

Autosar Dimmer

Generated by Doxygen 1.8.17

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 dataBuffer_t Struct Reference	5
3.2 doorContact_t Struct Reference	5
3.3 gpio_t Struct Reference	5
3.4 led_t Struct Reference	6
3.5 PduInfoType Struct Reference	6
3.6 PduType Struct Reference	6
3.7 switch_t Struct Reference	7
3.8 sysTask_t Struct Reference	7
3.9 sysTaskInfo_t Struct Reference	7
3.10 task_t Struct Reference	7
4 File Documentation	9
4.1 Com.c File Reference	9
4.1.1 Detailed Description	10
4.1.2 Function Documentation	10
4.1.2.1 Com_Init()	10
4.1.2.2 Com_ReceiveSignal()	10
4.1.2.3 Com_SendSignal()	11
4.1.2.4 Com_TriggerTransmit()	11
4.1.3 Variable Documentation	12
4.1.3.1 Com_ByteMasks	12
4.2 Com.h File Reference	12
4.2.1 Detailed Description	13
4.2.2 Function Documentation	13
4.2.2.1 Com_Init()	13
4.2.2.2 Com_ReceiveSignal()	13
4.2.2.3 Com_SendSignal()	14
4.2.2.4 Com_TriggerTransmit()	14
4.3 Com_Cfg.c File Reference	15
4.3.1 Detailed Description	15
4.3.2 Variable Documentation	15
4.3.2.1 PduInfo	15
4.4 Com_Cfg.h File Reference	16
4.4.1 Detailed Description	16
4.5 Dimmer.c File Reference	16
4.5.1 Detailed Description	17

4.5.2 Function Documentation	17
4.5.2.1 Dimmer_Init()	17
4.6 Dimmer.h File Reference	17
4.6.1 Detailed Description	18
4.6.2 Function Documentation	18
4.6.2.1 Dimmer_Init()	18
4.7 DoorContact.c File Reference	18
4.7.1 Detailed Description	19
4.7.2 Function Documentation	19
4.7.2.1 DoorContact_Init()	19
4.8 DoorContact.h File Reference	19
4.8.1 Detailed Description	20
4.8.2 Function Documentation	20
4.8.2.1 DoorContact_Init()	20
4.9 Gpio.c File Reference	20
4.9.1 Detailed Description	21
4.9.2 Function Documentation	21
4.9.2.1 Gpio_InitPins()	21
4.9.2.2 Function: Gpio_InitPins	21
4.9.2.3 Gpio_ReadPin()	21
4.9.2.4 Function: Gpio_ReadPin	22
4.9.2.5 Gpio_WritePin()	22
4.9.2.6 Function: Gpio_WritePin	22
4.10 Gpio.h File Reference	23
4.10.1 Detailed Description	23
4.10.2 Function Documentation	24
4.10.2.1 Gpio_InitPins()	24
4.10.2.2 Function: Gpio_InitPins	24
4.10.2.3 Function: Gpio_InitPins	24
4.10.2.4 Gpio_ReadPin()	24
4.10.2.5 Function: Gpio_ReadPin	24
4.10.2.6 Function: Gpio_ReadPin	25
4.10.2.7 Gpio_WritePin()	25
4.10.2.8 Function: Gpio_WritePin	25
4.10.2.9 Function: Gpio_WritePin	26
4.11 Led.c File Reference	26
4.11.1 Detailed Description	27
4.11.2 Function Documentation	27
4.11.2.1 Led_Init()	27
4.11.2.2 Function: Led_Init	27
4.11.2.3 Led_SetLedOff()	27
4.11.2.4 Function: Led_SetLedOff	27

4.11.2.5 Led_SetLedOn()	28
4.11.2.6 Function: Led_SetLedOn	28
4.11.2.7 Led_SetLedStatus()	28
4.11.2.8 Function: Led_SetLedStatus	28
4.12 Led.h File Reference	29
4.12.1 Detailed Description	29
4.12.2 Function Documentation	29
4.12.2.1 Led_Init()	30
4.12.2.2 Function: Led_Init	30
4.12.2.3 Function: Led_Init	30
4.12.2.4 Led_SetLedOff()	30
4.12.2.5 Function: Led_SetLedOff	30
4.12.2.6 Function: Led_SetLedOff	30
4.12.2.7 Led_SetLedOn()	31
4.12.2.8 Function: Led_SetLedOn	31
4.12.2.9 Function: Led_SetLedOn	31
4.12.2.10 Led_SetLedStatus()	31
4.12.2.11 Function: Led_SetLedStatus	31
4.12.2.12 Function: Led_SetLedStatus	32
4.13 Led_Cfg.h File Reference	32
4.13.1 Detailed Description	32
4.14 LeftDoor.c File Reference	32
4.14.1 Detailed Description	33
4.14.2 Function Documentation	33
4.14.2.1 LeftDoor_Init()	33
4.15 LeftDoor.h File Reference	33
4.15.1 Detailed Description	34
4.15.2 Function Documentation	34
4.15.2.1 LeftDoor_Init()	34
4.16 Lighting.c File Reference	34
4.16.1 Detailed Description	35
4.16.2 Function Documentation	35
4.16.2.1 Lighting_Init()	35
4.17 Lighting.h File Reference	35
4.17.1 Detailed Description	36
4.17.2 Function Documentation	36
4.17.2.1 Lighting_Init()	36
4.18 main.c File Reference	36
4.18.1 Detailed Description	37
4.19 RightDoor.c File Reference	37
4.19.1 Detailed Description	37
4.19.2 Function Documentation	38

4.19.2.1 RightDoor_Init()	38
4.20 RightDoor.h File Reference	38
4.20.1 Detailed Description	38
4.20.2 Function Documentation	38
4.20.2.1 RightDoor_Init()	39
4.21 Rte.c File Reference	39
4.21.1 Detailed Description	40
4.21.2 Function Documentation	40
4.21.2.1 Rte_Call_DimmerReceiveData()	40
4.21.2.2 Rte_Call_DoorContactSendData()	41
4.21.2.3 Rte_Call_LeftDoorGetStatus()	41
4.21.2.4 Rte_Call_LightingSetStatus()	41
4.21.2.5 Rte_Call_RightDoorGetStatus()	43
4.21.2.6 Rte_Read_DimmerStatus()	43
4.21.2.7 Rte_Read_DoorContact()	44
4.21.2.8 Rte_Read_LeftDoorStatus()	44
4.21.2.9 Rte_Read_RightDoorStatus()	45
4.21.2.10 Rte_SetRunnableEntity()	45
4.21.2.11 Rte_Write_DimmerStatus()	45
4.21.2.12 Rte_Write_DoorContactStatus()	46
4.21.2.13 Rte_Write_LeftDoorStatus()	46
4.21.2.14 Rte_Write_RightDoorStatus()	47
4.22 Rte.h File Reference	47
4.22.1 Detailed Description	48
4.22.2 Function Documentation	49
4.22.2.1 Rte_Call_DimmerReceiveData()	49
4.22.2.2 Rte_Call_DimmerWriteData()	49
4.22.2.3 Rte_Call_DoorContactSendData()	49
4.22.2.4 Rte_Call_LeftDoorGetStatus()	50
4.22.2.5 Rte_Call_LightingSetStatus()	51
4.22.2.6 Rte_Call_RightDoorGetStatus()	51
4.22.2.7 Rte_Read_DimmerStatus()	52
4.22.2.8 Rte_Read_DoorContact()	52
4.22.2.9 Rte_Read_LeftDoorStatus()	53
4.22.2.10 Rte_Read_RightDoorStatus()	53
4.22.2.11 Rte_SetRunnableEntity()	53
4.22.2.12 Rte_Write_DimmerStatus()	54
4.22.2.13 Rte_Write_DoorContactStatus()	54
4.22.2.14 Rte_Write_LeftDoorStatus()	55
4.22.2.15 Rte_Write_RightDoorStatus()	55
4.23 Sched.c File Reference	55
4.23.1 Detailed Description	56

4.23.2 Function Documentation	57
4.23.2.1 Sched_Init()	57
4.23.2.2 Sched_Sleep()	57
4.23.2.3 Sched_Start()	57
4.23.2.4 Sched_SuspendTask()	58
4.24 Sched.h File Reference	58
4.24.1 Detailed Description	58
4.24.2 Function Documentation	59
4.24.2.1 Sched_Init()	59
4.24.2.2 Sched_Sleep()	59
4.24.2.3 Sched_Start()	59
4.24.2.4 Sched_SuspendTask()	60
4.25 Sched_Cfg.c File Reference	60
4.25.1 Detailed Description	60
4.25.2 Variable Documentation	61
4.25.2.1 Sched_sysTaskInfo	61
4.26 Sched_Cfg.h File Reference	61
4.26.1 Detailed Description	61
4.27 Switch.c File Reference	62
4.27.1 Detailed Description	62
4.27.2 Function Documentation	62
4.27.2.1 Switch_GetSwitchStatus()	62
4.27.2.2 Function: Switch_GetSwitchStatus	62
4.27.2.3 Switch_Init()	63
4.27.2.4 Function: Switch_Init	63
4.28 Switch.h File Reference	63
4.28.1 Detailed Description	64
4.28.2 Function Documentation	64
4.28.2.1 Switch_GetSwitchStatus()	64
4.28.2.2 Function: Switch_GetSwitchStatus	64
4.28.2.3 Function: Switch_GetSwitchStatus	64
4.28.2.4 Switch_Init()	65
4.28.2.5 Function: Switch_Init	65
4.28.2.6 Function: Switch_Init	65
4.29 Switch_Cfg.c File Reference	65
4.29.1 Detailed Description	65
4.29.2 Variable Documentation	66
4.29.2.1 Switch_switches	66
4.30 Switch_Cfg.h File Reference	66
4.30.1 Detailed Description	66
4.31 Timer0.h File Reference	66
4.31.1 Detailed Description	67

4.31.2 Function Documentation	67
4.31.2.1 Timer0_ClearValue()	67
4.31.2.2 Function: Timer0_ClearValue	67
4.31.2.3 Function: Timer0_ClearValue	68
4.31.2.4 Timer0_GetValue()	68
4.31.2.5 Function: Timer0_GetValue	68
4.31.2.6 Function: Timer0_GetValue	68
4.31.2.7 Timer0_InterruptDisable()	68
4.31.2.8 Function: Timer0_InterruptDisable	68
4.31.2.9 Function: Timer0_InterruptDisable	68
4.31.2.10 Timer0_InterruptEnable()	69
4.31.2.11 Function: Timer0_InterruptEnable	69
4.31.2.12 Timer0_SetCallBack()	69
4.31.2.13 Function: Timer0_SetCallBack	69
4.31.2.14 Function: Timer0_SetCallBack	69
4.31.2.15 Timer0_SetTimeUS()	69
4.31.2.16 Function: Timer0_SetTimeUS	69
4.31.2.17 Function: Timer0_SetTimeUS	70
4.31.2.18 Timer0_Start()	70
4.31.2.19 Function: Timer0_Start	70
4.31.2.20 Function: Timer0_Start	70
4.31.2.21 Timer0_Stop()	71
4.31.2.22 Function: Timer0_Stop	71
4.31.2.23 Function: Timer0_Stop	71
4.32 Uart.c File Reference	71
4.32.1 Detailed Description	73
4.32.2 Function Documentation	73
4.32.2.1 __vector_13()	73
4.32.2.2 __vector_15()	73
4.32.2.3 Uart_Init()	73
4.32.2.4 Uart_Receive()	74
4.32.2.5 Uart_Send()	74
4.32.2.6 Uart_SetRxCb()	75
4.32.2.7 Uart_SetTxCb()	75
4.33 Uart.h File Reference	75
4.33.1 Detailed Description	76
4.33.2 Function Documentation	76
4.33.2.1 Uart_Init()	76
4.33.2.2 Uart_Receive()	77
4.33.2.3 Uart_Send()	77
4.33.2.4 Uart_SetRxCb()	78
4.33.2.5 Uart_SetTxCb()	78

4.34 Uart_Cfg.h File Reference	78
4.34.1 Detailed Description	79
Index	81

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

dataBuffer_t	5
doorContact_t	5
gpio_t	5
led_t	6
PduInfoType	6
PduType	6
switch_t	7
sysTask_t	7
sysTaskInfo_t	7
task_t	7

Chapter 2

File Index

2.1 File List

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

Com.c	This is the implementation for the COM	9
Com.h	This is the user interface for the COM	12
Com_Cfg.c	These are the configurations for the COM	15
Com_Cfg.h	This is the configuration header for the COM	16
Dimmer.c	This is the implementation for the dimmer application	16
Dimmer.h	This is the user interface for the Dimmer	17
DoorContact.c	This is the implementation for the Door Contact application	18
DoorContact.h	This is the user interface for the Door Contact	19
Gpio.c	This file is to be used as an implementation of the GPIO driver	20
Gpio.h	This file is to be used as an interface for the user of GPIO driver	23
Led.c	This file is to be used as an implementation for the Led Handler	26
Led.h	This file is to be used as an interface for the user of the Led Handler	29
Led_Cfg.h	This file is to be given to the user to configure the Led Handler	32
LeftDoor.c	This is the implementation for the Left Door	32
LeftDoor.h	This is the user interface for the Left Door	33
Lighting.c	This is the implementation for the Lighting	34
Lighting.h	This is the user interface for the Lighting	35
main.c	This is the main program that will run after the startup code	36

RightDoor.c	This is the implementation for the Right Door	37
RightDoor.h	This is the user interface for the Right Door	38
Rte.c	39
Rte.h	This is the user interface for the RTE	47
Sched.c	This file is the implementation for the Scheduler	55
Sched.h	This file is the user interface for the Scheduler	58
Sched_Cfg.c	This file contains the configurations implementation for the Scheduler	60
Sched_Cfg.h	This file contains the configurations for the Scheduler	61
Std_Types.h	??
Switch.c	This file is to be used as an implementation for the Switch Handler	62
Switch.h	This file is to be used as an interface for the user of the Switch Handler	63
Switch_Cfg.c	This file is to be used as an implementation of the configurations the user configured in the Switch_Cfg.h	65
Switch_Cfg.h	This file is to be given to the user to configure the Switch Handler	66
Timer0.h	This file is to be used as an interface for the user of Timer 0 driver	66
Uart.c	This is the implementation for the UART driver	71
Uart.h	This is the user interface for the UART driver	75
Uart_Cfg.h	These are the user's configurations for the UART driver	78

Chapter 3

Data Structure Documentation

3.1 `dataBuffer_t` Struct Reference

Data Fields

- `uint8_t * ptr`
- `uint32_t pos`
- `uint32_t size`
- `uint8_t state`

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

- [Uart.c](#)

3.2 `doorContact_t` Struct Reference

Data Fields

- `uint8_t id`
- `uint8_t data`

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

- [Rte.c](#)

3.3 `gpio_t` Struct Reference

Data Fields

- `uint8_t pins`
- `uint8_t mode`
- `uint8_t port`

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

- [Gpio.h](#)

3.4 led_t Struct Reference

Data Fields

- uint8_t **pin**
- uint8_t **port**
- uint8_t **activeState**

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

- [Led.h](#)

3.5 PduInfoType Struct Reference

Data Fields

- PduIdType **id**
- uint8_t **direction**
- uint16_t **nSignals**
- SignalIdType **signal** [PDU_MAX_NUMBER_OF_SIGNALS]
- uint8_t **signalStart** [PDU_MAX_NUMBER_OF_SIGNALS]
- uint8_t **signalWidth** [PDU_MAX_NUMBER_OF_SIGNALS]
- PduTriggerType **trig**
- uint16_t **triggerData**

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

- [Com.h](#)

3.6 PduType Struct Reference

Data Fields

- const [PduInfoType](#) * **pduInf**
- uint16_t **remainingTicks**
- uint16_t **periodicTicks**
- uint8_t **data** [COM_PDU_SIZE_IN_BYTES]
- uint8_t **trig**

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

- [Com.c](#)

3.7 switch_t Struct Reference

Data Fields

- uint32_t **pin**
- uint32_t **port**
- uint8_t **activeState**

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

- [Switch.h](#)

3.8 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](#)

3.9 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](#)

3.10 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 Com.c File Reference

This is the implementation for the COM.

```
#include "Std_Types.h"
#include "Com_Cfg.h"
#include "Com.h"
#include "Uart.h"
#include "Sched.h"
```

Data Structures

- struct [PduType](#)

Macros

- #define **COM_PDU_TRIGGERED** 0
- #define **COM_PDU_NOT_TRIGGERED** 1
- #define **COM_TICK_TIME** 5
- #define **COM_BYTE_SIZE** 8

Functions

- Std_ReturnType [Com_Init](#) (void)
Initialises the Com.
- Std_ReturnType [Com_SendSignal](#) (SignalIdType signalId, uint32_t data)
Sends a signal.
- Std_ReturnType [Com_ReceiveSignal](#) (SignalIdType signalId, uint32_t *data)
Receives a signal.
- Std_ReturnType [Com_TriggerTransmit](#) (PduldType pduld)
Triggers The Transmission.

Variables

- const uint8_t **Com_ByteMasks** [COM_BYTE_SIZE]
- const PduInfoType **PduInfo** [COM_NUMBER_OF_PDUS]
- const task_t **Com_task** = {&Com_Runnable, COM_TICK_TIME}

4.1.1 Detailed Description

This is the implementation for the COM.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-19

Copyright

Copyright (c) 2020

4.1.2 Function Documentation

4.1.2.1 Com_Init()

```
Std_ReturnType Com_Init (  
    void )
```

Initialises the Com.

Returns

Std_ReturnType E_OK E_NOT_OK

4.1.2.2 Com_ReceiveSignal()

```
Std_ReturnType Com_ReceiveSignal (  
    SignalIdType signalId,  
    uint32_t * data )
```

Receives a signal.

Parameters

<i>signal↔ Id</i>	The Id of the signal
<i>data</i>	the data to receive

Returns

Std_ReturnType E_OK E_NOT_OK

4.1.2.3 Com_SendSignal()

```
Std_ReturnType Com_SendSignal (
    SignalIdType signalId,
    uint32_t data )
```

Sends a signal.

Parameters

<i>signal↔ Id</i>	The Id of the signal
<i>data</i>	the data to send through the signal

Returns

Std_ReturnType E_OK E_NOT_OK

4.1.2.4 Com_TriggerTransmit()

```
Std_ReturnType Com_TriggerTransmit (
    PduIdType pduId )
```

Triggers The Transmission.

Parameters

<i>pdu↔ Id</i>	The Id of the Pdu
--------------------	-------------------

Returns

Std_ReturnType E_OK E_NOT_OK

4.1.3 Variable Documentation

4.1.3.1 Com_ByteMasks

```
const uint8_t Com_ByteMasks[COM_BYTE_SIZE]
```

Initial value:

```
= {
    0b1,
    0b11,
    0b111,
    0b1111,
    0b11111,
    0b111111,
    0b1111111,
    0b11111111,
    0b11111111,
    0b11111111
}
```

4.2 Com.h File Reference

This is the user interface for the COM.

Data Structures

- struct [PduInfoType](#)

Macros

- `#define PDU_MAX_NUMBER_OF_SIGNALS 6`
- `#define PDU_TRIGGER_NONE 0`
- `#define PDU_TRIGGER_PERIOD 1`
- `#define PDU_TRIGGER_SIGNAL 2`
- `#define PDU_SEND 0`
- `#define PDU_RECEIVE 1`

Typedefs

- `typedef uint16_t SignalIdType`
- `typedef uint16_t PduIdType`
- `typedef uint8_t PduTriggerType`

Functions

- Std_ReturnType [Com_Init](#) (void)
Initialises the Com.
- Std_ReturnType [Com_SendSignal](#) (SignalIdType signalId, uint32_t data)
Sends a signal.
- Std_ReturnType [Com_ReceiveSignal](#) (SignalIdType signalId, uint32_t *data)
Receives a signal.
- Std_ReturnType [Com_TriggerTransmit](#) (PduIdType pduId)
Triggers The Transmission.

4.2.1 Detailed Description

This is the user interface for the COM.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-19

Copyright

Copyright (c) 2020

4.2.2 Function Documentation

4.2.2.1 Com_Init()

```
Std_ReturnType Com_Init (  
    void )
```

Initialises the Com.

Returns

Std_ReturnType E_OK E_NOT_OK

4.2.2.2 Com_ReceiveSignal()

```
Std_ReturnType Com_ReceiveSignal (  
    SignalIdType signalId,  
    uint32_t * data )
```

Receives a signal.

Parameters

<i>signal↔ Id</i>	The Id of the signal
<i>data</i>	the data to receive

Returns

Std_ReturnType E_OK E_NOT_OK

4.2.2.3 Com_SendSignal()

```
Std_ReturnType Com_SendSignal (
    SignalIdType signalId,
    uint32_t data )
```

Sends a signal.

Parameters

<i>signal↔ Id</i>	The Id of the signal
<i>data</i>	the data to send through the signal

Returns

Std_ReturnType E_OK E_NOT_OK

4.2.2.4 Com_TriggerTransmit()

```
Std_ReturnType Com_TriggerTransmit (
    PduIdType pduId )
```

Triggers The Transmission.

Parameters

<i>pdu↔ Id</i>	The Id of the Pdu
--------------------	-------------------

Returns

Std_ReturnType E_OK E_NOT_OK

4.3 Com_Cfg.c File Reference

These are the configurations for the COM.

```
#include "Std_Types.h"
#include "Com.h"
#include "Com_Cfg.h"
```

Variables

- const [PduInfoType](#) **PduInfo** [COM_NUMBER_OF_PDUS]

4.3.1 Detailed Description

These are the configurations for the COM.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-19

Copyright

Copyright (c) 2020

4.3.2 Variable Documentation

4.3.2.1 PduInfo

```
const PduInfoType PduInfo[COM_NUMBER_OF_PDUS]
```

Initial value:

```
= {
    { DOOR_PDU,          PDU_SEND,          1,          {DOOR_STATE_SIGNAL},          {0},
      {1},              PDU_TRIGGER_PERIOD,          5          }
}
```

4.4 Com_Cfg.h File Reference

This is the configuration header for the COM.

Macros

- `#define COM_NUMBER_OF_PDUS 1`
- `#define COM_NUMBER_OF_SIGNALS 1`
- `#define COM_PDU_START 6`
- `#define COM_PDU_WIDTH 2`
- `#define COM_PDU_SIZE_IN_BYTES 1`
- `#define DOOR_PDU 0`
- `#define DOOR_STATE_SIGNAL 0`

4.4.1 Detailed Description

This is the configuration header for the COM.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-19

Copyright

Copyright (c) 2020

4.5 Dimmer.c File Reference

This is the implementation for the dimmer application.

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

Functions

- Std_ReturnType [Dimmer_Init](#) (void)
Initialise the dimmer application.

4.5.1 Detailed Description

This is the implementation for the dimmer application.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.5.2 Function Documentation

4.5.2.1 Dimmer_Init()

```
Std_ReturnType Dimmer_Init (  
    void )
```

Initialise the dimmer application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.6 Dimmer.h File Reference

This is the user interface for the Dimmer.

Functions

- Std_ReturnType [Dimmer_Init](#) (void)
Initialise the dimmer application.

4.6.1 Detailed Description

This is the user interface for the Dimmer.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.6.2 Function Documentation

4.6.2.1 Dimmer_Init()

```
Std_ReturnType Dimmer_Init (
    void )
```

Initialise the dimmer application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.7 DoorContact.c File Reference

This is the implementation for the Door Contact application.

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

Functions

- Std_ReturnType [DoorContact_Init](#) (void)
Initialise the door contact application.

4.7.1 Detailed Description

This is the implementation for the Door Contact application.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.7.2 Function Documentation

4.7.2.1 DoorContact_Init()

```
Std_ReturnType DoorContact_Init (  
    void )
```

Initialise the door contact application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.8 DoorContact.h File Reference

This is the user interface for the Door Contact.

Functions

- Std_ReturnType [DoorContact_Init](#) (void)
Initialise the door contact application.

4.8.1 Detailed Description

This is the user interface for the Door Contact.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.8.2 Function Documentation

4.8.2.1 DoorContact_Init()

```
Std_ReturnType DoorContact_Init (
    void )
```

Initialise the door contact application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.9 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_PIN 0`
- `#define GPIO_DDR 1`
- `#define GPIO_PORT 2`

Functions

- Std_ReturnType [Gpio_InitPins](#) ([gpio_t](#) *gpio)
Initializes pins mode and speed for a specific port.
- Std_ReturnType [Gpio_WritePin](#) (uint8_t port, uint8_t pin, uint8_t pinStatus)
Write a value to a pin(0/1)
- Std_ReturnType [Gpio_ReadPin](#) (uint8_t port, uint8_t pin, uint8_t *state)
Reads a value to a pin(0/1)

4.9.1 Detailed Description

This file is to be used as an implementation of the 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

<i>gpio</i>	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 Gpio_ReadPin()

```
Std_ReturnType Gpio_ReadPin (
    uint8_t port,
```

```
uint8_t pin,
uint8_t * state )
```

Reads a value to a pin(0/1)

4.9.2.4 Function: Gpio_ReadPin

Parameters

<i>port</i>	The port you want to read from <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to read from
<i>pin</i>	The pin you want to read <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
<i>state</i>	To return a status in <ul style="list-style-type: none"> 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

4.9.2.5 Gpio_WritePin()

```
Std_ReturnType Gpio_WritePin (
    uint8_t port,
    uint8_t pin,
    uint8_t pinStatus )
```

Write a value to a pin(0/1)

4.9.2.6 Function: Gpio_WritePin

Parameters

<i>port</i>	The port you want to configure <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to configure
<i>pin</i>	The pin you want to configure <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
<i>pinStatus</i>	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> GPIO_PIN_SET : Sets the pin value to 1
	<ul style="list-style-type: none"> 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 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_PULLUP** 1
- #define **GPIO_MODE_INPUT_FLOAT** 2
- #define **GPIO_PORTA** 0x39
- #define **GPIO_PORTB** 0x36
- #define **GPIO_PORTC** 0x33
- #define **GPIO_PORTD** 0x30

Functions

- Std_ReturnType [Gpio_InitPins](#) ([gpio_t](#) *gpio)
Initializes pins mode and speed for a specific port.
- Std_ReturnType [Gpio_WritePin](#) (uint8_t port, uint8_t pin, uint8_t pinStatus)
Write a value to a pin(0/1)
- Std_ReturnType [Gpio_ReadPin](#) (uint8_t port, uint8_t pin, uint8_t *state)
Reads a value to a pin(0/1)

4.10.1 Detailed Description

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

Author

Mark Attia

Date

February 6, 2020

4.10.2 Function Documentation

4.10.2.1 Gpio_InitPins()

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

Initializes pins mode and speed for a specific port.

4.10.2.2 Function: Gpio_InitPins

Parameters

<i>gpio</i>	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

4.10.2.3 Function: Gpio_InitPins

Parameters

<i>gpio</i>	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

4.10.2.4 Gpio_ReadPin()

```
Std_ReturnType Gpio_ReadPin (
    uint8_t port,
    uint8_t pin,
    uint8_t * state )
```

Reads a value to a pin(0/1)

4.10.2.5 Function: Gpio_ReadPin

Parameters

<i>port</i>	The port you want to read from <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to read from
<i>pin</i>	The pin you want to read <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
<i>state</i>	To return a status in <ul style="list-style-type: none"> 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

4.10.2.6 Function: Gpio_ReadPin

Parameters

<i>port</i>	The port you want to read from <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to read from
<i>pin</i>	The pin you want to read <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\
<i>state</i>	To return a status in <ul style="list-style-type: none"> 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

4.10.2.7 Gpio_WritePin()

```
Std_ReturnType Gpio_WritePin (
    uint8_t port,
    uint8_t pin,
    uint8_t pinStatus )
```

Write a value to a pin(0/1)

4.10.2.8 Function: Gpio_WritePin

Parameters

<i>port</i>	The port you want to configure <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to configure
<i>pin</i>	The pin you want to configure <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
<i>pinStatus</i>	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> 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

4.10.2.9 Function: Gpio_WritePin

Parameters

<i>port</i>	The port you want to configure <ul style="list-style-type: none"> GPIO_PORTX : The pin number you want to configure
<i>pin</i>	The pin you want to configure <ul style="list-style-type: none"> GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\
<i>pinStatus</i>	The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> 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

4.11 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](#) (uint8_t ledName)
Sets the Led on.
- Std_ReturnType [Led_SetLedOff](#) (uint8_t ledName)
Sets the Led off.
- Std_ReturnType [Led_SetLedStatus](#) (uint8_t ledName, uint8_t status)
Sets the Led off.

Variables

- const [led_t](#) [Led_leds](#) [LED_NUMBER_OF_LEDS]

4.11.1 Detailed Description

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

Author

Mark Attia

Date

January 22, 2020

4.11.2 Function Documentation

4.11.2.1 Led_Init()

```
Std_ReturnType Led_Init (  
    void )
```

Initializes GPIOs for the LEDs.

4.11.2.2 Function: Led_Init

Returns

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

4.11.2.3 Led_SetLedOff()

```
Std_ReturnType Led_SetLedOff (  
    uint8_t ledName )
```

Sets the Led off.

4.11.2.4 Function: Led_SetLedOff

Parameters

<i>ledName</i>	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

4.11.2.5 Led_SetLedOn()

```
Std_ReturnType Led_SetLedOn (  
    uint8_t ledName )
```

Sets the Led on.

4.11.2.6 Function: Led_SetLedOn**Parameters**

<i>ledName</i>	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

4.11.2.7 Led_SetLedStatus()

```
Std_ReturnType Led_SetLedStatus (  
    uint8_t ledName,  
    uint8_t status )
```

Sets the Led off.

4.11.2.8 Function: Led_SetLedStatus**Parameters**

<i>ledName</i>	The name of the LED
<i>pinStatus</i>	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none">• 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

4.12 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

Functions

- Std_ReturnType [Led_Init](#) (void)
Initializes GPIOs for the LEDs.
- Std_ReturnType [Led_SetLedOn](#) (uint8_t ledName)
Sets the Led on.
- Std_ReturnType [Led_SetLedOff](#) (uint8_t ledName)
Sets the Led off.
- Std_ReturnType [Led_SetLedStatus](#) (uint8_t ledName, uint8_t status)
Sets the Led off.

4.12.1 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

4.12.2 Function Documentation

4.12.2.1 Led_Init()

```
Std_ReturnType Led_Init (
    void )
```

Initializes GPIOs for the LEDs.

4.12.2.2 Function: Led_Init

Returns

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

4.12.2.3 Function: Led_Init

Returns

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

4.12.2.4 Led_SetLedOff()

```
Std_ReturnType Led_SetLedOff (
    uint8_t ledName )
```

Sets the Led off.

4.12.2.5 Function: Led_SetLedOff

Parameters

<i>ledName</i>	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

4.12.2.6 Function: Led_SetLedOff

Parameters

<i>ledName</i>	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

4.12.2.7 Led_SetLedOn()

```
Std_ReturnType Led_SetLedOn (
    uint8_t ledName )
```

Sets the Led on.

4.12.2.8 Function: Led_SetLedOn

Parameters

<i>ledName</i>	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

4.12.2.9 Function: Led_SetLedOn

Parameters

<i>ledName</i>	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

4.12.2.10 Led_SetLedStatus()

```
Std_ReturnType Led_SetLedStatus (
    uint8_t ledName,
    uint8_t status )
```

Sets the Led off.

4.12.2.11 Function: Led_SetLedStatus

Parameters

<i>ledName</i>	The name of the LED
<i>pinStatus</i>	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none">• 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

4.12.2.12 Function: Led_SetLedStatus**Parameters**

<i>ledName</i>	The name of the LED
<i>pinStatus</i>	The status of the pin (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> • 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

4.13 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 DIMMER_LAMP 0

4.13.1 Detailed Description

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

Author

Mark Attia

Date

January 22, 2020

4.14 LeftDoor.c File Reference

This is the implementation for the Left Door.

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

Functions

- Std_ReturnType [LeftDoor_Init](#) (void)
Initialise the left door application.

4.14.1 Detailed Description

This is the implementation for the Left Door.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.14.2 Function Documentation

4.14.2.1 LeftDoor_Init()

```
Std_ReturnType LeftDoor_Init (  
    void )
```

Initialise the left door application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.15 LeftDoor.h File Reference

This is the user interface for the Left Door.

Functions

- Std_ReturnType [LeftDoor_Init](#) (void)
Initialise the left door application.

4.15.1 Detailed Description

This is the user interface for the Left Door.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.15.2 Function Documentation

4.15.2.1 LeftDoor_Init()

```
Std_ReturnType LeftDoor_Init (  
    void )
```

Initialise the left door application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.16 Lighting.c File Reference

This is the implementation for the Lighting.

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

Functions

- Std_ReturnType [Lighting_Init](#) (void)
Initialise the lighting application.

4.16.1 Detailed Description

This is the implementation for the Lighting.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.16.2 Function Documentation

4.16.2.1 Lighting_Init()

```
Std_ReturnType Lighting_Init (  
    void )
```

Initialise the lighting application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.17 Lighting.h File Reference

This is the user interface for the Lighting.

Functions

- Std_ReturnType [Lighting_Init](#) (void)
Initialise the lighting application.

4.17.1 Detailed Description

This is the user interface for the Lighting.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.17.2 Function Documentation

4.17.2.1 Lighting_Init()

```
Std_ReturnType Lighting_Init (  
    void )
```

Initialise the lighting application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.18 main.c File Reference

This is the main program that will run after the startup code.

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

Functions

- `int main (void)`

4.18.1 Detailed Description

This is the main program that will run after the startup code.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-07

Copyright

Copyright (c) 2020

4.19 RightDoor.c File Reference

This is the implementation for the Right Door.

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

Functions

- `Std_ReturnType RightDoor_Init (void)`
Initialise the right door application.

4.19.1 Detailed Description

This is the implementation for the Right Door.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.19.2 Function Documentation

4.19.2.1 RightDoor_Init()

```
Std_ReturnType RightDoor_Init (  
    void )
```

Initialise the right door application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.20 RightDoor.h File Reference

This is the user interface for the Right Door.

Functions

- Std_ReturnType [RightDoor_Init](#) (void)
Initialise the right door application.

4.20.1 Detailed Description

This is the user interface for the Right Door.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.20.2 Function Documentation

4.20.2.1 RightDoor_Init()

```
Std_ReturnType RightDoor_Init (
    void )
```

Initialise the right door application.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21 Rte.c File Reference

```
#include "Std_Types.h"
#include "Sched.h"
#include "Switch.h"
#include "Led.h"
#include "Uart.h"
#include "Com_Cfg.h"
#include "Com.h"
#include "Rte.h"
```

Data Structures

- struct [doorContact_t](#)

Macros

- `#define RTE_NUMBER_OF_MODULES 5`

Functions

- Std_ReturnType [Rte_Call_LeftDoorGetStatus](#) (uint8_t *status)
Calls the switch to get the hardware door status.
- Std_ReturnType [Rte_Write_LeftDoorStatus](#) (uint8_t status)
Writes the door status to the RTE.
- Std_ReturnType [Rte_Call_RightDoorGetStatus](#) (uint8_t *status)
Calls the switch to get the hardware door status.
- Std_ReturnType [Rte_Write_RightDoorStatus](#) (uint8_t status)
Writes the door status to the RTE.
- Std_ReturnType [Rte_Read_LeftDoorStatus](#) (uint8_t *status)
Reads the door status from the RTE.
- Std_ReturnType [Rte_Read_RightDoorStatus](#) (uint8_t *status)
Reads the door status from the RTE.
- Std_ReturnType [Rte_Write_DimmerStatus](#) (uint8_t status)
Writes the dimmer status to the RTE.
- Std_ReturnType [Rte_Read_DimmerStatus](#) (uint8_t *status)
Reads the dimmer status from the RTE.

- Std_ReturnType [Rte_Call_LightingSetStatus](#) (uint8_t status)
Calls the Led to set the hardware lamp status.
- Std_ReturnType [Rte_Write_DoorContactStatus](#) (uint8_t status)
Writes the door contact status.
- Std_ReturnType [Rte_Call_DoorContactSendData](#) (void)
Sends Data of the door contact.
- Std_ReturnType [Rte_Call_DimmerReceiveData](#) (void)
Receives Data for the dimmer.
- Std_ReturnType [Rte_Read_DoorContact](#) (uint8_t *status)
Receives Data for the dimmer.
- Std_ReturnType [Rte_SetRunnableEntity](#) (runnable_t runnable, uint8_t module)
Sets the runnable for a certain module.

Variables

- const [task_t](#) [Rte_task](#) = {Rte_Runnable, 40}

4.21.1 Detailed Description

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.21.2 Function Documentation

4.21.2.1 [Rte_Call_DimmerReceiveData\(\)](#)

```
Std_ReturnType Rte_Call_DimmerReceiveData (
    void )
```

Receives Data for the dimmer.

Parameters

<i>receiveID</i>	The ID of the data
<i>data</i>	The Data to receive

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.2 Rte_Call_DoorContactSendData()

```
Std_ReturnType Rte_Call_DoorContactSendData (
    void )
```

Sends Data of the door contact.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.3 Rte_Call_LeftDoorGetStatus()

```
Std_ReturnType Rte_Call_LeftDoorGetStatus (
    uint8_t * status )
```

Calls the switch to get the hardware door status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.4 Rte_Call_LightingSetStatus()

```
Std_ReturnType Rte_Call_LightingSetStatus (
    uint8_t status )
```

Calls the Led to set the hardware lamp status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OF If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.5 Rte_Call_RightDoorGetStatus()

```
Std_ReturnType Rte_Call_RightDoorGetStatus (
    uint8_t * status )
```

Calls the switch to get the hardware door status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.6 Rte_Read_DimmerStatus()

```
Std_ReturnType Rte_Read_DimmerStatus (
    uint8_t * status )
```

Reads the dimmer status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OF If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.7 Rte_Read_DoorContact()

```
Std_ReturnType Rte_Read_DoorContact (
    uint8_t * status )
```

Receives Data for the dimmer.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_OPEN If the dimmer is on• DOOR_CLOSED If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.8 Rte_Read_LeftDoorStatus()

```
Std_ReturnType Rte_Read_LeftDoorStatus (
    uint8_t * status )
```

Reads the door status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.9 Rte_Read_RightDoorStatus()

```
Std_ReturnType Rte_Read_RightDoorStatus (
    uint8_t * status )
```

Reads the door status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.10 Rte_SetRunnableEntity()

```
Std_ReturnType Rte_SetRunnableEntity (
    runnable_t runnable,
    uint8_t module )
```

Sets the runnable for a certain module.

Parameters

<i>runnable</i>	the runnable to set
<i>module</i>	the module to set runnable to

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.11 Rte_Write_DimmerStatus()

```
Std_ReturnType Rte_Write_DimmerStatus (
    uint8_t status )
```

Writes the dimmer status to the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OF If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.12 Rte_Write_DoorContactStatus()

```
Std_ReturnType Rte_Write_DoorContactStatus (
    uint8_t status )
```

Writes the door contact status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_OPEN If the dimmer is on• DOOR_CLOSED If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.13 Rte_Write_LeftDoorStatus()

```
Std_ReturnType Rte_Write_LeftDoorStatus (
    uint8_t status )
```

Writes the door status to the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.21.2.14 Rte_Write_RightDoorStatus()

```
Std_ReturnType Rte_Write_RightDoorStatus (
    uint8_t status )
```

Writes the door status to the RTE.

Parameters

<i>status</i>	<p>The status of the door</p> <ul style="list-style-type: none"> • DOOR_CLOSED If the door is closed • DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22 Rte.h File Reference

This is the user interface for the RTE.

Macros

- #define **DIMMER_ON** 0
- #define **DIMMER_OFF** !DIMMER_ON
- #define **DOOR_CLOSED** 0
- #define **DOOR_OPEN** !DOOR_CLOSED
- #define **RTE_CONTACT_ID** 0
- #define **RTE_LEFT_DOOR_RUNNABLE** 0
- #define **RTE_RIGHT_DOOR_RUNNABLE** 1
- #define **RTE_DOOR_CONTACT_RUNNABLE** 2
- #define **RTE_DIMMER_RUNNABLE** 3
- #define **RTE_LIGHTING_RUNNABLE** 4

Typedefs

- typedef void(* **runnable_t**) (void)

Functions

- Std_ReturnType [Rte_Call_LeftDoorGetStatus](#) (uint8_t *status)
Calls the switch to get the hardware door status.
- Std_ReturnType [Rte_Write_LeftDoorStatus](#) (uint8_t status)
Writes the door status to the RTE.
- Std_ReturnType [Rte_Call_RightDoorGetStatus](#) (uint8_t *status)
Calls the switch to get the hardware door status.
- Std_ReturnType [Rte_Write_RightDoorStatus](#) (uint8_t status)
Writes the door status to the RTE.
- Std_ReturnType [Rte_Read_LeftDoorStatus](#) (uint8_t *status)
Reads the door status from the RTE.
- Std_ReturnType [Rte_Read_RightDoorStatus](#) (uint8_t *status)
Reads the door status from the RTE.
- Std_ReturnType [Rte_Write_DimmerStatus](#) (uint8_t status)
Writes the dimmer status to the RTE.
- Std_ReturnType [Rte_Read_DimmerStatus](#) (uint8_t *status)
Reads the dimmer status from the RTE.
- Std_ReturnType [Rte_Call_LightingSetStatus](#) (uint8_t status)
Calls the Led to set the hardware lamp status.
- Std_ReturnType [Rte_Write_DoorContactStatus](#) (uint8_t status)
Writes the door contact status.
- Std_ReturnType [Rte_Call_DoorContactSendData](#) (void)
Sends Data of the door contact.
- Std_ReturnType [Rte_Call_DimmerReceiveData](#) (void)
Receives Data for the dimmer.
- Std_ReturnType [Rte_Call_DimmerWriteData](#) (void)
Receives Data for the dimmer.
- Std_ReturnType [Rte_Read_DoorContact](#) (uint8_t *status)
Receives Data for the dimmer.
- Std_ReturnType [Rte_SetRunnableEntity](#) (runnable_t runnable, uint8_t module)
Sets the runnable for a certain module.

4.22.1 Detailed Description

This is the user interface for the RTE.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-04-06

Copyright

Copyright (c) 2020

4.22.2 Function Documentation

4.22.2.1 Rte_Call_DimmerReceiveData()

```
Std_ReturnType Rte_Call_DimmerReceiveData (  
    void )
```

Receives Data for the dimmer.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

Parameters

<i>receiveID</i>	The ID of the data
<i>data</i>	The Data to receive

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.2 Rte_Call_DimmerWriteData()

```
Std_ReturnType Rte_Call_DimmerWriteData (  
    void )
```

Receives Data for the dimmer.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.3 Rte_Call_DoorContactSendData()

```
Std_ReturnType Rte_Call_DoorContactSendData (  
    void )
```

Sends Data of the door contact.

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.4 Rte_Call_LeftDoorGetStatus()

```
Std_ReturnType Rte_Call_LeftDoorGetStatus (
    uint8_t * status )
```

Calls the switch to get the hardware door status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.5 Rte_Call_LightingSetStatus()

```
Std_ReturnType Rte_Call_LightingSetStatus (
    uint8_t status )
```

Calls the Led to set the hardware lamp status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OFF If the dimmer is off
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.6 Rte_Call_RightDoorGetStatus()

```
Std_ReturnType Rte_Call_RightDoorGetStatus (
    uint8_t * status )
```

Calls the switch to get the hardware door status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.7 Rte_Read_DimmerStatus()

```
Std_ReturnType Rte_Read_DimmerStatus (
    uint8_t * status )
```

Reads the dimmer status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OF If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.8 Rte_Read_DoorContact()

```
Std_ReturnType Rte_Read_DoorContact (
    uint8_t * status )
```

Receives Data for the dimmer.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_OPEN If the dimmer is on• DOOR_CLOSED If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.9 Rte_Read_LeftDoorStatus()

```
Std_ReturnType Rte_Read_LeftDoorStatus (
    uint8_t * status )
```

Reads the door status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.10 Rte_Read_RightDoorStatus()

```
Std_ReturnType Rte_Read_RightDoorStatus (
    uint8_t * status )
```

Reads the door status from the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_CLOSED If the door is closed• DOOR_OPEN If the door is open
---------------	--

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.11 Rte_SetRunnableEntity()

```
Std_ReturnType Rte_SetRunnableEntity (
    runnable_t runnable,
    uint8_t module )
```

Sets the runnable for a certain module.

Parameters

<i>runnable</i>	the runnable to set
<i>module</i>	the module to set runnable to

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.12 Rte_Write_DimmerStatus()

```
Std_ReturnType Rte_Write_DimmerStatus (  
    uint8_t status )
```

Writes the dimmer status to the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DIMMER_ON If the dimmer is on• DIMMER_OF If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.13 Rte_Write_DoorContactStatus()

```
Std_ReturnType Rte_Write_DoorContactStatus (  
    uint8_t status )
```

Writes the door contact status.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none">• DOOR_OPEN If the dimmer is on• DOOR_CLOSED If the dimmer is off
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.14 Rte_Write_LeftDoorStatus()

```
Std_ReturnType Rte_Write_LeftDoorStatus (
    uint8_t status )
```

Writes the door status to the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none"> • DOOR_CLOSED If the door is closed • DOOR_OPEN If the door is open
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.22.2.15 Rte_Write_RightDoorStatus()

```
Std_ReturnType Rte_Write_RightDoorStatus (
    uint8_t status )
```

Writes the door status to the RTE.

Parameters

<i>status</i>	The status of the door <ul style="list-style-type: none"> • DOOR_CLOSED If the door is closed • DOOR_OPEN If the door is open
---------------	---

Returns

Std_ReturnType E_OK If the function executed successfully E_NOT_OK If the function executed successfully

4.23 Sched.c File Reference

This file is the implementation for the Scheduler.

```
#include "Std_Types.h"
#include "Sched_Cfg.h"
#include "Sched.h"
#include "Timer0.h"
```

Data Structures

- struct [sysTask_t](#)

Macros

- #define **SCHED_TASK_RUNNING** 1
- #define **SCHED_TASK_SUSPENDED** 2

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]

4.23.1 Detailed Description

This file is the implementation for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

Copyright (c) 2020

4.23.2 Function Documentation

4.23.2.1 Sched_Init()

```
Std_ReturnType Sched_Init (  
    void )
```

The initialization for the Scheduler.

Returns

Std_ReturnType

4.23.2.2 Sched_Sleep()

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

Makes a task sleep for a while.

Parameters

<i>timeMS</i>	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

4.23.2.3 Sched_Start()

```
void Sched_Start (  
    void )
```

The scheduler that will run all the time.

4.23.2.4 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

4.24 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.

4.24.1 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

Copyright (c) 2020

4.24.2 Function Documentation

4.24.2.1 Sched_Init()

```
Std_ReturnType Sched_Init (  
    void )
```

The initialization for the Scheduler.

Returns

Std_ReturnType

4.24.2.2 Sched_Sleep()

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

Makes a task sleep for a while.

Parameters

<i>timeMS</i>	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

4.24.2.3 Sched_Start()

```
void Sched_Start (  
    void )
```

The scheduler that will run all the time.

4.24.2.4 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

4.25 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](#) Applnit_task
- const [task_t](#) Switch_task
- const [task_t](#) Rte_task
- const [task_t](#) Com_task
- const [sysTaskInfo_t](#) Sched_sysTaskInfo [SCHED_NUMBER_OF_TASKS]

4.25.1 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

4.25.2 Variable Documentation

4.25.2.1 Sched_sysTaskInfo

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

Initial value:

```
=  
{  
    {&AppInit_task,          0      },  
    {&Com_task,              20     },  
    {&Switch_task,          20     },  
    {&Rte_task,              20     }  
}
```

4.26 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** 8000000

4.26.1 Detailed Description

This file contains the configurations for the Scheduler.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-08

Copyright

Copyright (c) 2020

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](#) (uint8_t switchName, uint8_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 (
    uint8_t switchName,
    uint8_t * state )
```

Gets the status of the switch.

4.27.2.2 Function: Switch_GetSwitchStatus

Parameters

<i>switchName</i>	The name of the Switch
<i>state</i>	Save the status of the switch in <ul style="list-style-type: none"> • 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

Functions

- Std_ReturnType [Switch_Init](#) (void)
Initializes GPIOs for the Switches.
- Std_ReturnType [Switch_GetSwitchStatus](#) (uint8_t switchName, uint8_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 (
    uint8_t switchName,
    uint8_t * state )
```

Gets the status of the switch.

4.28.2.2 Function: Switch_GetSwitchStatus

Parameters

<i>switchName</i>	The name of the Switch
<i>state</i>	Save the status of the switch in <ul style="list-style-type: none">• 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

<i>switchName</i>	The name of the Switch
<i>state</i>	Save the status of the switch in <ul style="list-style-type: none">• 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_1, GPIO_PORTA, GPIO_PIN_RESET},  
    {GPIO_PIN_2, GPIO_PORTA, 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** 2
- #define **LEFT_DOOR** 0
- #define **RIGHT_DOOR** 1

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 Timer0.h File Reference

This file is to be used as an interface for the user of Timer 0 driver.

Macros

- #define **TMR0_PRESCALER_CLR** 0xF8
- #define **TMR0_DIV_1** 0x01
- #define **TMR0_DIV_8** 0x02
- #define **TMR0_DIV_64** 0x03
- #define **TMR0_DIV_256** 0x04
- #define **TMR0_DIV_1024** 0x05

Functions

- Std_ReturnType [Timer0_InterruptEnable](#) (void)
Enables the interrupt for the Timer0.
- Std_ReturnType [Timer0_InterruptDisable](#) (void)
Disables the interrupt for the Timer0.
- Std_ReturnType [Timer0_Start](#) (uint8_t prescaler)
Enables the Timer0 timer.
- Std_ReturnType [Timer0_SetTimeUS](#) (f64 timerClock, uint32_t timeUS)
Sets The reload time for timer 0.
- Std_ReturnType [Timer0_Stop](#) (void)
Disables the Timer0 timer.
- Std_ReturnType [Timer0_GetValue](#) (uint32_t *val)
Reads the current value inside the Timer0 timer.
- Std_ReturnType [Timer0_SetCallBack](#) (callback_t func)
Sets the callback function for the Timer0.
- Std_ReturnType [Timer0_ClearValue](#) (void)
Clears the value of the counter.

4.31.1 Detailed Description

This file is to be used as an interface for the user of Timer 0 driver.

This file is to be used as an implementation for the user of Timer 0 driver.

Author

Mark Attia

Date

January 22, 2020

4.31.2 Function Documentation

4.31.2.1 Timer0_ClearValue()

```
Std_ReturnType Timer0_ClearValue (  
    void )
```

Clears the value of the counter.

4.31.2.2 Function: Timer0_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: Timer0_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 Timer0_GetValue()

```
Std_ReturnType Timer0_GetValue (
    uint32_t * val )
```

Reads the current value inside the Timer0 timer.

4.31.2.5 Function: Timer0_GetValue

Parameters

<i>val</i>	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: Timer0_GetValue

Parameters

<i>val</i>	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 Timer0_InterruptDisable()

```
Std_ReturnType Timer0_InterruptDisable (
    void )
```

Disables the interrupt for the Timer0.

4.31.2.8 Function: Timer0_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: Timer0_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 Timer0_InterruptEnable()

```
Std_ReturnType Timer0_InterruptEnable (
    void )
```

Enables the interrupt for the Timer0.

4.31.2.11 Function: Timer0_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 Timer0_SetCallBack()

```
Std_ReturnType Timer0_SetCallBack (
    callback_t func )
```

Sets the callback function for the Timer0.

4.31.2.13 Function: Timer0_SetCallBack

Parameters

<i>func</i>	the callback function
-------------	-----------------------

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

4.31.2.14 Function: Timer0_SetCallBack

Parameters

<i>func</i>	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 Timer0_SetTimeUS()

```
Std_ReturnType Timer0_SetTimeUS (
    f64 timerClock,
    uint32_t timeUS )
```

Sets The reload time for timer 0.

4.31.2.16 Function: Timer0_SetTimeUS

Parameters

<i>timerClock</i>	The Timer clock frequency
<i>timeUS</i>	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

4.31.2.17 Function: Timer0_SetTimeUS**Parameters**

<i>timerClock</i>	The Timer clock frequency
<i>timeUS</i>	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

4.31.2.18 Timer0_Start()

```
Std_ReturnType Timer0_Start (
    uint8_t prescaler )
```

Enables the Timer0 timer.

4.31.2.19 Function: Timer0_Start**Parameters**

<i>prescaler</i>	the division value for system clock <ul style="list-style-type: none"> • 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
------------------	--

4.31.2.20 Function: Timer0_Start

Parameters

<i>prescaler</i>	<p>the division value for system clock</p> <ul style="list-style-type: none"> • 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 Timer0_Stop()

```
Std_ReturnType Timer0_Stop (
    void )
```

Disables the Timer0 timer.

4.31.2.22 Function: Timer0_Stop

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

4.31.2.23 Function: Timer0_Stop

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

4.32 Uart.c File Reference

This is the implementation for the UART driver.

```
#include "Std_Types.h"
#include "Uart_Cfg.h"
#include "Uart.h"
```

Data Structures

- struct [dataBuffer_t](#)

Macros

- `#define UDR (*(volatile uint8_t*)(0x2C))`
- `#define UBRRH (*(volatile uint8_t*)(0x40))`
- `#define UCSRC (*(volatile uint8_t*)(0x40))`
- `#define UCSRA (*(volatile uint8_t*)(0x2B))`
- `#define UCSRB (*(volatile uint8_t*)(0x2A))`
- `#define UBRRL (*(volatile uint8_t*)(0x29))`
- `#define SREG *((volatile uint8_t*)0x5F)`
- `#define GIE 0x80`
- `#define UART_INT_NUMBER 37`
- `#define UART_BUFFER_IDLE 0`
- `#define UART_BUFFER_BUSY 1`
- `#define UART_RXCIE_SET 0x80`
- `#define UART_TXCIE_SET 0x40`
- `#define UART_UDRIE_CLR 0xDF`
- `#define UART_RX_EN 0x10`
- `#define UART_TX_EN 0x08`
- `#define UART_CLR_PARITY 0xCF`
- `#define UART_NO_PARITY 0x00`
- `#define UART_BYTE 0x06`
- `#define UART_UCSRC_SELECT 0x80`
- `#define UART_NO_PRESCALER 0x1`

Typedefs

- `typedef void(* appNotify_t) (void)`

Functions

- void [__vector_13](#) (void)
The interrupt handler for the UART 1 module Receive Interrupt.
- void [__vector_15](#) (void)
The interrupt handler for the UART 1 module Transmission Complete.
- Std_ReturnType [Uart_Init](#) (uint32_t baudRate, uint32_t stopBits, uint32_t parity)
Initializes the UART.
- Std_ReturnType [Uart_Send](#) (uint8_t *data, uint16_t length)
Sends data through the UART.
- Std_ReturnType [Uart_Receive](#) (uint8_t *data, uint16_t length)
Receives data through the UART.
- Std_ReturnType [Uart_SetTxCb](#) (txCb_t func)
Sets the callback function that will be called when transmission is completed.
- Std_ReturnType [Uart_SetRxCb](#) (rxCb_t func)
Sets the callback function that will be called when receive is completed.

4.32.1 Detailed Description

This is the implementation for the UART driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-26

Copyright

Copyright (c) 2020

4.32.2 Function Documentation

4.32.2.1 `__vector_13()`

```
void __vector_13 (
    void )
```

The interrupt handler for the UART 1 module Receive Interrupt.

4.32.2.2 `__vector_15()`

```
void __vector_15 (
    void )
```

The interrupt handler for the UART 1 module Transmission Complete.

4.32.2.3 `Uart_Init()`

```
Std_ReturnType Uart_Init (
    uint32_t baudRate,
    uint32_t stopBits,
    uint32_t parity )
```

Initializes the UART.

Parameters

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits UART_ONE_STOP_BIT UART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission UART_ODD_PARITY UART_EVEN_PARITY UART_NO_PARITY

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.32.2.4 Uart_Receive()

```
Std_ReturnType Uart_Receive (
    uint8_t * data,
    uint16_t length )
```

Receives data through the UART.

Parameters

<i>data</i>	The buffer to receive data in
<i>length</i>	the length of the data in bytes

Returns

Std_ReturnType A Status E_OK: If the driver is ready to receive E_NOT_OK: If the driver can't receive data right now

4.32.2.5 Uart_Send()

```
Std_ReturnType Uart_Send (
    uint8_t * data,
    uint16_t length )
```

Sends data through the UART.

Parameters

<i>data</i>	The data to send
<i>length</i>	the length of the data in bytes

Returns

Std_ReturnType A Status E_OK: If the driver is ready to send E_NOT_OK: If the driver can't send data right now

4.32.2.6 Uart_SetRxCb()

```
Std_ReturnType Uart_SetRxCb (
    rxCb_t func )
```

Sets the callback function that will be called when receive is completed.

Parameters

<i>func</i>	the callback function
-------------	-----------------------

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.32.2.7 Uart_SetTxCb()

```
Std_ReturnType Uart_SetTxCb (
    txCb_t func )
```

Sets the callback function that will be called when transmission is completed.

Parameters

<i>func</i>	the callback function
-------------	-----------------------

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.33 Uart.h File Reference

This is the user interface for the UART driver.

Macros

- `#define UART_NO_PARITY 0x00`
- `#define UART_EVEN_PARITY 0x20`
- `#define UART_ODD_PARITY 0x30`
- `#define UART_STOP_BIT_CLR 0xF7`
- `#define UART_ONE_STOP_BIT 0x00`
- `#define UART_TWO_STOP_BITS 0x08`

Typedefs

- `typedef void(* txCb_t) (void)`
- `typedef void(* rxCb_t) (void)`

Functions

- Std_ReturnType [Uart_Init](#) (uint32_t baudRate, uint32_t stopBits, uint32_t parity)
Initializes the UART.
- Std_ReturnType [Uart_Send](#) (uint8_t *data, uint16_t length)
Sends data through the UART.
- Std_ReturnType [Uart_Receive](#) (uint8_t *data, uint16_t length)
Receives data through the UART.
- Std_ReturnType [Uart_SetTxCb](#) (txCb_t func)
Sets the callback function that will be called when transmission is completed.
- Std_ReturnType [Uart_SetRxCb](#) (rxCb_t func)
Sets the callback function that will be called when receive is completed.

4.33.1 Detailed Description

This is the user interface for the UART driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-26

Copyright

Copyright (c) 2020

4.33.2 Function Documentation

4.33.2.1 Uart_Init()

```
Std_ReturnType Uart_Init (
    uint32_t baudRate,
    uint32_t stopBits,
    uint32_t parity )
```

Initializes the UART.

Parameters

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits UART_ONE_STOP_BIT UART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission UART_ODD_PARITY UART_EVEN_PARITY UART_NO_PARITY

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.33.2.2 Uart_Receive()

```
Std_ReturnType Uart_Receive (
    uint8_t * data,
    uint16_t length )
```

Receives data through the UART.

Parameters

<i>data</i>	The buffer to receive data in
<i>length</i>	the length of the data in bytes

Returns

Std_ReturnType A Status E_OK: If the driver is ready to receive E_NOT_OK: If the driver can't receive data right now

4.33.2.3 Uart_Send()

```
Std_ReturnType Uart_Send (
    uint8_t * data,
    uint16_t length )
```

Sends data through the UART.

Parameters

<i>data</i>	The data to send
<i>length</i>	the length of the data in bytes

Returns

Std_ReturnType A Status E_OK: If the driver is ready to send E_NOT_OK: If the driver can't send data right now

4.33.2.4 Uart_SetRxCb()

```
Std_ReturnType Uart_SetRxCb (
    rxCb_t func )
```

Sets the callback function that will be called when receive is completed.

Parameters

<i>func</i>	the callback function
-------------	-----------------------

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.33.2.5 Uart_SetTxCb()

```
Std_ReturnType Uart_SetTxCb (
    txCb_t func )
```

Sets the callback function that will be called when transmission is completed.

Parameters

<i>func</i>	the callback function
-------------	-----------------------

Returns

Std_ReturnType A Status E_OK: If the function executed successfully E_NOT_OK: If the did not execute successfully

4.34 Uart_Cfg.h File Reference

These are the user's configurations for the UART driver.

Macros

- `#define UART_SYSTEM_CLK 8000000`

4.34.1 Detailed Description

These are the user's configurations for the UART driver.

Author

Mark Attia (markjosephattia@gmail.com)

Version

0.1

Date

2020-03-27

Copyright

Copyright (c) 2020

Index

- [__vector_13](#)
 - [Uart.c, 73](#)
 - [__vector_15](#)
 - [Uart.c, 73](#)
- [Com.c, 9](#)
 - [Com_ByteMasks, 12](#)
 - [Com_Init, 10](#)
 - [Com_ReceiveSignal, 10](#)
 - [Com_SendSignal, 11](#)
 - [Com_TriggerTransmit, 11](#)
- [Com.h, 12](#)
 - [Com_Init, 13](#)
 - [Com_ReceiveSignal, 13](#)
 - [Com_SendSignal, 14](#)
 - [Com_TriggerTransmit, 14](#)
- [Com_ByteMasks](#)
 - [Com.c, 12](#)
- [Com_Cfg.c, 15](#)
 - [PdulInfo, 15](#)
- [Com_Cfg.h, 16](#)
- [Com_Init](#)
 - [Com.c, 10](#)
 - [Com.h, 13](#)
- [Com_ReceiveSignal](#)
 - [Com.c, 10](#)
 - [Com.h, 13](#)
- [Com_SendSignal](#)
 - [Com.c, 11](#)
 - [Com.h, 14](#)
- [Com_TriggerTransmit](#)
 - [Com.c, 11](#)
 - [Com.h, 14](#)
- [dataBuffer_t, 5](#)
- [Dimmer.c, 16](#)
 - [Dimmer_Init, 17](#)
- [Dimmer.h, 17](#)
 - [Dimmer_Init, 18](#)
- [Dimmer_Init](#)
 - [Dimmer.c, 17](#)
 - [Dimmer.h, 18](#)
- [DoorContact.c, 18](#)
 - [DoorContact_Init, 19](#)
- [DoorContact.h, 19](#)
 - [DoorContact_Init, 20](#)
- [DoorContact_Init](#)
 - [DoorContact.c, 19](#)
 - [DoorContact.h, 20](#)
- [doorContact_t, 5](#)
- [Gpio.c, 20](#)
 - [Gpio_InitPins, 21](#)
 - [Gpio_ReadPin, 21](#)
 - [Gpio_WritePin, 22](#)
- [Gpio.h, 23](#)
 - [Gpio_InitPins, 24](#)
 - [Gpio_ReadPin, 24](#)
 - [Gpio_WritePin, 25](#)
- [Gpio_InitPins](#)
 - [Gpio.c, 21](#)
 - [Gpio.h, 24](#)
- [Gpio_ReadPin](#)
 - [Gpio.c, 21](#)
 - [Gpio.h, 24](#)
- [gpio_t, 5](#)
- [Gpio_WritePin](#)
 - [Gpio.c, 22](#)
 - [Gpio.h, 25](#)
- [Led.c, 26](#)
 - [Led_Init, 27](#)
 - [Led_SetLedOff, 27](#)
 - [Led_SetLedOn, 28](#)
 - [Led_SetLedStatus, 28](#)
- [Led.h, 29](#)
 - [Led_Init, 29](#)
 - [Led_SetLedOff, 30](#)
 - [Led_SetLedOn, 30](#)
 - [Led_SetLedStatus, 31](#)
- [Led_Cfg.h, 32](#)
- [Led_Init](#)
 - [Led.c, 27](#)
 - [Led.h, 29](#)
- [Led_SetLedOff](#)
 - [Led.c, 27](#)
 - [Led.h, 30](#)
- [Led_SetLedOn](#)
 - [Led.c, 28](#)
 - [Led.h, 30](#)
- [Led_SetLedStatus](#)
 - [Led.c, 28](#)
 - [Led.h, 31](#)
- [led_t, 6](#)
- [LeftDoor.c, 32](#)
 - [LeftDoor_Init, 33](#)
- [LeftDoor.h, 33](#)
 - [LeftDoor_Init, 34](#)
- [LeftDoor_Init](#)
 - [LeftDoor.c, 33](#)
 - [LeftDoor.h, 34](#)

Lighting.c, [34](#)
 Lighting_Init, [35](#)
 Lighting.h, [35](#)
 Lighting_Init, [36](#)
 Lighting_Init
 Lighting.c, [35](#)
 Lighting.h, [36](#)

 main.c, [36](#)

 PduInfo
 Com_Cfg.c, [15](#)
 PduInfoType, [6](#)
 PduType, [6](#)

 RightDoor.c, [37](#)
 RightDoor_Init, [38](#)
 RightDoor.h, [38](#)
 RightDoor_Init, [38](#)
 RightDoor_Init
 RightDoor.c, [38](#)
 RightDoor.h, [38](#)
 Rte.c, [39](#)
 Rte_Call_DimmerReceiveData, [40](#)
 Rte_Call_DoorContactSendData, [41](#)
 Rte_Call_LeftDoorGetStatus, [41](#)
 Rte_Call_LightingSetStatus, [41](#)
 Rte_Call_RightDoorGetStatus, [43](#)
 Rte_Read_DimmerStatus, [43](#)
 Rte_Read_DoorContact, [44](#)
 Rte_Read_LeftDoorStatus, [44](#)
 Rte_Read_RightDoorStatus, [44](#)
 Rte_SetRunnableEntity, [45](#)
 Rte_Write_DimmerStatus, [45](#)
 Rte_Write_DoorContactStatus, [46](#)
 Rte_Write_LeftDoorStatus, [46](#)
 Rte_Write_RightDoorStatus, [47](#)
 Rte.h, [47](#)
 Rte_Call_DimmerReceiveData, [49](#)
 Rte_Call_DimmerWriteData, [49](#)
 Rte_Call_DoorContactSendData, [49](#)
 Rte_Call_LeftDoorGetStatus, [49](#)
 Rte_Call_LightingSetStatus, [51](#)
 Rte_Call_RightDoorGetStatus, [51](#)
 Rte_Read_DimmerStatus, [52](#)
 Rte_Read_DoorContact, [52](#)
 Rte_Read_LeftDoorStatus, [52](#)
 Rte_Read_RightDoorStatus, [53](#)
 Rte_SetRunnableEntity, [53](#)
 Rte_Write_DimmerStatus, [54](#)
 Rte_Write_DoorContactStatus, [54](#)
 Rte_Write_LeftDoorStatus, [55](#)
 Rte_Write_RightDoorStatus, [55](#)
 Rte_Call_DimmerReceiveData
 Rte.c, [40](#)
 Rte.h, [49](#)
 Rte_Call_DimmerWriteData
 Rte.h, [49](#)
 Rte_Call_DoorContactSendData
 Rte.c, [41](#)
 Rte.h, [49](#)
 Rte_Call_LeftDoorGetStatus
 Rte.c, [41](#)
 Rte.h, [49](#)
 Rte_Call_LightingSetStatus
 Rte.c, [41](#)
 Rte.h, [51](#)
 Rte_Call_RightDoorGetStatus
 Rte.c, [43](#)
 Rte.h, [51](#)
 Rte_Read_DimmerStatus
 Rte.c, [43](#)
 Rte.h, [52](#)
 Rte_Read_DoorContact
 Rte.c, [44](#)
 Rte.h, [52](#)
 Rte_Read_LeftDoorStatus
 Rte.c, [44](#)
 Rte.h, [52](#)
 Rte_Read_RightDoorStatus
 Rte.c, [44](#)
 Rte.h, [53](#)
 Rte_SetRunnableEntity
 Rte.c, [45](#)
 Rte.h, [53](#)
 Rte_Write_DimmerStatus
 Rte.c, [45](#)
 Rte.h, [54](#)
 Rte_Write_DoorContactStatus
 Rte.c, [46](#)
 Rte.h, [54](#)
 Rte_Write_LeftDoorStatus
 Rte.c, [46](#)
 Rte.h, [55](#)
 Rte_Write_RightDoorStatus
 Rte.c, [47](#)
 Rte.h, [55](#)

 Sched.c, [55](#)
 Sched_Init, [57](#)
 Sched_Sleep, [57](#)
 Sched_Start, [57](#)
 Sched_SuspendTask, [57](#)
 Sched.h, [58](#)
 Sched_Init, [59](#)
 Sched_Sleep, [59](#)
 Sched_Start, [59](#)
 Sched_SuspendTask, [59](#)
 Sched_Cfg.c, [60](#)
 Sched_sysTaskInfo, [61](#)
 Sched_Cfg.h, [61](#)
 Sched_Init
 Sched.c, [57](#)
 Sched.h, [59](#)
 Sched_Sleep
 Sched.c, [57](#)
 Sched.h, [59](#)
 Sched_Start

- Sched.c, [57](#)
- Sched.h, [59](#)
- Sched_SuspendTask
 - Sched.c, [57](#)
 - Sched.h, [59](#)
- Sched_sysTaskInfo
 - Sched_Cfg.c, [61](#)
- Switch.c, [62](#)
 - Switch_GetSwitchStatus, [62](#)
 - Switch_Init, [63](#)
- Switch.h, [63](#)
 - Switch_GetSwitchStatus, [64](#)
 - Switch_Init, [65](#)
- Switch_Cfg.c, [65](#)
 - Switch_switches, [66](#)
- Switch_Cfg.h, [66](#)
- Switch_GetSwitchStatus
 - Switch.c, [62](#)
 - Switch.h, [64](#)
- Switch_Init
 - Switch.c, [63](#)
 - Switch.h, [65](#)
- Switch_switches
 - Switch_Cfg.c, [66](#)
- switch_t, [7](#)
- sysTask_t, [7](#)
- sysTaskInfo_t, [7](#)

- task_t, [7](#)
- Timer0.h, [66](#)
 - Timer0_ClearValue, [67](#)
 - Timer0_GetValue, [68](#)
 - Timer0_InterruptDisable, [68](#)
 - Timer0_InterruptEnable, [68](#)
 - Timer0_SetCallBack, [69](#)
 - Timer0_SetTimeUS, [69](#)
 - Timer0_Start, [70](#)
 - Timer0_Stop, [71](#)
- Timer0_ClearValue
 - Timer0.h, [67](#)
- Timer0_GetValue
 - Timer0.h, [68](#)
- Timer0_InterruptDisable
 - Timer0.h, [68](#)
- Timer0_InterruptEnable
 - Timer0.h, [68](#)
- Timer0_SetCallBack
 - Timer0.h, [69](#)
- Timer0_SetTimeUS
 - Timer0.h, [69](#)
- Timer0_Start
 - Timer0.h, [70](#)
- Timer0_Stop
 - Timer0.h, [71](#)

- Uart.c, [71](#)
 - __vector_13, [73](#)
 - __vector_15, [73](#)
 - Uart_Init, [73](#)
 - Uart_Receive, [74](#)
 - Uart_Send, [74](#)
 - Uart_SetRxCb, [75](#)
 - Uart_SetTxCb, [75](#)
- Uart.h, [75](#)
 - Uart_Init, [76](#)
 - Uart_Receive, [77](#)
 - Uart_Send, [77](#)
 - Uart_SetRxCb, [78](#)
 - Uart_SetTxCb, [78](#)
- Uart_Cfg.h, [78](#)
- Uart_Init
 - Uart.c, [73](#)
 - Uart.h, [76](#)
- Uart_Receive
 - Uart.c, [74](#)
 - Uart.h, [77](#)
- Uart_Send
 - Uart.c, [74](#)
 - Uart.h, [77](#)
- Uart_SetRxCb
 - Uart.c, [75](#)
 - Uart.h, [78](#)
- Uart_SetTxCb
 - Uart.c, [75](#)
 - Uart.h, [78](#)