define MCUCR    (*((volatile uint8_t*)0x55))
00217
00230 #define MCUCSR   (*((volatile uint8_t*)0x54))
00235 #endif /* REGISTERFILE_H_ */
```

# Index

- ADC
  - ATMEGA32 interrupts definitions, [13](#)
- ANA\_COMP
  - ATMEGA32 interrupts definitions, [13](#)
- App/app.c, [37](#)
- App/app.h, [37](#)
- ATMEGA32 interrupts definitions, [12](#)
  - ADC, [13](#)
  - ANA\_COMP, [13](#)
  - cli, [13](#)
  - EE\_RDY, [14](#)
  - EXT\_INT0, [14](#)
  - EXT\_INT1, [14](#)
  - EXT\_INT2, [14](#)
  - ISR, [15](#)
  - sei, [15](#)
  - SPI\_STC, [15](#)
  - SPM\_RDY, [16](#)
  - TIM0\_COMP, [16](#)
  - TIM0\_OVF, [16](#)
  - TIM1\_CAPT, [16](#)
  - TIM1\_COMPA, [16](#)
  - TIM1\_COMPB, [17](#)
  - TIM1\_OVF, [17](#)
  - TIM2\_COMP, [17](#)
  - TIM2\_OVF, [17](#)
  - TWI, [17](#)
  - USART\_RXC, [18](#)
  - USART\_TXC, [18](#)
  - USART\_UDRE, [18](#)
- Bit math, [22](#)
  - clrBit, [22](#)
  - getBit, [23](#)
  - setBit, [23](#)
  - toggleBit, [24](#)
- Button driver, [5](#)
  - buttonInit, [5](#)
  - buttonRead, [6](#)
- buttonInit
  - Button driver, [5](#)
- buttonRead
  - Button driver, [6](#)
- cli
  - ATMEGA32 interrupts definitions, [13](#)
- clrBit
  - Bit math, [22](#)
- DDRA
  - Port A registers, [28](#)
- DDRB
  - Port B registers, [29](#)
- DDRC
  - Port C registers, [30](#)
- DDRD
  - Port D registers, [32](#)
- Debug/App/app.d, [38](#)
- Debug/ECUAL/Button driver/Button.d, [38](#)
- Debug/ECUAL/LED driver/LED.d, [38](#)
- Debug/main.d, [39](#)
- Debug/MCAL/Dio driver/DIO.d, [39](#)
- Definition of data types, [24](#)
  - 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](#)
- DIO driver, [9](#)
  - DIO\_pinInit, [10](#)
  - DIO\_pinRead, [10](#)
  - DIO\_pinToggle, [11](#)
  - DIO\_pinWrite, [11](#)
- DIO\_pinInit
  - DIO driver, [10](#)
- DIO\_pinRead
  - DIO driver, [10](#)
- DIO\_pinToggle
  - DIO driver, [11](#)
- DIO\_pinWrite
  - DIO driver, [11](#)
- ECUAL layer, [5](#)
  - ECUAL/Button driver/Button.c, [40](#)
  - ECUAL/Button driver/Button.h, [40](#), [41](#)
  - ECUAL/LED driver/LED.c, [41](#)
  - ECUAL/LED driver/LED.h, [42](#)
- EE\_RDY
  - ATMEGA32 interrupts definitions, [14](#)
- EN\_pinDirection\_t
  - MCU ports, [20](#)
- EN\_pinErro\_t
  - MCU ports, [20](#)
- EN\_pinNum\_t
  - MCU ports, [20](#)
- EN\_pinState\_t

- MCU ports, [21](#)
- EXT
  - Ext interrupt.h, [48](#)
- Ext interrupt.h
  - EXT, [48](#)
- EXT\_INT0
  - ATMEGA32 interrupts definitions, [14](#)
- EXT\_INT1
  - ATMEGA32 interrupts definitions, [14](#)
- EXT\_INT2
  - ATMEGA32 interrupts definitions, [14](#)
- float128\_t
  - Definition of data types, [25](#)
- float32\_t
  - Definition of data types, [25](#)
- float64\_t
  - Definition of data types, [25](#)
- getBit
  - Bit math, [23](#)
- GICR
  - Interrupt registers, [33](#)
- GIFR
  - Interrupt registers, [33](#)
- High
  - MCU ports, [21](#)
- I/O registers, [27](#)
- Input
  - MCU ports, [20](#)
- Interrupt registers, [33](#)
  - GICR, [33](#)
  - GIFR, [33](#)
  - MCUCR, [34](#)
  - MCUCSR, [34](#)
- Interrupts driver, [12](#)
- ISR
  - ATMEGA32 interrupts definitions, [15](#)
- LED driver, [6](#)
  - ledInit, [7](#)
  - ledOff, [7](#)
  - ledOn, [8](#)
  - ledToggle, [8](#)
- ledInit
  - LED driver, [7](#)
- ledOff
  - LED driver, [7](#)
- ledOn
  - LED driver, [8](#)
- ledToggle
  - LED driver, [8](#)
- Low
  - MCU ports, [21](#)
- main
  - main.c, [43](#)
- main.c, [43](#)
- main, [43](#)
- MCAL layer, [9](#)
- MCAL/Dio driver/DIO.c, [43](#), [44](#)
- MCAL/Dio driver/DIO.h, [47](#)
- MCAL/Ext interrupt driver/Ext interrupt.c, [48](#)
- MCAL/Ext interrupt driver/Ext interrupt.h, [48](#)
- MCAL/Interrupt/Interrupt.h, [49](#)
- MCU ports, [18](#)
  - EN\_pinDirection\_t, [20](#)
  - EN\_pinErro\_t, [20](#)
  - EN\_pinNum\_t, [20](#)
  - EN\_pinState\_t, [21](#)
  - High, [21](#)
  - Input, [20](#)
  - Low, [21](#)
  - OK, [20](#)
  - Output, [20](#)
  - 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, [21](#)
  - PC4, [21](#)
  - PC5, [21](#)
  - PC6, [21](#)
  - PC7, [21](#)
  - PD0, [21](#)
  - PD1, [21](#)
  - PD2, [21](#)
  - PD3, [21](#)
  - PD4, [21](#)
  - PD5, [21](#)
  - PD6, [21](#)
  - PD7, [21](#)
  - PORTA\_OFFSET, [19](#)
  - PORTB\_OFFSET, [19](#)
  - PORTC\_OFFSET, [19](#)
  - PORTD\_OFFSET, [19](#)
  - WRONG\_PIN\_DIR, [20](#)
  - WRONG\_PIN\_NUM, [20](#)
  - WRONG\_PIN\_STATE, [20](#)
- MCU Registers, [27](#)
- MCUCR



- Interrupt registers, [34](#)
- MCUCSR
  - Interrupt registers, [34](#)
- OK
  - MCU ports, [20](#)
- Output
  - MCU ports, [20](#)
- PA0
  - MCU ports, [21](#)
- PA1
  - MCU ports, [21](#)
- PA2
  - MCU ports, [21](#)
- PA3
  - MCU ports, [21](#)
- PA4
  - MCU ports, [21](#)
- PA5
  - MCU ports, [21](#)
- PA6
  - MCU ports, [21](#)
- PA7
  - MCU ports, [21](#)
- PB0
  - MCU ports, [21](#)
- PB1
  - MCU ports, [21](#)
- PB2
  - MCU ports, [21](#)
- PB3
  - MCU ports, [21](#)
- PB4
  - MCU ports, [21](#)
- PB5
  - MCU ports, [21](#)
- PB6
  - MCU ports, [21](#)
- PB7
  - MCU ports, [21](#)
- PC0
  - MCU ports, [21](#)
- PC1
  - MCU ports, [21](#)
- PC2
  - MCU ports, [21](#)
- PC3
  - MCU ports, [21](#)
- PC4
  - MCU ports, [21](#)
- PC5
  - MCU ports, [21](#)
- PC6
  - MCU ports, [21](#)
- PC7
  - MCU ports, [21](#)
- PD0
  - MCU ports, [21](#)
- PD1
  - MCU ports, [21](#)
- PD2
  - MCU ports, [21](#)
- PD3
  - MCU ports, [21](#)
- PD4
  - MCU ports, [21](#)
- PD5
  - MCU ports, [21](#)
- PD6
  - MCU ports, [21](#)
- PD7
  - MCU ports, [21](#)
- PINA
  - Port A registers, [28](#)
- PINB
  - Port B registers, [29](#)
- PINC
  - Port C registers, [31](#)
- PIND
  - Port D registers, [32](#)
- Port A registers, [28](#)
  - DDRA, [28](#)
  - PINA, [28](#)
  - PORTA, [28](#)
- Port B registers, [29](#)
  - DDRB, [29](#)
  - PINB, [29](#)
  - PORTB, [30](#)
- Port C registers, [30](#)
  - DDRC, [30](#)
  - PINC, [31](#)
  - PORTC, [31](#)
- Port D registers, [32](#)
  - DDRD, [32](#)
  - PIND, [32](#)
  - PORTD, [32](#)
- PORTA
  - Port A registers, [28](#)
- PORTA\_OFFSET
  - MCU ports, [19](#)
- PORTB
  - Port B registers, [30](#)
- PORTB\_OFFSET
  - MCU ports, [19](#)
- PORTC
  - Port C registers, [31](#)
- PORTC\_OFFSET
  - MCU ports, [19](#)
- PORTD
  - Port D registers, [32](#)
- PORTD\_OFFSET
  - MCU ports, [19](#)
- sei
  - ATMEGA32 interrupts definitions, [15](#)
- Service layer, [26](#)
- Service/ATmega32Port.h, [50](#)

Service/BitMath.h, [51](#)  
Service/dataTypes.h, [52](#)  
Service/RegisterFile.h, [52](#), [53](#)  
setBit  
    Bit math, [23](#)  
sint16\_t  
    Definition of data types, [25](#)  
sint32\_t  
    Definition of data types, [25](#)  
sint8\_t  
    Definition of data types, [25](#)  
SPI\_STC  
    ATMEGA32 interrupts definitions, [15](#)  
SPM\_RDY  
    ATMEGA32 interrupts definitions, [16](#)  
  
TIM0\_COMP  
    ATMEGA32 interrupts definitions, [16](#)  
TIM0\_OVF  
    ATMEGA32 interrupts definitions, [16](#)  
TIM1\_CAPT  
    ATMEGA32 interrupts definitions, [16](#)  
TIM1\_COMPA  
    ATMEGA32 interrupts definitions, [16](#)  
TIM1\_COMPB  
    ATMEGA32 interrupts definitions, [17](#)  
TIM1\_OVF  
    ATMEGA32 interrupts definitions, [17](#)  
TIM2\_COMP  
    ATMEGA32 interrupts definitions, [17](#)  
TIM2\_OVF  
    ATMEGA32 interrupts definitions, [17](#)  
toggleBit  
    Bit math, [24](#)  
TWI  
    ATMEGA32 interrupts definitions, [17](#)  
  
uint16\_t  
    Definition of data types, [26](#)  
uint32\_t  
    Definition of data types, [26](#)  
uint8\_t  
    Definition of data types, [26](#)  
USART\_RXC  
    ATMEGA32 interrupts definitions, [18](#)  
USART\_TXC  
    ATMEGA32 interrupts definitions, [18](#)  
USART\_UDRE  
    ATMEGA32 interrupts definitions, [18](#)  
  
WRONG\_PIN\_DIR  
    MCU ports, [20](#)  
WRONG\_PIN\_NUM  
    MCU ports, [20](#)  
WRONG\_PIN\_STATE  
    MCU ports, [20](#)