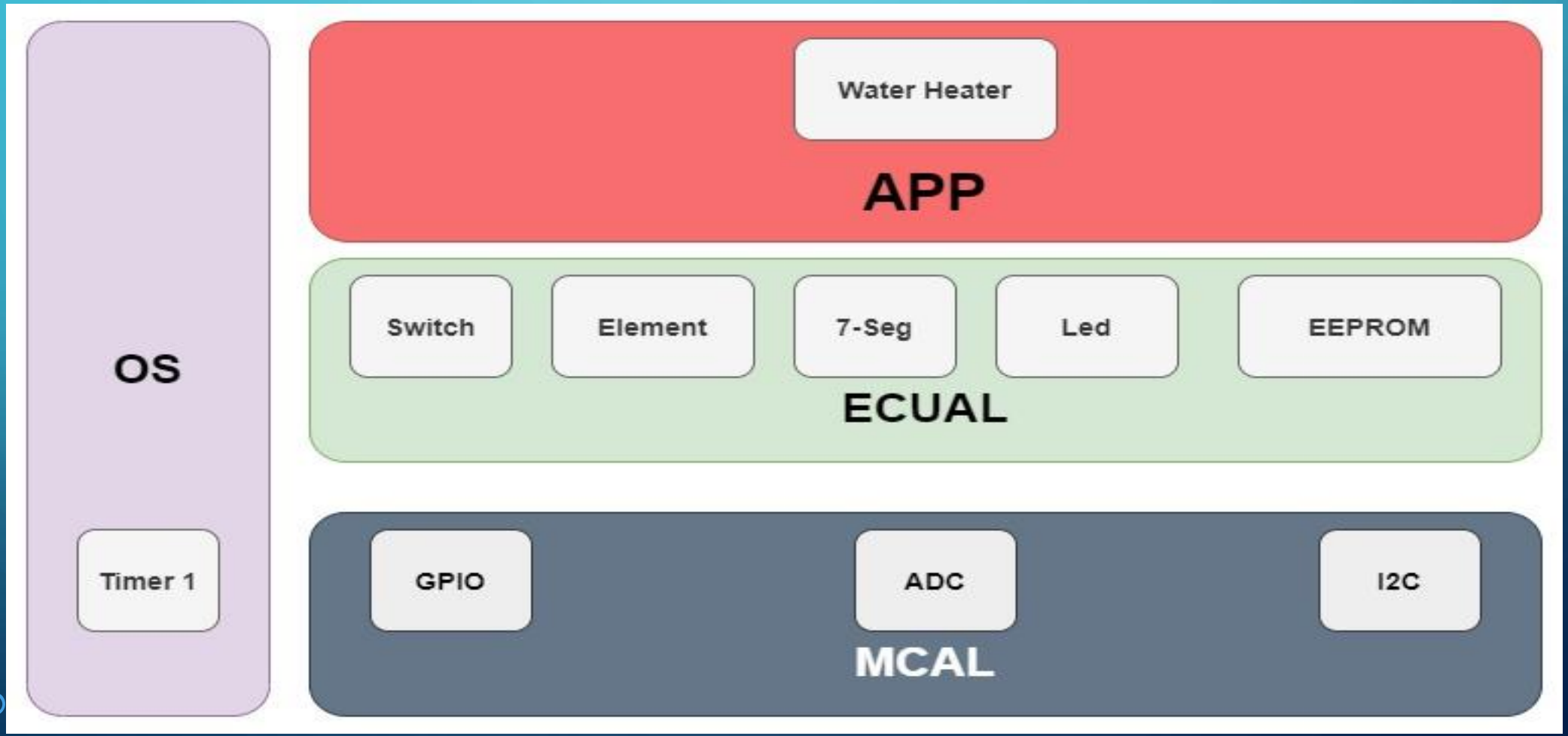
WATER HEATER

PIN LAYOUT

Function	Pin	Port
On/Off Button	5	B
Up Button	3	B
Down Button	4	B
Heating Led	7	B
*The Remaining Pins Are Defined By Default In Board 4 PicsimLab		

STATIC ARCHITECTURE



DYNAMIC ARCHITECTURE

Task	Periodicity (ms)	Priority (Least Value Is Higher Priority)	First Delay (Ticks)
WaterHeater_InitTask	/ Runs One Time Only	0	0
Switch_task	5	1	1
WaterHeater_Task	25	2	1
WaterHeater_CheckSwitches	25	2.0	1
WaterHeater_AddReading	100	2.1	1
WaterHeater_TakeAction	100	2.2	1
WaterHeater_UpdateCfgModeCounter	500	2.3	1
WaterHeater_Blink	500	2.4	1
SSeg_task	25	3	2
Clock Frequency		8 MHz	
Tick Time (ms)		5	
Major Cycle (ms)		25	

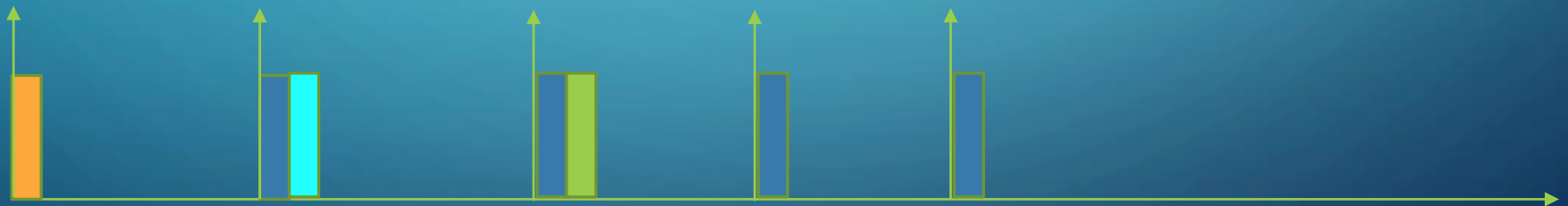
SCHEDULABILITY CHECK

WaterHeater_InitTask

WaterHeater_Task

Switch_task

SSeg_task



By:

Mark Joseph

Water Heater

Generated by Doxygen 1.8.17

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 element_t Struct Reference

Data Fields

- Gpio_Pins_t **pin**
- Gpio_Port_t **port**
- Gpio_PinStatus_t **activeState**

The documentation for this struct was generated from the following file:

- Element.h

3.2 gpio_t Struct Reference

Data Fields

- Gpio_Pins_t **pins**
- Gpio_Mode_t **mode**
- Gpio_Port_t **port**

The documentation for this struct was generated from the following file:

- Gpio.h

3.3 led_t Struct Reference

Data Fields

- Gpio_Pins_t **pin**
- Gpio_Port_t **port**
- Gpio_PinStatus_t **activeState**

The documentation for this struct was generated from the following file:

- Led.h

3.4 sseg_t Struct Reference

Data Fields

- Gpio_Pins_t **dPin** [SSEG_NUMBER_OF_PINS]
- Gpio_Port_t **dPort** [SSEG_NUMBER_OF_PINS]
- Gpio_Pins_t **enPin** [SSEG_NUMBER_OF_SSEGS]
- Gpio_Port_t **enPort** [SSEG_NUMBER_OF_SSEGS]
- uint8_t **common** [SSEG_NUMBER_OF_SSEGS]

The documentation for this struct was generated from the following file:

- SSeg.h

3.5 switch_t Struct Reference

Data Fields

- Gpio_Pins_t **pin**
- Gpio_Port_t **port**
- Gpio_PinStatus_t **activeState**

The documentation for this struct was generated from the following file:

- Switch.h

3.6 sysTask_t Struct Reference

Data Fields

- const sysTaskInfo_t * **taskInfo**
- uint32_t **remainToExec**
- uint32_t **periodTicks**
- uint8_t **state**
- uint32_t **sleepTimes**

The documentation for this struct was generated from the following file:

- Sched.c

3.7 sysTaskInfo_t Struct Reference

Data Fields

- const task_t * **task**
- uint32_t **delayTicks**

The documentation for this struct was generated from the following file:

- Sched.h

3.8 task_t Struct Reference

Data Fields

- taskRunnable_t **runnable**
- uint32_t **periodicTimeMS**

The documentation for this struct was generated from the following file:

- Sched.h

Chapter 4

File Documentation

4.1 Adc.c File Reference

The implementation for the ADC driver.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Adc.h"
```

Macros

- #define **ADC_CON0_REG** ∗(uint8_t∗)0x1F
- #define **ADC_CON1_REG** ∗(uint8_t∗)0x9F
- #define **ADC_DATA_H** ∗(uint8_t∗)0x1E
- #define **ADC_DATA_L** ∗(uint8_t∗)0x9E
- #define **ADC_CONV_DONE** 0x04
- #define **ADC_CONV_START** 0x04
- #define **ADC_CH_CLR** 0xC7
- #define **ADC_INIT_CONF_CON0** 0x01
- #define **ADC_INIT_CONF_CON1** 0x80

Functions

- Std_ReturnType Adc_Init (void)

  The ADC port and configurations initialization.
- Std_ReturnType Adc_GetValue (Adc_Value_t ∗value)

  Gets the value of a specific channel.
- Std_ReturnType Adc_SelectChannel (Adc_Channel_t channel)

  Selects An Adc Channel.

4.1.1 Detailed Description

The implementation for the ADC driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-09

Copyright

4.1.2 Function Documentation

4.1.2.1 Adc_GetValue()

```
Std_ReturnType Adc_GetValue (
            Adc_Value_t * value )
```

Gets the value of a specific channel.

Parameters

value	the value that will be returned

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.1.2.2 Adc_Init()

```
Std_ReturnType Adc_Init (
            void  )
```

The ADC port and configurations initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.1.2.3 Adc_SelectChannel()

```
Std_ReturnType Adc_SelectChannel (
            Adc_Channel_t channel )
```

Selects An Adc Channel.

Parameters

channel	The Channel To Be Selected • ADC_CH_x

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.2 Adc.h File Reference

This file is the user interface for the ADC driver.

Macros

- #define **ADC_CH_0** 0b000000
- #define **ADC_CH_1** 0b001000
- #define **ADC_CH_2** 0b010000
- #define **ADC_CH_3** 0b011000
- #define **ADC_CH_4** 0b100000
- #define **ADC_CH_5** 0b101000
- #define **ADC_CH_6** 0b110000
- #define **ADC_CH_7** 0b111000

Typedefs

- typedef uint16_t **Adc_Value_t**
- typedef uint8_t **Adc_Channel_t**

Functions

- Std_ReturnType Adc_Init (void)

  The ADC port and configurations initialization.
- Std_ReturnType Adc_GetValue (Adc_Value_t *value)

  Gets the value of a specific channel.
- Std_ReturnType Adc_SelectChannel (Adc_Channel_t channel)

  Selects An Adc Channel.

4.2.1 Detailed Description

This file is the user interface for the ADC driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-09

Copyright

4.2.2 Function Documentation

4.2.2.1 Adc_GetValue()

```
Std_ReturnType Adc_GetValue (
            Adc_Value_t * value )
```

Gets the value of a specific channel.

Parameters

value	the value that will be returned

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.2.2.2 Adc_Init()

```
Std_ReturnType Adc_Init (
            void  )
```

The ADC port and configurations initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.2.2.3 Adc_SelectChannel()

```
Std_ReturnType Adc_SelectChannel (
            Adc_Channel_t channel )
```

Selects An Adc Channel.

Parameters

channel	The Channel To Be Selected • ADC_CH_x

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.3 Eeprom.c File Reference

This is the implementation for the EEPROM driver.

```
#include "Std_Types.h"
#include "Eeprom.h"
#include "I2c.h"
```

Macros

- #define **EEPROM_READ** 0xA1
- #define **EEPROM_WRITE** 0xA0
- #define **EEPROM_SECOND_BYTE** 0x08

Functions

- Std_ReturnType Eeprom_Init (void)
  Initializes the EEPROM.
- Std_ReturnType Eeprom_WriteByte (Eeprom_Address_t address, uint8_t data)
  Writes a byte to the EEPROM.
- Std_ReturnType Eeprom_ReadByte (Eeprom_Address_t address, uint8_t *data)
  Reads a byte from the EEPROM.

4.3.1 Detailed Description

This is the implementation for the EEPROM driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

Copyright (c) 2020

4.3.2 Function Documentation

4.3.2.1 Eeprom_Init()

```
Std_ReturnType Eeprom_Init (
            void  )
```

Initializes the EEPROM.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.3.2.2 Eeprom_ReadByte()

```
Std_ReturnType Eeprom_ReadByte (
            Eeprom_Address_t address,
            uint8_t * data )
```

Reads a byte from the EEPROM.

Parameters

address	The address to read data from
data	The data to read

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.3.2.3 Eeprom_WriteByte()

```
Std_ReturnType Eeprom_WriteByte (
            Eeprom_Address_t address,
            uint8_t data )
```

Writes a byte to the EEPROM.

Parameters

address	The address to write data in
data	The data to write

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.4 Eeprom.h File Reference

This is the user interface for the EEPROM driver.

Typedefs

- typedef uint16_t **Eeprom_Address_t**

Functions

- Std_ReturnType Eeprom_Init (void)

  Initializes the EEPROM.
- Std_ReturnType Eeprom_WriteByte (Eeprom_Address_t address, uint8_t data)

  Writes a byte to the EEPROM.
- Std_ReturnType Eeprom_ReadByte (Eeprom_Address_t address, uint8_t *data)

  Reads a byte from the EEPROM.

4.4.1 Detailed Description

This is the user interface for the EEPROM driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.4.2 Function Documentation

4.4.2.1 Eeprom_Init()

```
Std_ReturnType Eeprom_Init (
            void  )
```

Initializes the EEPROM.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.4.2.2 Eeprom_ReadByte()

```
Std_ReturnType Eeprom_ReadByte (
            Eeprom_Address_t address,
            uint8_t * data )
```

Reads a byte from the EEPROM.

Parameters

address	The address to read data from
data	The data to read

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.4.2.3 Eeprom_WriteByte()

```
Std_ReturnType Eeprom_WriteByte (
            Eeprom_Address_t address,
            uint8_t data )
```

Writes a byte to the EEPROM.

Parameters

address	The address to write data in
data	The data to write

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.5 Element.c File Reference

This file is to be used as an implementation for the Element Handler.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Element.h"
```

Functions

- Std_ReturnType Element_Init (void)
  Initializes GPIOs for the ELEMENTs.
- Std_ReturnType Element_SetElementOn (Element_Name_t elementName)
  Sets the Element on.
- Std_ReturnType Element_SetElementOff (Element_Name_t elementName)
  Sets the Element off.
- Std_ReturnType Element_SetElementStatus (Element_Name_t elementName, Element_State_t status)
  Sets the Element off.

Variables

- const element_t **Element_elements** [ELEMENT_NUMBER_OF_ELEMENTS]

4.5.1 Detailed Description

This file is to be used as an implementation for the Element Handler.

Author

Mark Attia

Date

January 22, 2020

4.5.2 Function Documentation

4.5.2.1 Element_Init()

```
Std_ReturnType Element_Init (
            void  )
```

Initializes GPIOs for the ELEMENTs.

4.5.2.2 Function: Element_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.5.2.3 Element_SetElementOff()

```
Std_ReturnType Element_SetElementOff (
            Element_Name_t elementName )
```

Sets the Element off.

4.5.2.4 Function: Element_SetElementOff

Parameters

elementName	The name of the ELEMENT

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.5.2.5 Element_SetElementOn()

```
Std_ReturnType Element_SetElementOn (
            Element_Name_t elementName )
```

Sets the Element on.

4.5.2.6 Function: Element_SetElementOn

Parameters

elementName	The name of the Element

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.5.2.7 Element_SetElementStatus()

```
Std_ReturnType Element_SetElementStatus (
            Element_Name_t elementName,
            Element_State_t status )
```

Sets the Element off.

4.5.2.8 Function: Element_SetElementStatus

Parameters

elementName	The name of the ELEMENT
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • ELEMENT_ON : Sets the pin value to 1 • ELEMENT_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6 Element.h File Reference

This file is to be used as an interface for the user of the Element Handler.

```
#include "Element_Cfg.h"
```

Data Structures

- struct element_t

Macros

- #define **ELEMENT_ON** 0
- #define **ELEMENT_OFF** 1

Typedefs

- typedef uint8_t **Element_Name_t**
- typedef uint8_t **Element_State_t**

Functions

- Std_ReturnType Element_Init (void)

  Initializes GPIOs for the ELEMENTs.
- Std_ReturnType Element_SetElementOn (Element_Name_t elementName)

  Sets the Element on.
- Std_ReturnType Element_SetElementOff (Element_Name_t elementName)

  Sets the Element off.
- Std_ReturnType Element_SetElementStatus (Element_Name_t elementName, Element_State_t status)

  Sets the Element off.

4.6.1 Detailed Description

This file is to be used as an interface for the user of the Element Handler.

Author

Mark Attia

Date

January 22, 2020

4.6.2 Function Documentation

4.6.2.1 Element_Init()

```
Std_ReturnType Element_Init (
            void  )
```

Initializes GPIOs for the ELEMENTs.

4.6.2.2 Function: Element_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.3 Function: Element_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.4 Element_SetElementOff()

```
Std_ReturnType Element_SetElementOff (
            Element_Name_t elementName )
```

Sets the Element off.

4.6.2.5 Function: Element_SetElementOff

Parameters

elementName	The name of the ELEMENT

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.6 Function: Element_SetElementOff

Parameters

elementName	The name of the ELEMENT

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.7 Element_SetElementOn()

```
Std_ReturnType Element_SetElementOn (
            Element_Name_t elementName )
```

Sets the Element on.

4.6.2.8 Function: Element_SetElementOn

Parameters

elementName	The name of the ELEMENT

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.9 Function: Element_SetElementOn

Parameters

elementName	The name of the Element

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.10 Element_SetElementStatus()

```
Std_ReturnType Element_SetElementStatus (
            Element_Name_t elementName,
            Element_State_t status )
```

Sets the Element off.

4.6.2.11 Function: Element_SetElementStatus

Parameters

elementName	The name of the ELEMENT
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • ELEMENT_ON : Sets the pin value to 1 • ELEMENT_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.6.2.12 Function: Element_SetElementStatus

Parameters

elementName	The name of the ELEMENT
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • ELEMENT_ON : Sets the pin value to 1 • ELEMENT_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.7 Element_Cfg.h File Reference

This file is to be given to the user to configure the Element Handler.

Macros

- #define **ELEMENT_NUMBER_OF_ELEMENTS** 2
- #define **WATER_HEATER_HEATING_ELEMENT** 0
- #define **WATER_HEATER_COOLING_ELEMENT** 1

4.7.1 Detailed Description

This file is to be given to the user to configure the Element Handler.

Author

Mark Attia

Date

January 22, 2020

4.8 Gpio.c File Reference

This file is to be used as an implementation of the GPIO driver.

```
#include "Std_Types.h"
#include "Gpio.h"
```

Macros

- #define **GPIO_TRIS** 0x80
- #define **GPIO_PORTB_PULLUP_CLR** 0x10
- #define **GPIO_OPTION_REG** 0x81

Functions

- Std_ReturnType Gpio_InitPins (gpio_t *gpio)
  Initializes pins mode and speed for a specific port.
- Std_ReturnType Gpio_WritePin (Gpio_Port_t port, Gpio_Pins_t pin, Gpio_PinStatus_t pinStatus)
  Write a value to a pin(0/1)
- Std_ReturnType Gpio_ReadPin (Gpio_Port_t port, Gpio_Pins_t pin, Gpio_PinStatus_t *state)
  Reads a value to a pin(0/1)
- Std_ReturnType Gpio_SetPortBPullup (Gpio_PullupStatus_t pullupState)
  Sets Port B Pullup State.

4.8.1 Detailed Description

This file is to be used as an implementation of the GPIO driver.

Author

Mark Attia

Date

February 6, 2020

4.8.2 Function Documentation

4.8.2.1 Gpio_InitPins()

```
Std_ReturnType Gpio_InitPins (
            gpio_t * gpio )
```

Initializes pins mode and speed for a specific port.

4.8.2.2 Function: Gpio_InitPins

Parameters

gpio	An object of type gpio_t to set pins for

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.8.2.3 Gpio_ReadPin()

```
Std_ReturnType Gpio_ReadPin (
            Gpio_Port_t port,
            Gpio_Pins_t pin,
            Gpio_PinStatus_t * state )
```

Reads a value to a pin(0/1)

4.8.2.4 Function: Gpio_ReadPin

Parameters

port	The port you want to read from • GPIO_PORTX : The pin number you want to read from
pin	The pin you want to read • GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
state	To return a status in • GPIO_PIN_SET : The pin is set to 1 • GPIO_PIN_RESET : The pin is set to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.8.2.5 Gpio_SetPortBPullup()

```
Std_ReturnType Gpio_SetPortBPullup (
            Gpio_PullupStatus_t pullupState )
```

Sets Port B Pullup State.

4.8.2.6 Function: Gpio_SetPortBPullup

Parameters

pullupState	The port you want to read from • GPIO_PORTB_PULLUP_EN • GPIO_PORTB_PULLUP_DIS

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.8.2.7 Gpio_WritePin()

```
Std_ReturnType Gpio_WritePin (
            Gpio_Port_t port,
            Gpio_Pins_t pin,
            Gpio_PinStatus_t pinStatus )
```

Write a value to a pin(0/1)

4.8.2.8 Function: Gpio_WritePin

Parameters

port	The port you want to configure • GPIO_PORTX : The pin number you want to configure
pin	The pin you want to configure • GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
pinStatus	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) • GPIO_PIN_SET : Sets the pin value to 1 • GPIO_PIN_RESET : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9 Gpio.h File Reference

This file is to be used as an interface for the user of GPIO driver.

Data Structures

- struct gpio_t

Macros

- #define **GPIO_PIN_SET** 0
- #define **GPIO_PIN_RESET** !GPIO_PIN_SET
- #define **GPIO_PIN_0** 0x01
- #define **GPIO_PIN_1** 0x02
- #define **GPIO_PIN_2** 0x04
- #define **GPIO_PIN_3** 0x08
- #define **GPIO_PIN_4** 0x10
- #define **GPIO_PIN_5** 0x20
- #define **GPIO_PIN_6** 0x40
- #define **GPIO_PIN_7** 0x80
- #define **GPIO_PIN_ALL** 0xFF
- #define **GPIO_MODE_OUTPUT_PP** 0
- #define **GPIO_MODE_INPUT** 1
- #define **GPIO_PORTA** 0x05
- #define **GPIO_PORTB** 0x06
- #define **GPIO_PORTC** 0x07
- #define **GPIO_PORTD** 0x08
- #define **GPIO_PORTE** 0x09
- #define **GPIO_PORTB_PULLUP_EN** 0x7F
- #define **GPIO_PORTB_PULLUP_DIS** 0xFF

Typedefs

- typedef uint8_t **Gpio_Pins_t**
- typedef uint8_t **Gpio_Mode_t**
- typedef uint8_t **Gpio_Port_t**
- typedef uint8_t **Gpio_PinStatus_t**
- typedef uint8_t **Gpio_PullupStatus_t**

Functions

- Std_ReturnType Gpio_InitPins (gpio_t *gpio)

  Initializes pins mode and speed for a specific port.
- Std_ReturnType Gpio_WritePin (Gpio_Port_t port, Gpio_Pins_t pin, Gpio_PinStatus_t pinStatus)

  Write a value to a pin(0/1)
- Std_ReturnType Gpio_ReadPin (Gpio_Port_t port, Gpio_Pins_t pin, Gpio_PinStatus_t *state)

  Reads a value to a pin(0/1)
- Std_ReturnType Gpio_SetPortBPullup (Gpio_PullupStatus_t pullupState)

  Sets Port B Pullup State.

4.9.1 Detailed Description

This file is to be used as an interface for the user of GPIO driver.

Author

Mark Attia

Date

February 6, 2020

4.9.2 Function Documentation

4.9.2.1 Gpio_InitPins()

```
Std_ReturnType Gpio_InitPins (
            gpio_t * gpio )
```

Initializes pins mode and speed for a specific port.

4.9.2.2 Function: Gpio_InitPins

Parameters

gpio	An object of type gpio_t to set pins for

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.3 Function: Gpio_InitPins

Parameters

gpio	An object of type gpio_t to set pins for

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.4 Gpio_ReadPin()

```
Std_ReturnType Gpio_ReadPin (
            Gpio_Port_t port,
            Gpio_Pins_t pin,
            Gpio_PinStatus_t * state )
```

Reads a value to a pin(0/1)

4.9.2.5 Function: Gpio_ReadPin

Parameters

port	The port you want to read from • GPIO_PORTX : The pin number you want to read from
pin	The pin you want to read • GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
state	To return a status in • GPIO_PIN_SET : The pin is set to 1 • GPIO_PIN_RESET : The pin is set to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.6 Function: Gpio_ReadPin

Parameters

port	The port you want to read from • GPIO_PORTX : The pin number you want to read from
pin	The pin you want to read • GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
state	To return a status in • GPIO_PIN_SET : The pin is set to 1 • GPIO_PIN_RESET : The pin is set to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.7 Gpio_SetPortBPullup()

```
Std_ReturnType Gpio_SetPortBPullup (
            Gpio_PullupStatus_t pullupState )
```

Sets Port B Pullup State.

4.9.2.8 Function: Gpio_SetPortBPullup

Parameters

pullupState	The port you want to read from • GPIO_PORTB_PULLUP_EN • GPIO_PORTB_PULLUP_DIS

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.9 Function: Gpio_SetPortBPullup

Parameters

pullupState	The port you want to read from • GPIO_PORTB_PULLUP_EN • GPIO_PORTB_PULLUP_DIS

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.10 Gpio_WritePin()

```
Std_ReturnType Gpio_WritePin (
            Gpio_Port_t port,
            Gpio_Pins_t pin,
            Gpio_PinStatus_t pinStatus )
```

Write a value to a pin(0/1)

4.9.2.11 Function: Gpio_WritePin

Parameters

port	The port you want to configure • GPIO_PORTX : The pin number you want to configure
pin	The pin you want to configure • GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
pinStatus	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) • GPIO_PIN_SET : Sets the pin value to 1 • GPIO_PIN_RESET : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.9.2.12 Function: Gpio_WritePin

Parameters

port	The port you want to configure • GPIO_PORTX : The pin number you want to configure
pin	The pin you want to configure • GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
pinStatus	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) • GPIO_PIN_SET : Sets the pin value to 1 • GPIO_PIN_RESET : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10 I2c.c File Reference

This is the implementation for the I2C Driver.

```
#include "Std_Types.h"
#include "I2c.h"
#include "I2c_Cfg.h"
#include "Gpio.h"
```

Macros

- #define **I2C_SSPBUF** ∗(uint8_t∗)0x13
- #define **I2C_SSPCON** ∗(uint8_t∗)0x14
- #define **I2C_SSPCON2** ∗(uint8_t∗)0x91
- #define **I2C_SSPADD** ∗(uint8_t∗)0x93
- #define **I2C_SSPSTAT** ∗(uint8_t∗)0x94
- #define **INTERRUPT_PIR1** ∗(uint8_t∗)0x0C
- #define **INTERRUPT_SSPIF** 0x08
- #define **INTERRUPT_SSPIF_CLR** 0xF7
- #define **I2C_SSPCON_CONF** 0x28
- #define **I2C_SSPCON2_CONF** 0x00
- #define **I2C_SSPSTAT_CONF** 0x00
- #define **I2C_READABLE** 0x04
- #define **I2C_SEN** 0x01
- #define **I2C_PEN** 0x04
- #define **I2C_RCEN** 0x08
- #define **I2C_SSPCON2_EN** 0x1F
- #define **I2C_ACK_EN** 0x10
- #define **I2C_ACK_STAT** 0x40
- #define **I2C_ACK_DT_CLR** 0xEF
- #define **I2C_ACK_DT** 0x00
- #define **I2C_NO_ACK_DT** 0x10

Functions

- Std_ReturnType I2C_Master_Init (void)

  I2C Initialization.
- Std_ReturnType I2c_Start (void)

  I2C Start of frame.
- Std_ReturnType I2c_Stop (void)

  I2C Stop of frame.
- Std_ReturnType I2c_ACK (void)

  I2C Send Ack.
- Std_ReturnType I2c_NACK (void)

  I2C Send No Ack.
- Std_ReturnType I2c_Read (uint8_t ∗data)

  I2C Reads A Byte of Data.
- Std_ReturnType I2c_Write (uint8_t ∗ack, uint8_t data)

  I2C Writes A Byte.

4.10.1 Detailed Description

This is the implementation for the I2C Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.10.2 Function Documentation

4.10.2.1 I2c_ACK()

```
Std_ReturnType I2c_ACK (
            void  )
```

I2C Send Ack.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.2 I2C_Master_Init()

```
Std_ReturnType I2C_Master_Init (
            void  )
```

I2C Initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.3 I2c_NACK()

```
Std_ReturnType I2c_NACK (
            void  )
```

I2C Send No Ack.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.4 I2c_Read()

```
Std_ReturnType I2c_Read (
            uint8_t * data )
```

I2C Reads A Byte of Data.

Parameters

data	The data to be stored

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.5 I2c_Start()

```
Std_ReturnType I2c_Start (
            void  )
```

I2C Start of frame.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.6 I2c_Stop()

```
Std_ReturnType I2c_Stop (
            void  )
```

I2C Stop of frame.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.10.2.7 I2c_Write()

```
Std_ReturnType I2c_Write (
            uint8_t * ack,
            uint8_t data )
```

I2C Writes A Byte.

Parameters

ack	The Ack Returned • I2C_ACK • I2C_NO_ACK
data	The Data to Be Written

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11 I2c.h File Reference

This is the user interface for the I2C Driver.

Macros

- #define **I2C_NO_ACK** 0
- #define **I2C_ACK** !I2C_NO_ACK

Functions

- Std_ReturnType I2C_Master_Init (void)

  I2C Initialization.
- Std_ReturnType I2c_Start (void)

  I2C Start of frame.
- Std_ReturnType I2c_Stop (void)

  I2C Stop of frame.
- Std_ReturnType I2c_ACK (void)

  I2C Send Ack.
- Std_ReturnType I2c_NACK (void)

  I2C Send No Ack.
- Std_ReturnType I2c_Read (uint8_t *data)

  I2C Reads A Byte of Data.
- Std_ReturnType I2c_Write (uint8_t *ack, uint8_t data)

  I2C Writes A Byte.

4.11.1 Detailed Description

This is the user interface for the I2C Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

Copyright (c) 2020

4.11.2 Function Documentation

4.11.2.1 I2c_ACK()

```
Std_ReturnType I2c_ACK (
            void  )
```

I2C Send Ack.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.2 I2C_Master_Init()

```
Std_ReturnType I2C_Master_Init (
            void  )
```

I2C Initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.3 I2c_NACK()

```
Std_ReturnType I2c_NACK (
            void  )
```

I2C Send No Ack.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.4 I2c_Read()

```
Std_ReturnType I2c_Read (
            uint8_t * data )
```

I2C Reads A Byte of Data.

Parameters

data	The data to be stored

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.5 I2c_Start()

```
Std_ReturnType I2c_Start (
            void  )
```

I2C Start of frame.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.6 I2c_Stop()

```
Std_ReturnType I2c_Stop (
            void  )
```

I2C Stop of frame.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.11.2.7 I2c_Write()

```
Std_ReturnType I2c_Write (
            uint8_t * ack,
            uint8_t data )
```

I2C Writes A Byte.

Parameters

ack	The Ack Returned • I2C_ACK • I2C_NO_ACK
data	The Data to Be Written

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.12 I2c_Cfg.h File Reference

This is the user's Configurations for the I2C Driver.

Macros

- #define **I2C_BaudRate** 100000
- #define **I2C_CLK_FREQ** 8000000

4.12.1 Detailed Description

This is the user's Configurations for the I2C Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.13 Int.c File Reference

This is the implementation for the interrupts.

```
#include "Std_Types.h"
#include "Int.h"
```

Macros

- #define **PIF** *(uint8_t*)0x0C
- #define **CCP1_INT_FLAG** 0x04
- #define **CCP1_INT_FLAG_CLR** 0xFB

Functions

- void __interrupt () ISR()

  Global Interrupt Service Routine.

Variables

- interruptCb_t **Timer1_func** = NULL

4.13.1 Detailed Description

This is the implementation for the interrupts.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-15

Copyright

4.13.2 Function Documentation

4.13.2.1 __interrupt()

```
void __interrupt ( )
```

Global Interrupt Service Routine.

4.14 Int.h File Reference

This is the user interface for the interrupts.

Typedefs

- typedef void(∗ **interruptCb_t**) (void)

Variables

- interruptCb_t **Timer1_func**

4.14.1 Detailed Description

This is the user interface for the interrupts.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-15

Copyright

Copyright (c) 2020

4.15 Led.c File Reference

This file is to be used as an implementation for the Led Handler.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Led.h"
```

Functions

- Std_ReturnType Led_Init (void)

  Initializes GPIOs for the LEDs.
- Std_ReturnType Led_SetLedOn (Led_Name_t ledName)

  Sets the Led on.
- Std_ReturnType Led_SetLedOff (Led_Name_t ledName)

  Sets the Led off.
- Std_ReturnType Led_SetLedStatus (Led_Name_t ledName, Led_State_t status)

  Sets the Led off.

Variables

- const led_t **Led_leds** [LED_NUMBER_OF_LEDS]

4.15.1 Detailed Description

This file is to be used as an implementation for the Led Handler.

Author

Mark Attia

Date

January 22, 2020

4.15.2 Function Documentation

4.15.2.1 Led_Init()

```
Std_ReturnType Led_Init (
             void  )
```

Initializes GPIOs for the LEDs.

4.15.2.2 Function: Led_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.15.2.3 Led_SetLedOff()

```
Std_ReturnType Led_SetLedOff (
             Led_Name_t ledName )
```

Sets the Led off.

4.15.2.4 Function: Led_SetLedOff

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.15.2.5 Led_SetLedOn()

```
Std_ReturnType Led_SetLedOn (
            Led_Name_t ledName )
```

Sets the Led on.

4.15.2.6 Function: Led_SetLedOn

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.15.2.7 Led_SetLedStatus()

```
Std_ReturnType Led_SetLedStatus (
            Led_Name_t ledName,
            Led_State_t status )
```

Sets the Led off.

4.15.2.8 Function: Led_SetLedStatus

Parameters

ledName	The name of the LED
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • LED_ON : Sets the pin value to 1 • LED_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16 Led.h File Reference

This file is to be used as an interface for the user of the Led Handler.

```
#include "Led_Cfg.h"
```

Data Structures

- struct led_t

Macros

- #define **LED_ON** 0
- #define **LED_OFF** 1

Typedefs

- typedef uint8_t **Led_Name_t**
- typedef uint8_t **Led_State_t**

Functions

- Std_ReturnType Led_Init (void)
  Initializes GPIOs for the LEDs.
- Std_ReturnType Led_SetLedOn (Led_Name_t ledName)
  Sets the Led on.
- Std_ReturnType Led_SetLedOff (Led_Name_t ledName)
  Sets the Led off.
- Std_ReturnType Led_SetLedStatus (Led_Name_t ledName, Led_State_t status)
  Sets the Led off.

4.16.1 Detailed Description

This file is to be used as an interface for the user of the Led Handler.

Author

Mark Attia

Date

January 22, 2020

4.16.2 Function Documentation

4.16.2.1 Led_Init()

```
Std_ReturnType Led_Init (
            void  )
```

Initializes GPIOs for the LEDs.

4.16.2.2 Function: Led_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.3 Function: Led_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.4 Led_SetLedOff()

```
Std_ReturnType Led_SetLedOff (
            Led_Name_t ledName )
```

Sets the Led off.

4.16.2.5 Function: Led_SetLedOff

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.6 Function: Led_SetLedOff

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.7 Led_SetLedOn()

```
Std_ReturnType Led_SetLedOn (
            Led_Name_t ledName )
```

Sets the Led on.

4.16.2.8 Function: Led_SetLedOn

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.9 Function: Led_SetLedOn

Parameters

ledName	The name of the LED

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.10 Led_SetLedStatus()

```
Std_ReturnType Led_SetLedStatus (
            Led_Name_t ledName,
            Led_State_t status )
```

Sets the Led off.

4.16.2.11 Function: Led_SetLedStatus

Parameters

ledName	The name of the LED
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • LED_ON : Sets the pin value to 1 • LED_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.16.2.12 Function: Led_SetLedStatus

Parameters

ledName	The name of the LED
pinStatus	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) • LED_ON : Sets the pin value to 1 • LED_OFF : Resets the pin value to 0

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.17 Led_Cfg.h File Reference

This file is to be given to the user to configure the Led Handler.

Macros

- #define **LED_NUMBER_OF_LEDS** 1
- #define **WATER_HEATER_HEATING_LED** 0

4.17.1 Detailed Description

This file is to be given to the user to configure the Led Handler.

Author

Mark Attia

Date

January 22, 2020

4.18 Sched.c File Reference

This file is the implementation for the Scheduler.

```
#include "Std_Types.h"
#include "Sched_Cfg.h"
#include "Sched.h"
#include "Int.h"
#include "Timer1.h"
```

Data Structures

- struct sysTask_t

Macros

- #define **SCHED_TASK_RUNNING** 1
- #define **SCHED_TASK_SUSPENDED** 2
- #define **FLAG_RAISED** 1
- #define **FLAG_LOWERED** 0

Typedefs

- typedef uint8_t **Sched_Flag_t**

Functions

- void Sched_Start (void)

  The scheduler that will run all the time.
- Std_ReturnType Sched_Init (void)

  The initialization for the Scheduler.
- Std_ReturnType Sched_SuspendTask (void)

  Suspends a running task.
- Std_ReturnType Sched_Sleep (uint32_t timeMS)

  Makes a task sleep for a while.

Variables

- const sysTaskInfo_t **Sched_sysTaskInfo** [SCHED_NUMBER_OF_TASKS]

4.18.1 Detailed Description

This file is the implementation for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

4.18.2 Function Documentation

4.18.2.1 Sched_Init()

```
Std_ReturnType Sched_Init (
            void  )
```

The initialization for the Scheduler.

Returns

Std_ReturnType

4.18.2.2 Sched_Sleep()

```
Std_ReturnType Sched_Sleep (
            uint32_t timeMS )
```

Makes a task sleep for a while.

Parameters

timeMS	The sleep time in milli seconds

Returns

Std_ReturnType E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.18.2.3 Sched_Start()

```
void Sched_Start (
             void  )
```

The scheduler that will run all the time.

4.18.2.4 Sched_SuspendTask()

```
Std_ReturnType Sched_SuspendTask (
             void  )
```

Suspends a running task.

Returns

Std_ReturnType E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.19 Sched.h File Reference

This file is the user interface for the Scheduler.

Data Structures

- struct task_t
- struct sysTaskInfo_t

Typedefs

- typedef void(* **taskRunnable_t**) (void)

Functions

- void Sched_Start (void)
  The scheduler that will run all the time.
- Std_ReturnType Sched_Init (void)
  The initialization for the Scheduler.
- Std_ReturnType Sched_SuspendTask (void)
  Suspends a running task.
- Std_ReturnType Sched_Sleep (uint32_t timeMS)
  Makes a task sleep for a while.

4.19.1 Detailed Description

This file is the user interface for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

4.19.2 Function Documentation

4.19.2.1 Sched_Init()

```
Std_ReturnType Sched_Init (
            void  )
```

The initialization for the Scheduler.

Returns

Std_ReturnType

4.19.2.2 Sched_Sleep()

```
Std_ReturnType Sched_Sleep (
            uint32_t timeMS )
```

Makes a task sleep for a while.

Parameters

timeMS	The sleep time in milli seconds

Returns

Std_ReturnType E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.19.2.3 Sched_Start()

```
void Sched_Start (
             void  )
```

The scheduler that will run all the time.

4.19.2.4 Sched_SuspendTask()

```
Std_ReturnType Sched_SuspendTask (
             void  )
```

Suspends a running task.

Returns

Std_ReturnType E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.20 Sched_Cfg.c File Reference

This file contains the configurations implementation for the Scheduler.

```
#include "Std_Types.h"
#include "Sched_Cfg.h"
#include "Sched.h"
```

Variables

- const task_t **WaterHeater_InitTask**
- const task_t **WaterHeater_Task**
- const task_t **SSeg_task**
- const task_t **Switch_task**
- const sysTaskInfo_t **Sched_sysTaskInfo** [SCHED_NUMBER_OF_TASKS]

4.20.1 Detailed Description

This file contains the configurations implementation for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

Copyright (c) 2020

4.20.2 Variable Documentation

4.20.2.1 Sched_sysTaskInfo

```
const sysTaskInfo_t Sched_sysTaskInfo[SCHED_NUMBER_OF_TASKS]
```

Initial value:

```
=
{

    {&WaterHeater_InitTask,             0      },
    {&Switch_task,                      1     },
    {&WaterHeater_Task,                 1     },
    {&SSeg_task,                        2     }
}
```

4.21 Sched_Cfg.h File Reference

This file contains the configurations for the Scheduler.

Macros

- #define **SCHED_NUMBER_OF_TASKS** 4
- #define **SCHED_TICK_TIME_MS** 5
- #define **SCHED_SYS_CLK** 2000000

4.21.1 Detailed Description

This file contains the configurations for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

4.22 SSeg.c File Reference

This is the implementation for the Seven Segment Display Driver.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "SSeg.h"
#include "Sched.h"
```

Functions

- Std_ReturnType SSeg_Init (void)

  The Seven Segment initialization.
- Std_ReturnType SSeg_SetDisplay (SSeg_display_t display)

  Sets The Seven Segments Display On And Off.
- Std_ReturnType SSeg_SetNum (SSeg_name_t name, uint8_t digit)

  Sets A Digit For A Specific Seven Segment.
- void SSeg_Runnable (void)

  The Seven Segment Runnable.

Variables

- const char **numsA** [10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F}
- const char **numsC** [10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90}
- const sseg_t **SSeg_sseg**
- const task_t **SSeg_task** = {SSeg_Runnable, 25}

4.22.1 Detailed Description

This is the implementation for the Seven Segment Display Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.22.2 Function Documentation

4.22.2.1 SSeg_Init()

```
Std_ReturnType SSeg_Init (
            void  )
```

The Seven Segment initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.22.2.2 SSeg_Runnable()

```
void SSeg_Runnable (
            void  )
```

The Seven Segment Runnable.

4.22.2.3 SSeg_SetDisplay()

```
Std_ReturnType SSeg_SetDisplay (
            SSeg_display_t display )
```

Sets The Seven Segments Display On And Off.

Parameters

display	The Display State • SSEG_ON • SSEG_OFF

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.22.2.4 SSeg_SetNum()

```
Std_ReturnType SSeg_SetNum (
            SSeg_name_t name,
            uint8_t digit )
```

Sets A Digit For A Specific Seven Segment.

Parameters

name	The Name Of The Seven Segment
digit	The Digit To Set

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.23 SSeg.h File Reference

This is the user interface for the Seven Segment Display Driver.

```
#include "SSeg_Cfg.h"
```

Data Structures

- struct sseg_t

Macros

- #define **SSEG_COMMON_ANODE** 'A'
- #define **SSEG_COMMON_CATHODE** 'C'
- #define **SSEG_ON** 0
- #define **SSEG_OFF** !SSEG_ON

Typedefs

- typedef uint8_t **SSeg_display_t**
- typedef uint8_t **SSeg_name_t**

Functions

- Std_ReturnType SSeg_Init (void)

  The Seven Segment initialization.
- Std_ReturnType SSeg_SetNum (SSeg_name_t name, uint8_t digit)

  Sets A Digit For A Specific Seven Segment.
- Std_ReturnType SSeg_SetDisplay (SSeg_display_t display)

  Sets The Seven Segments Display On And Off.

4.23.1 Detailed Description

This is the user interface for the Seven Segment Display Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

Copyright (c) 2020

4.23.2 Function Documentation

4.23.2.1 SSeg_Init()

```
Std_ReturnType SSeg_Init (
            void  )
```

The Seven Segment initialization.

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.23.2.2 SSeg_SetDisplay()

```
Std_ReturnType SSeg_SetDisplay (
            SSeg_display_t display )
```

Sets The Seven Segments Display On And Off.

Parameters

display	The Display State • SSEG_ON • SSEG_OFF

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.23.2.3 SSeg_SetNum()

```
Std_ReturnType SSeg_SetNum (
            SSeg_name_t name,
            uint8_t digit )
```

Sets A Digit For A Specific Seven Segment.

Parameters

name	The Name Of The Seven Segment
digit	The Digit To Set

Returns

Std_ReturnType A Status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.24 SSeg_Cfg.c File Reference

These are the configurations for the Seven Segment Display Driver.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "SSeg_Cfg.h"
#include "SSeg.h"
```

Variables

- const sseg_t **SSeg_sseg**

4.24.1 Detailed Description

These are the configurations for the Seven Segment Display Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

Copyright (c) 2020

4.24.2 Variable Documentation

4.24.2.1 SSeg_sseg

```
const sseg_t SSeg_sseg
```

Initial value:

```
= {
    .dPin = { GPIO_PIN_0, GPIO_PIN_1, GPIO_PIN_2, GPIO_PIN_3, GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6},
    .dPort = { GPIO_PORTD, GPIO_PORTD, GPIO_PORTD, GPIO_PORTD, GPIO_PORTD, GPIO_PORTD, GPIO_PORTD},
    .enPin = { GPIO_PIN_4, GPIO_PIN_5},
    .enPort = { GPIO_PORTA, GPIO_PORTA},
    .common = {SSEG_COMMON_ANODE, SSEG_COMMON_ANODE}
}
```

4.25 SSeg_Cfg.h File Reference

This is The Configurations header for the Seven Segment Display Driver.

Macros

- #define **SSEG_NUMBER_OF_SSEGS** 2
- #define **SSEG_NUMBER_OF_PINS** 7
- #define **SSEG_TENS** 0
- #define **SSEG_ONES** 1

4.25.1 Detailed Description

This is The Configurations header for the Seven Segment Display Driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.26 Std_Types.h File Reference

The Standard Types.

Macros

- #define **NULL** ((void∗)0)
- #define **E_OK** (0)
- #define **E_NOT_OK** (1)
- #define **STD_LOW** (0)
- #define **STD_HIGH** (1)
- #define **STD_IDLE** (0)
- #define **STD_ACTIVE** (1)
- #define **STD_OFF** (0)
- #define **STD_ON** (1)

Typedefs

- typedef unsigned char **u8**
- typedef unsigned char **uint8_t**
- typedef signed char **s8**
- typedef signed char **sint8_t**
- typedef unsigned int **u16**
- typedef unsigned int **uint16_t**
- typedef signed short int **s16**
- typedef signed short int **sint16_t**
- typedef unsigned long int **u32**
- typedef unsigned long int **uint32_t**
- typedef signed long int **s32**
- typedef signed long int **sint32_t**
- typedef unsigned long long int **u64**
- typedef unsigned long long int **uint64_t**
- typedef signed long long int **s64**
- typedef signed long long int **sint64_t**
- typedef float **f32**
- typedef float **float32_t**
- typedef double **f64**
- typedef double **float64_t**
- typedef uint8_t **Std_ReturnType**

4.26.1 Detailed Description

The Standard Types.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.27 Switch.c File Reference

This file is to be used as an implementation for the Switch Handler.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Switch.h"
#include "Sched.h"
```

Functions

- Std_ReturnType Switch_Init (void)

  Initializes GPIOs for the Switches.
- Std_ReturnType Switch_GetSwitchStatus (Switch_Name_t switchName, Switch_State_t *state)

  Gets the status of the switch.

Variables

- const switch_t **Switch_switches** [SWITCH_NUMBER_OF_SWITCHES]
- const task_t **Switch_task** = {Switch_Runnable, 5}

4.27.1 Detailed Description

This file is to be used as an implementation for the Switch Handler.

Author

Mark Attia

Date

January 22, 2020

4.27.2 Function Documentation

4.27.2.1 Switch_GetSwitchStatus()

```
Std_ReturnType Switch_GetSwitchStatus (
            Switch_Name_t switchName,
            Switch_State_t * state )
```

Gets the status of the switch.

4.27.2.2 Function: Switch_GetSwitchStatus

Parameters

switchName	The name of the Switch
state	Save the status of the switch in • SWITCH_PRESSED : if the switch is pressed • SWITCH_NOT_PRESSED : if the switch is not pressed

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.27.2.3 Switch_Init()

```
Std_ReturnType Switch_Init (
            void  )
```

Initializes GPIOs for the Switches.

4.27.2.4 Function: Switch_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.28 Switch.h File Reference

This file is to be used as an interface for the user of the Switch Handler.

```
#include "Switch_Cfg.h"
```

Data Structures

- struct switch_t

Macros

- #define **SWITCH_NOT_PRESSED** 1
- #define **SWITCH_PRESSED** 0

Typedefs

- typedef uint8_t **Switch_Name_t**
- typedef uint8_t **Switch_State_t**

Functions

- Std_ReturnType Switch_Init (void)

  Initializes GPIOs for the Switches.
- Std_ReturnType Switch_GetSwitchStatus (Switch_Name_t switchName, Switch_State_t ∗state)

  Gets the status of the switch.

4.28.1 Detailed Description

This file is to be used as an interface for the user of the Switch Handler.

Author

Mark Attia

Date

January 22, 2020

4.28.2 Function Documentation

4.28.2.1 Switch_GetSwitchStatus()

```
Std_ReturnType Switch_GetSwitchStatus (
            Switch_Name_t switchName,
            Switch_State_t * state )
```

Gets the status of the switch.

4.28.2.2 Function: Switch_GetSwitchStatus

Parameters

switchName	The name of the Switch
state	Save the status of the switch in • SWITCH_PRESSED : if the switch is pressed • SWITCH_NOT_PRESSED : if the switch is not pressed

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.28.2.3 Function: Switch_GetSwitchStatus

Parameters

switchName	The name of the Switch
state	Save the status of the switch in • SWITCH_PRESSED : if the switch is pressed • SWITCH_NOT_PRESSED : if the switch is not pressed

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.28.2.4 Switch_Init()

```
Std_ReturnType Switch_Init (
            void  )
```

Initializes GPIOs for the Switches.

4.28.2.5 Function: Switch_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.28.2.6 Function: Switch_Init

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.29 Switch_Cfg.c File Reference

This file is to be used as an implementation of the configurations the user configured in the Switch_Cfg.h.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Switch.h"
```

Variables

- const switch_t **Switch_switches** [SWITCH_NUMBER_OF_SWITCHES]

4.29.1 Detailed Description

This file is to be used as an implementation of the configurations the user configured in the Switch_Cfg.h.

Author

Mark Attia

Date

January 22, 2020

4.29.2 Variable Documentation

4.29.2.1 Switch_switches

```
const switch_t Switch_switches[SWITCH_NUMBER_OF_SWITCHES]
```

Initial value:

```
= {
    {GPIO_PIN_5, GPIO_PORTB, GPIO_PIN_RESET},
    {GPIO_PIN_4, GPIO_PORTB, GPIO_PIN_RESET},
    {GPIO_PIN_3, GPIO_PORTB, GPIO_PIN_RESET}
}
```

4.30 Switch_Cfg.h File Reference

This file is to be given to the user to configure the Switch Handler.

Macros

- #define **SWITCH_USE_RTOS**
- #define **SWITCH_NUMBER_OF_SWITCHES** 3
- #define **WATER_HEATER_ON_OFF_BUTTON** 0
- #define **WATER_HEATER_DOWN_BUTTON** 1
- #define **WATER_HEATER_UP_BUTTON** 2

4.30.1 Detailed Description

This file is to be given to the user to configure the Switch Handler.

Author

Mark Attia

Date

January 22, 2020

4.31 Timer1.h File Reference

This file is to be used as an implementation for the user of Timer 1 driver.

Macros

- #define **TMR1_DIV_1** 0x00
- #define **TMR1_DIV_2** 0x10
- #define **TMR1_DIV_4** 0x20
- #define **TMR1_DIV_8** 0x30

Typedefs

- typedef uint8_t **Timer1_Prescaler_t**

Functions

- Std_ReturnType Timer1_InterruptEnable (void)
  Enables the interrupt for the Timer1.
- Std_ReturnType Timer1_InterruptDisable (void)
  Disables the interrupt for the Timer1.
- Std_ReturnType Timer1_Start (Timer1_Prescaler_t prescaler)
  Enables the Timer1 timer.
- Std_ReturnType Timer1_SetTimeUS (f64 timerClock, uint32_t timeUS)
  Sets The reload time for timer 0.
- Std_ReturnType Timer1_Stop (void)
  Disables the Timer1 timer.
- Std_ReturnType Timer1_GetValue (uint16_t ∗val)
  Reads the current value inside the Timer1 timer.
- Std_ReturnType Timer1_SetCallBack (interruptCb_t func)
  Sets the callback function for the Timer1.
- Std_ReturnType Timer1_ClearValue (void)
  Clears the value of the counter.

4.31.1 Detailed Description

This file is to be used as an implementation for the user of Timer 1 driver.

This file is to be used as an interface for the user of Timer 1 driver.

Author

Mark Attia

Date

January 22, 2020

4.31.2 Function Documentation

4.31.2.1 Timer1_ClearValue()

```
Std_ReturnType Timer1_ClearValue (
            void  )
```

Clears the value of the counter.

4.31.2.2 Function: Timer1_ClearValue

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.3 Function: Timer1_ClearValue

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.4 Timer1_GetValue()

```
Std_ReturnType Timer1_GetValue (
            uint16_t * val )
```

Reads the current value inside the Timer1 timer.

4.31.2.5 Function: Timer1_GetValue

Parameters

val	a pointer to return data in

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.6 Function: Timer1_GetValue

Parameters

val	a pointer to return data in

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.7 Timer1_InterruptDisable()

```
Std_ReturnType Timer1_InterruptDisable (
            void  )
```

Disables the interrupt for the Timer1.

4.31.2.8 Function: Timer1_InterruptDisable

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.9 Function: Timer1_InterruptDisable

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.10 Timer1_InterruptEnable()

```
Std_ReturnType Timer1_InterruptEnable (
            void  )
```

Enables the interrupt for the Timer1.

4.31.2.11 Function: Timer1_InterruptEnable

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.12 Function: Timer1_InterruptEnable

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.13 Timer1_SetCallBack()

```
Std_ReturnType Timer1_SetCallBack (
            interruptCb_t func )
```

Sets the callback function for the Timer1.

4.31.2.14 Function: Timer1_SetCallBack

Parameters

func	the callback function

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.15 Function: Timer1_SetCallBack

Parameters

func	the callback function

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.16 Timer1_SetTimeUS()

```
Std_ReturnType Timer1_SetTimeUS (
            f64 timerClock,
            uint32_t timeUS )
```

Sets The reload time for timer 0.

4.31.2.17 Function: Timer1_SetTimeUS

Parameters

timerClock	The Timer clock frequency
timeUS	The time in Micro seconds

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

Sets The reload time for timer 0.

4.31.2.18 Function: Timer1_SetTimeUS

Parameters

timerClock	The Timer clock frequency
timeUS	The time in Micro seconds

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.19 Timer1_Start()

```
Std_ReturnType Timer1_Start (
            Timer1_Prescaler_t prescaler )
```

Enables the Timer1 timer.

4.31.2.20 Function: Timer1_Start

Parameters

prescaler	the division value for system clock • TMR0_DIV_1 • TMR0_DIV_8 • TMR0_DIV_64 • TMR0_DIV_256 • TMR0_DIV_1024

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.21 Function: Timer1_Start

Parameters

prescaler	the division value for system clock • TMR0_DIV_1 • TMR0_DIV_2 • TMR0_DIV_4 • TMR0_DIV_8

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.22 Timer1_Stop()

```
Std_ReturnType Timer1_Stop (
            void  )
```

Disables the Timer1 timer.

4.31.2.23 Function: Timer1_Stop

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.31.2.24 Function: Timer1_Stop

Returns

: A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly

4.32 WaterHeater.c File Reference

This is the implementation for the Electric Water Heater Application.

```
#include "Std_Types.h"
#include "Gpio.h"
#include "Switch.h"
#include "Element.h"
#include "Led.h"
#include "SSeg.h"
#include "Adc.h"
#include "Eeprom.h"
#include "Sched.h"
#include "WaterHeater.h"
#include "WaterHeater_Cfg.h"
```

Macros

- #define **WATER_HEATER_NUMBER_OF_READINGS** 10
- #define **WATER_HEATER_TEMP_DATA_ADDRESS** (Eeprom_Address_t)0x0000
- #define **WATER_HEATER_INITIAL_TEMP** 60
- #define **WATER_HEATER_OFF_MODE** 0
- #define **WATER_HEATER_TEMPRATURE_SETTING_MODE** 1
- #define **WATER_HEATER_RUNNING_MODE** 2
- #define **WATER_HEATER_LOWER_LIMIT** 35
- #define **WATER_HEATER_UPPER_LIMIT** 75
- #define **WATER_HEATER_CHANGE_RATE** 5
- #define **WATER_HEATER_HEATING_ELEMENT_RUNNING** 0
- #define **WATER_HEATER_COOLING_ELEMENT_RUNNING** 1
- #define **WATER_HEATER_NO_ELEMENT_RUNNING** 2
- #define **WATER_HEATER_100_MS_MASK** 0x07
- #define **WATER_HEATER_100_MS_MASK_OK** 0
- #define **WATER_HEATER_HALF_SEC_MASK** 20
- #define **WATER_HEATER_5_SEC** 10
- #define **WATER_HEATER_COUNTER_RESET_VALUE** 0
- #define **WATER_HEATER_INDEX_RESET_VALUE** 0
- #define **WATER_HEATER_TEMPRATURE_SENSOR_FACTOR** 2
- #define **WATER_HEATER_GET_ONES**(data) (data%10)
- #define **WATER_HEATER_GET_TENS**(data) ((data/10)%10)
- #define **WATER_HEATER_INIT_TASK_PERIODICITY** 5
- #define **WATER_HEATER_MAIN_TASK_PERIODICITY** 25

Typedefs

- typedef uint8_t **temperature_t**
- typedef uint8_t **heaterMode_t**
- typedef uint8_t **runningElement_t**
- typedef uint8_t **secCounter_t**
- typedef temperature_t **tempratureReadings_t**[WATER_HEATER_NUMBER_OF_READINGS]

Variables

- const task_t **WaterHeater_InitTask** = {WaterHeater_Init, WATER_HEATER_INIT_TASK_PERIODICITY}
- const task_t **WaterHeater_Task** = {WaterHeater_Runnable, WATER_HEATER_MAIN_TASK_PERIODIC↩ ITY}

4.32.1 Detailed Description

This is the implementation for the Electric Water Heater Application.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.33 WaterHeater.h File Reference

This is the user interface for the Electric Water Heater Application.

4.33.1 Detailed Description

This is the user interface for the Electric Water Heater Application.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

4.34 WaterHeater_Cfg.h File Reference

This is the user's configurations for the Electric Water Heater Application.

4.34.1 Detailed Description

This is the user's configurations for the Electric Water Heater Application.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-07-05

Copyright

Index