On-Demand Traffic Control

Generated by Doxygen 1.9.4

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Module Documentation

3.1 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	*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()

```
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	*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)

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	*ledPin*	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()

```
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	*ledPin*	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 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	*ledPin*	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 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	*ledPin*	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 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	*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()

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

This function reads the state of the pin.

DIO_pinRead

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

Parameters

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

Return values

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

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	*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()

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

This function writes High or Low on the pin.

DIO_pinWrite

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

Parameters

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

Return values

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

Definition at line 90 of file DIO.c.

3.6 ATMEGA32 external interrupts driver

External interrupts driver.

Enumerations

- enum EN_interruptNum_t { INT2 = 5 , INT0 , INT1 }

  External interrupt number.
- enum EN_interruptSenseControl_t { LOW_LEVEL , ANY_LOGICAL_CHANGE , FALLING_EDGE , RISING_EDGE }

  External interrupt sense control.
- enum EN_interruptError_t { INT_OK , WRONG_INT_NUM , WRONG_SENSE_CONTROL }

  External interrupt errors.

Functions

- EN_interruptError_t Ext_interruptInit (EN_interruptNum_t interruptNum, EN_interruptSenseControl_t interruptSenseControl)

  External interrupt init.

External interrupts pins

- These are the pins which connected to each interrupt.
- It should be configured as Input.

- #define INT0_PIN (PD2 - PORTD_OFFSET)
- #define INT1_PIN (PD3 - PORTD_OFFSET)
- #define INT2_PIN (PB2 - PORTB_OFFSET)

INT0 sense control

- These two bits ISC00 and ISC01 which located in MCUCR register control the INT0 sense control.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

-

- #define ISC00 0
- #define ISC01 1

INT1 sense control

- These two bits ISC10 and ISC11 which located in MCUCR register control the INT1 sense control.

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

-

- #define ISC10 2
- #define ISC11 3

INT2 sense control

- This bit ISC2 which located in MCUCSR register control the INT2 sense control.

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

-

- #define ISC2 6

3.6.1 Detailed Description

External interrupts driver.

3.6.2 Macro Definition Documentation

3.6.2.1 INT0_PIN

```
#define INT0_PIN (PD2 - PORTD_OFFSET)
```

This Pin connected to INT0 interrupt

Definition at line 29 of file Ext interrupt.h.

3.6.2.2 INT1_PIN

```
#define INT1_PIN (PD3 - PORTD_OFFSET)
```

This Pin connected to INT1 interrupt

Definition at line 30 of file Ext interrupt.h.

3.6.2.3 INT2_PIN

```
#define INT2_PIN (PB2 - PORTB_OFFSET)
```

This Pin connected to INT2 interrupt

Definition at line 31 of file Ext interrupt.h.

3.6.2.4 ISC00

```
#define ISC00 0
```

Interrupt Sense Control 0 Bit 0

Definition at line 47 of file Ext interrupt.h.

3.6.2.5 ISC01

```
#define ISC01 1
```

Interrupt Sense Control 0 Bit 1

Definition at line 48 of file Ext interrupt.h.

3.6.2.6 ISC10

```
#define ISC10 2
```

Interrupt Sense Control 1 Bit 0

Definition at line 65 of file Ext interrupt.h.

3.6.2.7 ISC11

```
#define ISC11 3
```

Interrupt Sense Control 1 Bit 1

Definition at line 66 of file Ext interrupt.h.

3.6.2.8 ISC2

```
#define ISC2 6
```

Interrupt Sense Control 2 Bit 6

Definition at line 81 of file Ext interrupt.h.

3.6.3 Enumeration Type Documentation

3.6.3.1 EN_interruptError_t

```
enum EN_interruptError_t
```

External interrupt errors.

- This enum contains the values for interrupt errors.

Enumerator

INT_OK	enum value shows that INTx parameters is right.
WRONG_INT_NUM	enum value shows that INTx number is wrong.
WRONG_SENSE_CONTROL	enum value shows that INTx sense control is wrong.

Definition at line 120 of file Ext interrupt.h.

3.6.3.2 EN_interruptNum_t

```
enum EN_interruptNum_t
```

External interrupt number.

- This enum contains the bit number for each interrupt in GICR register.
- Setting these bits will enables the interrupts.
- Clearing these bits will disables the interrupts.

Enumerator

INT2	enum value for external interrupt 2
INT0	enum value for external interrupt 0
INT1	enum value for external interrupt 1

Definition at line 92 of file Ext interrupt.h.

3.6.3.3 EN_interruptSenseControl_t

```
enum EN_interruptSenseControl_t
```

External interrupt sense control.

- This enum contains the values for interrupt sense control.
- each value represent the exact value that should be written in the MCUCR register this for INT0 and INT1 and MCUCSR register for INT2.

  Note

- INT2 has just rising and falling edge sense control.

Enumerator

LOW_LEVEL	The low level generates an interrupt request.
ANY_LOGICAL_CHANGE	Any logical change generates an interrupt request.
FALLING_EDGE	The falling edge generates an interrupt request
RISING_EDGE	The rising edge generates an interrupt request

Definition at line 107 of file Ext interrupt.h.

3.6.4 Function Documentation

3.6.4.1 Ext_interruptInit()

```
EN_interruptError_t Ext_interruptInit (
            EN_interruptNum_t interruptNum,
            EN_interruptSenseControl_t interruptSenseControl )
```

External interrupt init.

- This function configures INTx sense control.
- This function enables INTx.

Parameters

in	*interruptNum*	This is the interrupt number that needed to be enabled.
in	*interruptSenseControl*	This is the value of the interrupt sense control which the interrupt will activated at it.

Return values

INT_OK	If interruptNum and interruptSenseControl are corrects.
WRONG_INT_NUM	If interruptNum is wrong.
WRONG_SENSE_CONTROL	If interruptSenseControl is wrong.

Definition at line 9 of file Ext interrupt.c.

3.7 Interrupts driver

Modules

- ATMEGA32 external interrupts driver

  External interrupts driver.
- ATMEGA32 interrupts definitions

  Interrupts request handlers.

3.7.1 Detailed Description

3.8 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.

3.8.1 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.

3.8.2 Macro Definition Documentation

3.8.2.1 ADC

```
#define ADC __vector_16
```

This Macro defines ADC Conversion Complete Handler

Definition at line 63 of file Interrupt.h.

3.8.2.2 ANA_COMP

```
#define ANA_COMP __vector_18
```

This Macro defines Analog Comparator Handler

Definition at line 65 of file Interrupt.h.

3.8.2.3 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.

3.8.2.4 EE_RDY

```
#define EE_RDY __vector_17
```

This Macro defines EEPROM Ready Handler

Definition at line 64 of file Interrupt.h.

3.8.2.5 EXT_INT0

```
#define EXT_INT0 __vector_1
```

This Macro defines IRQ0 Handler

Definition at line 48 of file Interrupt.h.

3.8.2.6 EXT_INT1

```
#define EXT_INT1 __vector_2
```

This Macro defines IRQ1 Handler

Definition at line 49 of file Interrupt.h.

3.8.2.7 EXT_INT2

```
#define EXT_INT2 __vector_3
```

This Macro defines IRQ2 Handler

Definition at line 50 of file Interrupt.h.

3.8.2.8 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.

3.8.2.9 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.

3.8.2.10 SPI_STC

```
#define SPI_STC __vector_12
```

This Macro defines SPI Transfer Complete Handler

Definition at line 59 of file Interrupt.h.

3.8.2.11 SPM_RDY

```
#define SPM_RDY __vector_20
```

This Macro defines Store Program Memory Ready Handler

Definition at line 67 of file Interrupt.h.

3.8.2.12 TIM0_COMP

```
#define TIM0_COMP __vector_10
```

This Macro defines Timer0 Compare Handler

Definition at line 57 of file Interrupt.h.

3.8.2.13 TIM0_OVF

```
#define TIM0_OVF __vector_11
```

This Macro defines Timer0 Overflow Handler

Definition at line 58 of file Interrupt.h.

3.8.2.14 TIM1_CAPT

```
#define TIM1_CAPT __vector_6
```

This Macro defines Timer1 Capture Handler

Definition at line 53 of file Interrupt.h.

3.8.2.15 TIM1_COMPA

```
#define TIM1_COMPA __vector_7
```

This Macro defines Timer1 CompareA Handler

Definition at line 54 of file Interrupt.h.

3.8.2.16 TIM1_COMPB

```
#define TIM1_COMPB __vector_8
```

This Macro defines Timer1 CompareB Handler

Definition at line 55 of file Interrupt.h.

3.8.2.17 TIM1_OVF

```
#define TIM1_OVF __vector_9
```

This Macro defines Timer1 Overflow Handler

Definition at line 56 of file Interrupt.h.

3.8.2.18 TIM2_COMP

```
#define TIM2_COMP __vector_4
```

This Macro defines Timer2 Compare Handler

Definition at line 51 of file Interrupt.h.

3.8.2.19 TIM2_OVF

```
#define TIM2_OVF __vector_5
```

This Macro defines Timer2 Overflow Handler

Definition at line 52 of file Interrupt.h.

3.8.2.20 TWI

```
#define TWI __vector_19
```

This Macro defines Two-wire Serial Interface Handler

Definition at line 66 of file Interrupt.h.

3.8.2.21 USART_RXC

```
#define USART_RXC __vector_13
```

This Macro defines USART RX Complete Handler

Definition at line 60 of file Interrupt.h.

3.8.2.22 USART_TXC

```
#define USART_TXC __vector_15
```

This Macro defines USART TX Complete Handler

Definition at line 62 of file Interrupt.h.

3.8.2.23 USART_UDRE

```
#define USART_UDRE __vector_14
```

This Macro defines UDR Empty Handler

Definition at line 61 of file Interrupt.h.

3.9 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 }

3.9.1 Detailed Description

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

3.9.2 Macro Definition Documentation

3.9.2.1 PORTA_OFFSET

```
#define PORTA_OFFSET 0
```

This macro defines the start of the PORTA pins

Definition at line 62 of file ATmega32Port.h.

3.9.2.2 PORTB_OFFSET

```
#define PORTB_OFFSET 8
```

This macro defines the start of the PORTB pins

Definition at line 63 of file ATmega32Port.h.

3.9.2.3 PORTC_OFFSET

```
#define PORTC_OFFSET 16
```

This macro defines the start of the PORTC pins

Definition at line 64 of file ATmega32Port.h.

3.9.2.4 PORTD_OFFSET

```
#define PORTD_OFFSET 24
```

This macro defines the start of the PORTD pins

Definition at line 65 of file ATmega32Port.h.

3.9.3 Enumeration Type Documentation

3.9.3.1 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.

3.9.3.2 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.

3.9.3.3 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.

3.9.3.4 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.

3.10 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.

3.10.1 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.

3.10.2 Macro Definition Documentation

3.10.2.1 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	*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.

3.10.2.2 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	*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.

3.10.2.3 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	*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.

3.10.2.4 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.

3.11 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

3.11.1 Detailed Description

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

3.11.2 Typedef Documentation

3.11.2.1 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.

3.11.2.2 float32_t

```
typedef float float32_t
```

This is define a memory size of 4 byte float

Definition at line 21 of file dataTypes.h.

3.11.2.3 float64_t

```
typedef double float64_t
```

This is define a memory size of 8 byte float

Definition at line 22 of file dataTypes.h.

3.11.2.4 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.

3.11.2.5 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.

3.11.2.6 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.

3.11.2.7 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.

3.11.2.8 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.

3.11.2.9 uint8_t

```
typedef unsigned char uint8_t
```

This is define a memory size of 1 byte

Definition at line 15 of file dataTypes.h.

3.12 Service layer

Modules

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

3.12.1 Detailed Description

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

3.13 MCU Registers

Modules

- I/O registers
- Interrupt registers

3.13.1 Detailed Description

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

3.14 I/O registers

Modules

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

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

3.15 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.

3.15.1 Detailed Description

3.15.2 Macro Definition Documentation

3.15.2.1 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 68 of file RegisterFile.h.

3.15.2.2 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 76 of file RegisterFile.h.

3.15.2.3 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 60 of file RegisterFile.h.

3.16 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.

3.16.1 Detailed Description

3.16.2 Macro Definition Documentation

3.16.2.1 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 102 of file RegisterFile.h.

3.16.2.2 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 110 of file RegisterFile.h.

3.16.2.3 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 94 of file RegisterFile.h.

3.17 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.

3.17.1 Detailed Description

3.17.2 Macro Definition Documentation

3.17.2.1 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 133 of file RegisterFile.h.

3.17.2.2 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 141 of file RegisterFile.h.

3.17.2.3 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 125 of file RegisterFile.h.

3.18 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.

3.18.1 Detailed Description

3.18.2 Macro Definition Documentation

3.18.2.1 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 164 of file RegisterFile.h.

3.18.2.2 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 172 of file RegisterFile.h.

3.18.2.3 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 156 of file RegisterFile.h.

3.19 Interrupt registers

Macros

- #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.

3.19.1 Detailed Description

3.19.2 Macro Definition Documentation

3.19.2.1 GICR

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

General Interrupt Control Register.

Bit	7	6	5	4	3	2	1	0	
	INT1	**INT0**	**INT2**	–	–	–	**IVSEL**	**IVCE**	**GICR**
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 189 of file RegisterFile.h.

3.19.2.2 GIFR

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

General Interrupt Flag Register.

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
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 200 of file RegisterFile.h.

3.19.2.3 MCUCR

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

MCU Control Register.

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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 217 of file RegisterFile.h.

3.19.2.4 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 231 of file RegisterFile.h.

Chapter 4

File Documentation

4.1 App/app.c File Reference

4.2 app.c

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                    Author   :  Ehab Omara
      */
00003 /*                                                    Date     :  8/10/2022 12:03:35 PM
      */
00004 /*                                                    File name:  app.c
      */
00005
      /*****************************************************************************************************************
```

4.3 App/app.h File Reference

4.4 app.h

Go to the documentation of this file.

```
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  .././../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h \
00009  .././MCAL/Ext\ interrupt\ driver/Ext\ interrupt.h \
00010  .././MCAL/Ext\ interrupt\ driver/../../Service/ATmega32Port.h \
00011  .././MCAL/Ext\ interrupt\ driver/../../Service/RegisterFile.h \
00012  .././MCAL/Ext\ interrupt\ driver/../Interrupt/Interrupt.h \
00013  .././MCAL/Ext\ interrupt\ driver/../../Service/BitMath.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 .././../ECUAL/LED\ driver/../../MCAL/Dio\ driver/../../Service/dataTypes.h:
00030
00031 .././MCAL/Ext\ interrupt\ driver/Ext\ interrupt.h:
00032
00033 .././MCAL/Ext\ interrupt\ driver/../../Service/ATmega32Port.h:
00034
00035 .././MCAL/Ext\ interrupt\ driver/../../Service/RegisterFile.h:
00036
00037 .././MCAL/Ext\ interrupt\ driver/../Interrupt/Interrupt.h:
00038
00039 .././MCAL/Ext\ interrupt\ driver/../../Service/BitMath.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  ../MCAL/Dio\ driver/../../Service/dataTypes.h \
00008  ../MCAL/Dio\ driver/../Interrupt/Interrupt.h
00009
00010 ../MCAL/Dio\ driver/DIO.h:
00011
00012 ../MCAL/Dio\ driver/../../Service/ATmega32Port.h:
00013
00014 ../MCAL/Dio\ driver/../../Service/BitMath.h:
00015
00016 ../MCAL/Dio\ driver/../../Service/dataTypes.h:
00017
00018 ../MCAL/Dio\ driver/../../Service/RegisterFile.h:
00019
00020 ../MCAL/Dio\ driver/../../Service/dataTypes.h:
00021
00022 ../MCAL/Dio\ driver/../Interrupt/Interrupt.h:
```

4.15 Debug/MCAL/Ext interrupt driver/Ext interrupt.d File Reference

4.16 Ext interrupt.d

Go to the documentation of this file.

```
00001 MCAL/Ext interrupt driver/Ext interrupt.d \
00002  MCAL/Ext interrupt driver/Ext interrupt.o:  \
00003  ../MCAL/Ext\ interrupt\ driver/Ext\ interrupt.c \
00004  ../MCAL/Ext\ interrupt\ driver/Ext\ interrupt.h \
00005  ../MCAL/Ext\ interrupt\ driver/../../Service/ATmega32Port.h \
00006  ../MCAL/Ext\ interrupt\ driver/../../Service/RegisterFile.h \
00007  ../MCAL/Ext\ interrupt\ driver/../../Service/dataTypes.h \
00008  ../MCAL/Ext\ interrupt\ driver/../Interrupt/Interrupt.h \
00009  ../MCAL/Ext\ interrupt\ driver/../../Service/BitMath.h
00010
00011 ../MCAL/Ext\ interrupt\ driver/Ext\ interrupt.h:
00012
00013 ../MCAL/Ext\ interrupt\ driver/../../Service/ATmega32Port.h:
00014
00015 ../MCAL/Ext\ interrupt\ driver/../../Service/RegisterFile.h:
00016
00017 ../MCAL/Ext\ interrupt\ driver/../../Service/dataTypes.h:
00018
00019 ../MCAL/Ext\ interrupt\ driver/../Interrupt/Interrupt.h:
00020
00021 ../MCAL/Ext\ interrupt\ driver/../../Service/BitMath.h:
```

4.17 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.

4.18 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)
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 }
```

4.19 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.

4.20 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_ */
```

4.21 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)

  turn the led off.
- EN_pinNum_t ledToggle (EN_pinNum_t ledPin)

  toggle the led state.

4.22 LED.c

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00002 /*                                                      Author    :  Ehab Omara
      */
00003 /*                                                      Date      :  8/12/2022 9:42:19 PM
      */
00004 /*                                                      File name:  LED.c
      */
00005
      /*****************************************************************************************************************************
00006 #include "LED.h"
00007
00008
00009
00010 EN_pinErro_t ledInit(EN_pinNum_t ledPin)
00011 {
00012     return DIO_pinInit(ledPin,Output);
00013 }
00014 /*****************************************************************************/
00015 EN_pinErro_t ledOn(EN_pinNum_t ledPin)
00016 {
00017     return DIO_pinWrite(ledPin,High);
00018 }
00019 /*****************************************************************************/
00020 EN_pinErro_t ledOff(EN_pinNum_t ledPin)
00021 {
00022     return DIO_pinWrite(ledPin,Low);
00023 }
00024 /*****************************************************************************/
00025 EN_pinNum_t ledToggle(EN_pinNum_t ledPin)
00026 {
00027     return DIO_pinToggle(ledPin);
00028 }
```

4.23 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.

4.24 LED.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00002 /*                                                      Author    :  Ehab Omara
      */
00003 /*                                                      Date      :  8/12/2022 9:42:50 PM
      */
```

```
00004 /*                                                      File name:  LED.h
      */
00005
      /*****************************************************************************************************************
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);
00047 /*****************************************************************************/
00060 EN_pinErro_t ledOff(EN_pinNum_t ledPin);
00061 /*****************************************************************************/
00076 EN_pinNum_t ledToggle(EN_pinNum_t ledPin);
00081 #endif /* LED_H_ */
```

4.25 main.c File Reference

```
#include "./ECUAL/LED driver/LED.h"
#include "MCAL/Ext interrupt driver/Ext interrupt.h"
```

Functions

- int main (void)

4.25.1 Function Documentation

4.25.1.1 main()

```
int main (
            void  )
```

Definition at line 10 of file main.c.

4.26 main.c

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                      Author   :  Ehab Omara
      */
00003 /*                                                      Date     :  8/10/2022 12:00:19 PM
      */
00004 /*                                                      File name:  main.c
      */
00005
      /*****************************************************************************************************************
00006
00007 #include "./ECUAL/LED driver/LED.h"
00008 #include "MCAL/Ext interrupt driver/Ext interrupt.h"
00009
00010 int main(void)
```

```
00011 {
00012     ledInit(PA0);
00013     ledInit(PB0);
00014     Ext_interruptInit(INT0,ANY_LOGICAL_CHANGE);
00015     while (1)
00016     {
00017         ledOn(PB0);
00018 
00019     }
00020     return 0;
00021 }
00022 
00023 
```

4.27 MCAL/Dio driver/DIO.c File Reference

```
#include "DIO.h"
#include "../Interrupt/Interrupt.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.
- ISR (EXT_INT0)

4.27.1 Function Documentation

4.27.1.1 ISR()

```
ISR (
              EXT_INT0  )
```

Definition at line 230 of file DIO.c.

4.28 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 #include "../Interrupt/Interrupt.h"
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_pinErro_t DIO_pinWrite(EN_pinNum_t pinNum,EN_pinState_t pinState)
00091 {
00092     EN_pinErro_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
      /*************************************************************************************************************************************
00230 ISR(EXT_INT0)
00231 {
00232     DIO_pinToggle(PA0);
00233 }
```

4.29 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.

4.29.1 Detailed Description

Author

: Ehab Omara

Date

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

Definition in file DIO.h.

4.30 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_ */
```

4.31 MCAL/Ext interrupt driver/Ext interrupt.c File Reference

```
#include "Ext interrupt.h"
```

Functions

- EN_interruptError_t Ext_interruptInit (EN_interruptNum_t interruptNum, EN_interruptSenseControl_t interruptSenseControl)

  External interrupt init.

4.32 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
00007
00008 #include "Ext interrupt.h"
00009 EN_interruptError_t Ext_interruptInit(EN_interruptNum_t interruptNum,EN_interruptSenseControl_t
      interruptSenseControl)
00010 {
00011     EN_interruptError_t interruptError = INT_OK;
00012     if (interruptNum == INT0)
00013     {
00014         //check if the value of the interruptSenseControl is correct
00015         if (interruptSenseControl >=  LOW_LEVEL &&interruptSenseControl <= RISING_EDGE)
00016         {
00017             //enable INT0
00018             setBit(GICR,INT0);
00019             //clearing interruptSenseControl old value
00020             MCUCR&=(~(ISC00«0x03));
00021             //setting interruptSenseControl new value
00022             MCUCR|=interruptSenseControl«ISC00;
00023             //set INT0 pin as input
00024             clrBit(DDRD,INT0_PIN);
00025         }
00026         else
00027         {
00028             interruptError = WRONG_SENSE_CONTROL;
00029         }
00030     }
00031     else if (interruptNum == INT1)
00032     {
00033
00034         //check if the value of the interruptSenseControl is correct
00035         if (interruptSenseControl >=  LOW_LEVEL &&interruptSenseControl <= RISING_EDGE)
00036         {
00037             //enable INT1
00038             setBit(GICR,INT1);
00039             //clearing interruptSenseControl old value
00040             MCUCR&=(~(0x03«ISC10));
00041             //setting interruptSenseControl new value
00042             MCUCR|=interruptSenseControl«ISC10;
00043             //set INT1 pin as input
00044             clrBit(DDRD,INT1_PIN);
00045         }
00046         else
00047         {
00048             interruptError = WRONG_SENSE_CONTROL;
00049         }
00050     }
00051     else if (interruptNum == INT2)
00052     {
00053
00054         //check if the value of the interruptSenseControl is correct
00055         if (interruptSenseControl ==  FALLING_EDGE )
00056         {
00057             //enable INT1
00058             setBit(GICR,INT2);
00059             clrBit(MCUCSR,ISC2);
00060             //set INT2 pin as input
00061             clrBit(DDRB,INT2_PIN);
00062         }
00063         else if(interruptSenseControl == RISING_EDGE)
00064         {
00065             //enable INT1
00066             setBit(GICR,INT2);
00067             setBit(MCUCSR,ISC2);
00068             //set INT2 pin as input
00069             clrBit(DDRB,INT2_PIN);
00070         }
00071         else
00072         {
00073             interruptError = WRONG_SENSE_CONTROL;
00074         }
00075     }
00076     else
```

```
00077     {
00078         interruptError = WRONG_INT_NUM;
00079     }
00080     if (interruptError == INT_OK)
00081     {
00082         //enable global interrupt
00083         sei();
00084     }
00085     return interruptError;
00086 }
00087
00088
```

4.33 MCAL/Ext interrupt driver/Ext interrupt.h File Reference

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

Macros

External interrupts pins

- *These are the pins which connected to each interrupt.*
- *It should be configured as Input.*

- #define INT0_PIN (PD2 - PORTD_OFFSET)
- #define INT1_PIN (PD3 - PORTD_OFFSET)
- #define INT2_PIN (PB2 - PORTB_OFFSET)

INT0 sense control

- *These two bits ISC00 and ISC01 which located in MCUCR register control the INT0 sense control.*

ISC01	*ISC00*	*Description*
0	*0*	*The low level of INT0 generates an interrupt request.*
0	*1*	*Any logical change on INT0 generates an interrupt request.*
1	*0*	*The falling edge of INT0 generates an interrupt request.*
1	*1*	*The rising edge of INT0 generates an interrupt request.*

-

- #define ISC00 0
- #define ISC01 1

INT1 sense control

- *These two bits ISC10 and ISC11 which located in MCUCR register control the INT1 sense control.*

ISC11	*ISC10*	*Description*
0	*0*	*The low level of INT1 generates an interrupt request.*
0	*1*	*Any logical change on INT1 generates an interrupt request.*
1	*0*	*The falling edge of INT1 generates an interrupt request.*
1	*1*	*The rising edge of INT1 generates an interrupt request.*

-

- #define ISC10 2
- #define ISC11 3

INT2 sense control

- *This bit ISC2 which located in MCUCSR register control the INT2 sense control.*

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

-

- #define ISC2 6

Enumerations

- enum EN_interruptNum_t { INT2 = 5 , INT0 , INT1 }

  External interrupt number.
- enum EN_interruptSenseControl_t { LOW_LEVEL , ANY_LOGICAL_CHANGE , FALLING_EDGE , RISING_EDGE }

  External interrupt sense control.
- enum EN_interruptError_t { INT_OK , WRONG_INT_NUM , WRONG_SENSE_CONTROL }

  External interrupt errors.

Functions

- EN_interruptError_t Ext_interruptInit (EN_interruptNum_t interruptNum, EN_interruptSenseControl_t interruptSenseControl)

  External interrupt init.

4.34 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 #include "../../Service/ATmega32Port.h"
00011 #include "../../Service/RegisterFile.h"
00012 #include "../Interrupt/Interrupt.h"
00013 #include "../../Service/BitMath.h"
00014
00029 #define INT0_PIN (PD2 - PORTD_OFFSET)
00030 #define INT1_PIN (PD3 - PORTD_OFFSET)
00031 #define INT2_PIN (PB2 - PORTB_OFFSET)
```

```
00033
00047 #define ISC00 0
00048 #define ISC01 1
00050
      /*************************************************************************************************************************
00051
00065 #define ISC10 2
00066 #define ISC11 3
00068
00069
      /*************************************************************************************************************************
00070
00081 #define ISC2 6
00083
      /*************************************************************************************************************************
00084
00092 typedef enum
00093 {
00094     INT2 = 5,
00095     INT0,
00096     INT1
00097 }EN_interruptNum_t;
00098
      /*************************************************************************************************************************
00107 typedef enum
00108 {
00109     LOW_LEVEL,
00110     ANY_LOGICAL_CHANGE,
00111     FALLING_EDGE,
00112     RISING_EDGE
00113 }EN_interruptSenseControl_t;
00114
      /*************************************************************************************************************************
00120 typedef enum
00121 {
00122     INT_OK,
00123     WRONG_INT_NUM,
00124     WRONG_SENSE_CONTROL
00125 }EN_interruptError_t;
00126
      /*************************************************************************************************************************
00139 EN_interruptError_t Ext_interruptInit(EN_interruptNum_t interruptNum,EN_interruptSenseControl_t
      interruptSenseControl);
00141 #endif /* EXT_INTERRUPT_H_ */
```

4.35 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.

4.36 Interrupt.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                          Author   :  Ehab Omara
      */
00003 /*                                                          Date     :  8/13/2022 1:08:16 AM
      */
00004 /*                                                          File name:  Interrupt.h
      */
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_ */
```

4.37 MCAL/Timer driver/Timer_0.c File Reference

4.38 Timer_0.c

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                          Author   :  Ehab Omara
      */
00003 /*                                                          Date     :  8/14/2022 12:55:03 PM
      */
00004 /*                                                          File name:  Timer_0.c
      */
00005
      /*****************************************************************************************************************
```

4.39 MCAL/Timer driver/Timer_0.h File Reference

4.40 Timer_0.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                    Author   :  Ehab Omara
      */
00003 /*                                                    Date     :  8/14/2022 12:55:53 PM
      */
00004 /*                                                    File name:  Timer_0.h
      */
00005
      /*****************************************************************************************************************
00006
00007 #ifndef TIMER_0_H_
00008 #define TIMER_0_H_
00009
00010
00011
00012
00013
00014 #endif /* TIMER_0_H_ */
```

4.41 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 }

4.42 ATmega32Port.h

Go to the documentation of this file.

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

4.43 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.

4.44 BitMath.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************************
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_ */
```

4.45 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

4.46 dataTypes.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************************
00002 /*                                                      Author   :  Ehab Omara
      */
00003 /*                                                      Date     :  8/10/2022 12:06:28 PM
      */
00004 /*                                                      File name:  dataTypes.h
      */
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_ */
```

4.47 Service/RegisterFile.h File Reference

```
#include "dataTypes.h"
```

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.

4.48 RegisterFile.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00002 /*                                                    Author   :  Ehab Omara
      */
00003 /*                                                    Date     :  8/10/2022 12:06:56 PM
      */
00004 /*                                                    File name:  RegisterFile.h
      */
00005
      /*****************************************************************************************************************
00006
00007 #ifndef REGISTERFILE_H_
00008 #define REGISTERFILE_H_
00009
00010 #include "dataTypes.h"
00011 /*
00012 * if the DDRx is set to be output and we write High to the PORTx
00013 * this will activate the internal Pull up resistor.
00014 */
00015
00046 /************************************************************ Port A registers
      ************************************************************/
00060 #define PORTA   (*((volatile uint8_t*)0x3B))  //1->high output              0->low output
00068 #define DDRA    (*((volatile uint8_t*)0x3A))  //1->to make it output       0->to make it input
00076 #define PINA    (*((volatile uint8_t*)0x39))  //this register to read a value from a pin
00080 /************************************************************ Port B registers
      ************************************************************/
00094 #define PORTB   (*((volatile uint8_t*)0x38))
00102 #define DDRB    (*((volatile uint8_t*)0x37))
00110 #define PINB    (*((volatile uint8_t*)0x36))
00112 /************************************************************ Port C registers
      ************************************************************/
00125 #define PORTC   (*((volatile uint8_t*)0x35))
00133 #define DDRC    (*((volatile uint8_t*)0x34))
00141 #define PINC    (*((volatile uint8_t*)0x33))
00143 /************************************************************ Port D registers
      ************************************************************/
00156 #define PORTD   (*((volatile uint8_t*)0x32))
00164 #define DDRD    (*((volatile uint8_t*)0x31))
00172 #define PIND    (*((volatile uint8_t*)0x30))
00174 /************************************************************ Interrupts registers
      ************************************************************/
00189 #define GICR    (*((volatile uint8_t*)0x5B))
00190
00200 #define GIFR    (*((volatile uint8_t*)0x5A))
00217 #define MCUCR   (*((volatile uint8_t*)0x55))
00218
00231 #define MCUCSR  (*((volatile uint8_t*)0x54))
00236 #endif /* REGISTERFILE_H_ */
```

Index