

Bootloader

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
3.1.1 Detailed Description	5
3.2 Fpec_t Struct Reference	5
3.3 gpio_t Struct Reference	6
3.3.1 Detailed Description	6
3.4 gpioReg_t Struct Reference	6
3.5 header_t Struct Reference	6
3.6 nvic_t Struct Reference	7
3.6.1 Detailed Description	7
3.7 packet_t Struct Reference	7
3.8 switch_t Struct Reference	7
3.8.1 Detailed Description	8
3.9 Uart_cfg_t Struct Reference	8
3.10 uart_t Struct Reference	8
3.10.1 Detailed Description	8
4 File Documentation	9
4.1 Bootloader.c File Reference	9
4.1.1 Detailed Description	10
4.2 Fpec.h File Reference	10
4.2.1 Detailed Description	10
4.2.2 Function Documentation	11
4.2.2.1 Fpec_ErasePage()	11
4.2.2.2 Fpec_Lock()	11
4.2.2.3 Fpec_MassErase()	11
4.2.2.4 Fpec_Unlock()	12
4.2.2.5 Fpec_WriteBlock()	12
4.2.2.6 Fpec_WriteHalfWord()	12
4.3 Gpio.c File Reference	13
4.3.1 Detailed Description	13
4.3.2 Function Documentation	13
4.3.2.1 Gpio_InitPins()	13
4.3.2.2 Function: Gpio_InitPins	13
4.3.2.3 Gpio_ReadPin()	14
4.3.2.4 Function: Gpio_ReadPin	14
4.3.2.5 Gpio_WritePin()	14

4.3.2.6 Function: Gpio_WritePin	14
4.4 Gpio.h File Reference	15
4.4.1 Detailed Description	16
4.4.2 Function Documentation	16
4.4.2.1 Gpio_InitPins()	16
4.4.2.2 Function: Gpio_InitPins	16
4.4.2.3 Function: Gpio_InitPins	17
4.4.2.4 Gpio_ReadPin()	17
4.4.2.5 Function: Gpio_ReadPin	17
4.4.2.6 Function: Gpio_ReadPin	18
4.4.2.7 Gpio_WritePin()	18
4.4.2.8 Function: Gpio_WritePin	18
4.4.2.9 Function: Gpio_WritePin	19
4.5 HRCc.h File Reference	19
4.5.1 Detailed Description	19
4.5.2 Function Documentation	20
4.5.2.1 HRCc_EnPortClock()	20
4.5.2.2 HRCc_SystemClockInit()	20
4.6 Nvic.c File Reference	20
4.6.1 Detailed Description	22
4.6.2 Function Documentation	22
4.6.2.1 Nvic_ClearFault()	22
4.6.2.2 Nvic_ClearPending()	22
4.6.2.3 Nvic_ConfigGroupSize()	23
4.6.2.4 Nvic_DisableInterrupt()	23
4.6.2.5 Nvic_DisablePeripheral()	23
4.6.2.6 Nvic_EnableInterrupt()	24
4.6.2.7 Nvic_EnablePeripheral()	24
4.6.2.8 Nvic_FilterPriority()	24
4.6.2.9 Nvic_GenerateSoftwareInterrupt()	25
4.6.2.10 Nvic_GetGroupPriority()	25
4.6.2.11 Nvic_GetSubpriority()	25
4.6.2.12 Nvic_IsInterruptActive()	26
4.6.2.13 Nvic_ResetSystem()	26
4.6.2.14 Nvic_SetFault()	27
4.6.2.15 Nvic_SetGroupPriority()	27
4.6.2.16 Nvic_SetPending()	27
4.6.2.17 Nvic_SetSubpriority()	28
4.7 Nvic.h File Reference	28
4.7.1 Detailed Description	30
4.7.2 Macro Definition Documentation	31
4.7.2.1 NVIC_IRQNUM_WWDG	31

4.7.3 Function Documentation	31
4.7.3.1 Nvic_ClearFault()	31
4.7.3.2 Nvic_ClearPending()	31
4.7.3.3 Nvic_ConfigGroupSize()	32
4.7.3.4 Nvic_DisableInterrupt()	32
4.7.3.5 Nvic_DisablePeripheral()	32
4.7.3.6 Nvic_EnableInterrupt()	32
4.7.3.7 Nvic_EnablePeripheral()	33
4.7.3.8 Nvic_FilterPriority()	33
4.7.3.9 Nvic_GenerateSoftwareInterrupt()	33
4.7.3.10 Nvic_GetGroupPriority()	34
4.7.3.11 Nvic_GetSubpriority()	34
4.7.3.12 Nvic_IsInterruptActive()	35
4.7.3.13 Nvic_ResetSystem()	35
4.7.3.14 Nvic_SetFault()	35
4.7.3.15 Nvic_SetGroupPriority()	36
4.7.3.16 Nvic_SetPending()	36
4.7.3.17 Nvic_SetSubpriority()	36
4.8 Protocol.c File Reference	37
4.8.1 Detailed Description	38
4.8.2 Function Documentation	38
4.8.2.1 Protocol_Receive()	38
4.8.2.2 Protocol_Send()	39
4.8.2.3 Protocol_SendAck()	39
4.9 Protocol.h File Reference	39
4.9.1 Detailed Description	40
4.9.2 Function Documentation	40
4.9.2.1 Protocol_Receive()	40
4.9.2.2 Protocol_Send()	41
4.9.2.3 Protocol_SendAck()	41
4.10 Rcc.c File Reference	41
4.10.1 Detailed Description	43
4.10.2 Function Documentation	43
4.10.2.1 Rcc_GetSystemClockStatus()	43
4.10.2.2 Function: Rcc_GetSystemClockStatus	43
4.10.2.3 Rcc_IsClockReady()	43
4.10.2.4 Function: Rcc_IsReady	43
4.10.2.5 Rcc_ResetAhbPeriph()	44
4.10.2.6 Function: Rcc_ResetAhbPeriph	44
4.10.2.7 Rcc_ResetApb1Periph()	44
4.10.2.8 Function: Rcc_ResetApb1Periph	45
4.10.2.9 Rcc_ResetApb2Periph()	46

4.10.2.10 Function: Rcc_ResetApb2Periph	46
4.10.2.11 Rcc_SelectMcoClock()	47
4.10.2.12 Function: Rcc_SelectMcoClock	47
4.10.2.13 Rcc_SetAhbPeriphClockState()	48
4.10.2.14 Function: Rcc_SetAhbPeriphClockState	48
4.10.2.15 Rcc_SetApb1PeriphClockState()	49
4.10.2.16 Function: Rcc_SetApb1PeriphClockState	49
4.10.2.17 Rcc_SetApb2PeriphClockState()	50
4.10.2.18 Function: Rcc_SetApb2PeriphClockState	51
4.10.2.19 Rcc_SetClockState()	52
4.10.2.20 Function: Rcc_SetClockState	52
4.10.2.21 Rcc_SetPllMultiplier()	52
4.10.2.22 Function: Rcc_SetPllMultiplier	52
4.10.2.23 Rcc_SetPllSource()	53
4.10.2.24 Function: Rcc_SetPllSource	53
4.10.2.25 Rcc_SetPrescaler()	53
4.10.2.26 Function: Rcc_SetPrescaler	53
4.10.2.27 Rcc_SwitchSystemClock()	54
4.10.2.28 Function: Rcc_SwitchSystemClock	54
4.11 Rcc.h File Reference	54
4.11.1 Detailed Description	58
4.11.2 Function Documentation	59
4.11.2.1 Rcc_GetSystemClockStatus()	59
4.11.2.2 Function: Rcc_GetSystemClockStatus	59
4.11.2.3 Function: Rcc_GetSystemClockStatus	59
4.11.2.4 Rcc_IsClockReady()	59
4.11.2.5 Function: Rcc_IsReady	59
4.11.2.6 Function: Rcc_IsReady	60
4.11.2.7 Rcc_ResetAhbPeriph()	60
4.11.2.8 Function: Rcc_ResetAhbPeriph	60
4.11.2.9 Function: Rcc_ResetAhbPeriph	61
4.11.2.10 Rcc_ResetApb1Periph()	61
4.11.2.11 Function: Rcc_ResetApb1Periph	61
4.11.2.12 Function: Rcc_ResetApb1Periph	62
4.11.2.13 Rcc_ResetApb2Periph()	63
4.11.2.14 Function: Rcc_ResetApb2Periph	63
4.11.2.15 Function: Rcc_ResetApb2Periph	64
4.11.2.16 Rcc_SelectMcoClock()	65
4.11.2.17 Function: Rcc_SelectMcoClock	65
4.11.2.18 Function: Rcc_SelectMcoClock	66
4.11.2.19 Rcc_SetAhbPeriphClockState()	66
4.11.2.20 Function: Rcc_SetAhbPeriphClockState	66

4.11.2.21 Function: Rcc_SetAhbPeriphClockState	67
4.11.2.22 Rcc_SetApb1PeriphClockState()	68
4.11.2.23 Function: Rcc_SetApb1PeriphClockState	68
4.11.2.24 Function: Rcc_SetApb1PeriphClockState	69
4.11.2.25 Rcc_SetApb2PeriphClockState()	70
4.11.2.26 Function: Rcc_SetApb2PeriphClockState	71
4.11.2.27 Function: Rcc_SetApb2PeriphClockState	71
4.11.2.28 Rcc_SetClockState()	72
4.11.2.29 Function: Rcc_SetClockState	72
4.11.2.30 Function: Rcc_SetClockState	73
4.11.2.31 Rcc_SetPllMultiplier()	73
4.11.2.32 Function: Rcc_SetPllMultiplier	73
4.11.2.33 Function: Rcc_SetPllMultiplier	74
4.11.2.34 Rcc_SetPllSource()	74
4.11.2.35 Function: Rcc_SetPllSource	74
4.11.2.36 Function: Rcc_SetPllSource	74
4.11.2.37 Rcc_SetPrescaler()	75
4.11.2.38 Function: Rcc_SetPrescaler	75
4.11.2.39 Function: Rcc_SetPrescaler	76
4.11.2.40 Rcc_SwitchSystemClock()	76
4.11.2.41 Function: Rcc_SwitchSystemClock	76
4.11.2.42 Function: Rcc_SwitchSystemClock	77
4.12 Switch.c File Reference	77
4.12.1 Detailed Description	78
4.12.2 Function Documentation	78
4.12.2.1 Switch_GetSwitchStatus()	78
4.12.2.2 Function: Switch_GetSwitchStatus	78
4.12.2.3 Switch_Init()	78
4.12.2.4 Function: Switch_Init	79
4.13 Switch.h File Reference	79
4.13.1 Detailed Description	79
4.13.2 Function Documentation	79
4.13.2.1 Switch_GetSwitchStatus()	79
4.13.2.2 Function: Switch_GetSwitchStatus	79
4.13.2.3 Function: Switch_GetSwitchStatus	80
4.13.2.4 Switch_Init()	80
4.13.2.5 Function: Switch_Init	80
4.13.2.6 Function: Switch_Init	80
4.14 Switch_Cfg.c File Reference	81
4.14.1 Detailed Description	81
4.14.2 Variable Documentation	81
4.14.2.1 Switch_switches	81

4.15 Switch_Cfg.h File Reference	81
4.15.1 Detailed Description	82
4.16 Uart.c File Reference	82
4.16.1 Detailed Description	84
4.16.2 Function Documentation	84
4.16.2.1 Uart_Init()	84
4.16.2.2 Uart_Receive()	85
4.16.2.3 Uart_ReceiveSync()	85
4.16.2.4 Uart_Send()	86
4.16.2.5 Uart_SendBreak()	86
4.16.2.6 Uart_SendSync()	87
4.16.2.7 Uart_SetBreakCb()	87
4.16.2.8 Uart_SetRxCb()	88
4.16.2.9 Uart_SetTxCb()	88
4.16.2.10 USART1_IRQHandler()	89
4.16.2.11 USART2_IRQHandler()	89
4.16.2.12 USART3_IRQHandler()	89
4.16.3 Variable Documentation	89
4.16.3.1 Uart_Address	89
4.16.3.2 Uart_DmaRxChannelNumber	90
4.16.3.3 Uart_DmaTxChannelNumber	90
4.17 Uart.h File Reference	90
4.17.1 Detailed Description	91
4.17.2 Function Documentation	91
4.17.2.1 Uart_Init()	91
4.17.2.2 Uart_Receive()	92
4.17.2.3 Uart_ReceiveSync()	93
4.17.2.4 Uart_Send()	93
4.17.2.5 Uart_SendBreak()	94
4.17.2.6 Uart_SendSync()	94
4.17.2.7 Uart_SetBreakCb()	95
4.17.2.8 Uart_SetRxCb()	95
4.17.2.9 Uart_SetTxCb()	96
4.18 Uart_Cfg.h File Reference	96
4.18.1 Detailed Description	96
Index	97

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

dataBuffer_t	The Uart Data Buffer	5
Fpec_t	5
gpio_t	The GPIO Type contains GPIO configurations	6
gpioReg_t	6
header_t	6
nvic_t	The NVIC Registers	7
packet_t	7
switch_t	The Switch pin layout	7
Uart_cfg_t	8
uart_t	The UART Registers	8

Chapter 2

File Index

2.1 File List

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

Bootloader.c	A Simple Bootloader Application Over Uart Using A Designed Software Protocol	9
Fpec.h	This is the user interface for the FPEC Driver	10
Gpio.c	This file is to be used as an implementation of the GPIO driver	13
Gpio.h	This file is to be used as an interface for the user of GPIO driver	15
HRcc.h	This is the user interface for the RCC Handler	19
Nvic.c	This file is an implementation for the NVIC driver for the arm cortex m3	20
Nvic.h	This file is a user interface for the NVIC driver for the arm cortex m3	28
Protocol.c	This is the implementation for a self designed software protocol	37
Protocol.h	This is the user interface for the self designed software protocol	39
Rcc.c	This file is to be used as an implementation of the RCC driver	41
Rcc.h	This file is to be used as an interface for the user of RCC driver	54
Std_Types.h		??
Switch.c	This file is to be used as an implementation for the Switch Handler	77
Switch.h	This file is to be used as an interface for the user of the Switch Handler	79
Switch_Cfg.c	This file is to be used as an implementation of the configurations the user configured in the	
Switch_Cfg.h		81
Switch_Cfg.h	This file is to be given to the user to configure the Switch Handler	81
Uart.c	This is the implementation for the UART driver	82
Uart.h	This is the user interface for the UART driver	90
Uart_Cfg.h	Those are the user configurations for the Uart Driver	96

Chapter 3

Data Structure Documentation

3.1 `dataBuffer_t` Struct Reference

The Uart Data Buffer.

Data Fields

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

3.1.1 Detailed Description

The Uart Data Buffer.

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

- [Uart.c](#)

3.2 `Fpec_t` Struct Reference

Data Fields

- `uint32_t ACR`
- `uint32_t KEYR`
- `uint32_t OPTKEYR`
- `uint32_t SR`
- `uint32_t CR`
- `uint32_t AR`
- `uint32_t DUMMY`
- `uint32_t OBR`
- `uint32_t WRPR`

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

- `Fpec.c`

3.3 gpio_t Struct Reference

The GPIO Type contains GPIO configurations.

```
#include <Gpio.h>
```

Data Fields

- uint32_t **pins**
- uint32_t **speed**
- uint32_t **mode**
- uint32_t **port**

3.3.1 Detailed Description

The GPIO Type contains GPIO configurations.

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

- [Gpio.h](#)

3.4 gpioReg_t Struct Reference

Data Fields

- uint64_t **CR**
- uint32_t **IDR**
- uint32_t **ODR**
- uint32_t **BSR**
- uint32_t **BRR**
- uint32_t **LCK**

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

- [Gpio.c](#)

3.5 header_t Struct Reference

Data Fields

- uint32_t **key**
- uint16_t **type**
- uint16_t **length**

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

- [Protocol.c](#)

3.6 nvic_t Struct Reference

The NVIC Registers.

Data Fields

- uint32_t **SETEN** [8]
- uint32_t **_RESERVED0** [24]
- uint32_t **CLREN** [8]
- uint32_t **_RSERVED1** [24]
- uint32_t **SETPND** [8]
- uint32_t **_RESERVED2** [24]
- uint32_t **CLRPND** [8]
- uint32_t **_RESERVED3** [24]
- uint32_t **AB** [8]
- uint32_t **_RESERVED4** [56]
- uint8_t **PRI** [240]
- uint32_t **_RESERVED5** [644]
- uint32_t **STIR**

3.6.1 Detailed Description

The NVIC Registers.

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

- [Nvic.c](#)

3.7 packet_t Struct Reference

Data Fields

- ```
union {
 header_t header
 uint8_t headerData [PROTOCOL_HEADER_SIZE]
};
```

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

- [Protocol.c](#)

## 3.8 switch\_t Struct Reference

The Switch pin layout.

```
#include <Switch.h>
```

## Data Fields

- uint32\_t **pin**
- uint32\_t **port**
- uint8\_t **activeState**

### 3.8.1 Detailed Description

The Switch pin layout.

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

- [Switch.h](#)

## 3.9 Uart\_cfg\_t Struct Reference

### Data Fields

- uint32\_t **baudRate**
- uint32\_t **stopBits**
- uint32\_t **parity**
- uint32\_t **flowControl**
- uint32\_t **sysClk**
- uint32\_t **linEn**
- uint8\_t **interrupts**
- uint8\_t **uartModule**

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

- [Uart.h](#)

## 3.10 uart\_t Struct Reference

The UART Registers.

### Data Fields

- uint32\_t **SR**
- uint32\_t **DR**
- uint32\_t **BRR**
- uint32\_t **CR1**
- uint32\_t **CR2**
- uint32\_t **CR3**
- uint32\_t **GTPR**

### 3.10.1 Detailed Description

The UART Registers.

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

- [Uart.c](#)



## Chapter 4

# File Documentation

### 4.1 Bootloader.c File Reference

A Simple Bootloader Application Over Uart Using A Designed Software Protocol.

```
#include "Std_Types.h"
#include "Uart.h"
#include "Fpec.h"
#include "Rcc.h"
#include "HRcc.h"
#include "Nvic.h"
#include "Gpio.h"
#include "Switch.h"
#include "Protocol.h"
```

#### Macros

- #define **MAX\_DATA\_TO\_BUFFER** 1024
- #define **APP\_EXIST** 0xAAAA
- #define **APP\_NOT\_EXIST** 0xBBBB

#### Typedefs

- typedef void(\* **app\_t**) (void)

#### Functions

- int **main** (void)

#### Variables

- const uint16\_t \* **appExistMarker** = (uint16\_t\*)0x0800FFE0
- const uint16\_t \* **addressMarker** = (uint16\_t\*)0x0800FFD0

### 4.1.1 Detailed Description

A Simple Bootloader Application Over Uart Using A Designed Software Protocol.

**Author**

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com))

**Version**

0.1

**Date**

2020-05-25

**Copyright**

Copyright (c) 2020

## 4.2 Fpec.h File Reference

This is the user interface for the FPEC Driver.

### Functions

- Std\_ReturnType [Fpec\\_Lock](#) (void)  
*Locks The FPEC.*
- Std\_ReturnType [Fpec\\_Unlock](#) (void)  
*Unlocks The FPEC.*
- Std\_ReturnType [Fpec\\_WriteHalfWord](#) (uint16\_t \*address, uint16\_t data)  
*Writes A Half Word To The Flash.*
- Std\_ReturnType [Fpec\\_WriteBlock](#) (uint16\_t \*flashAddress, uint16\_t \*srcAddress, uint16\_t blockSize)  
*Writes A Block To The Flash.*
- Std\_ReturnType [Fpec\\_ErasePage](#) (uint32\_t \*pageAddress)  
*Erases A Page In The Flash.*
- Std\_ReturnType [Fpec\\_MassErase](#) (void)  
*Erases The Flash Completely !!*

### 4.2.1 Detailed Description

This is the user interface for the FPEC Driver.

This is the implementation for the FPEC Driver.

**Author**

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com))

**Version**

0.1

**Date**

2020-05-09

**Copyright**

Copyright (c) 2020

## 4.2.2 Function Documentation

### 4.2.2.1 Fpec\_ErasePage()

```
Std_ReturnType Fpec_ErasePage (
 uint32_t * pageAddress)
```

Erases A Page In The Flash.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <i>pageAddress</i> | The Address Of The Page |
|--------------------|-------------------------|

#### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

### 4.2.2.2 Fpec\_Lock()

```
Std_ReturnType Fpec_Lock (
 void)
```

Locks The FPEC.

#### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

### 4.2.2.3 Fpec\_MassErase()

```
Std_ReturnType Fpec_MassErase (
 void)
```

Erases The Flash Completely !!

#### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

#### 4.2.2.4 Fpec\_Unlock()

```
Std_ReturnType Fpec_Unlock (
 void)
```

Unlocks The FPEC.

##### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

#### 4.2.2.5 Fpec\_WriteBlock()

```
Std_ReturnType Fpec_WriteBlock (
 uint16_t * flashAddress,
 uint16_t * srcAddress,
 uint16_t blockSize)
```

Writes A Block To The Flash.

##### Parameters

|                     |                                       |
|---------------------|---------------------------------------|
| <i>flashAddress</i> | The Address In Flash                  |
| <i>srcAddress</i>   | The Source Address To Fetch Data From |
| <i>blockSize</i>    | The Size Of The Block In Half Words   |

##### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

#### 4.2.2.6 Fpec\_WriteHalfWord()

```
Std_ReturnType Fpec_WriteHalfWord (
 uint16_t * address,
 uint16_t data)
```

Writes A Half Word To The Flash.

##### Parameters

|                |                        |
|----------------|------------------------|
| <i>address</i> | The Address In Flash   |
| <i>data</i>    | The Half Word To Write |

#### Returns

Std\_ReturnType A Status E\_OK : If The Function Executed Successfully E\_NOT\_OK : If The Function Didn't Execute Successfully

## 4.3 Gpio.c File Reference

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

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

### Data Structures

- struct [gpioReg\\_t](#)

### Macros

- #define **GPIO\_MODE\_INPUT\_MASK** 0xF0
- #define **GPIO\_MODE\_MASK** 0x0C

### Functions

- Std\_ReturnType [Gpio\\_InitPins](#) ([gpio\\_t](#) \*gpio)  
*Initializes pins mode and speed for a specific port.*
- Std\_ReturnType [Gpio\\_WritePin](#) (uint32\_t port, uint32\_t pin, uint32\_t pinStatus)  
*Write a value to a pin(0/1)*
- Std\_ReturnType [Gpio\\_ReadPin](#) (uint32\_t port, uint32\_t pin, uint8\_t \*state)  
*Reads a value to a pin(0/1)*

#### 4.3.1 Detailed Description

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

##### Author

Mark Attia

##### Date

February 6, 2020

#### 4.3.2 Function Documentation

##### 4.3.2.1 Gpio\_InitPins()

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

Initializes pins mode and speed for a specific port.

##### 4.3.2.2 Function: Gpio\_InitPins

**Parameters**

|             |                                                          |
|-------------|----------------------------------------------------------|
| <i>gpio</i> | An object of type <a href="#">gpio_t</a> 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

**4.3.2.3 Gpio\_ReadPin()**

```
Std_ReturnType Gpio_ReadPin (
 uint32_t port,
 uint32_t pin,
 uint8_t * state)
```

Reads a value to a pin(0/1)

**4.3.2.4 Function: Gpio\_ReadPin****Parameters**

|              |                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>  | The port you want to read from <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to read from</li> </ul>                       |
| <i>pin</i>   | The pin you want to read <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\</li> </ul>  |
| <i>state</i> | To return a status in <ul style="list-style-type: none"> <li>GPIO_PIN_SET : The pin is set to 1</li> <li>GPIO_PIN_RESET : The pin is set to 0</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.3.2.5 Gpio\_WritePin()**

```
Std_ReturnType Gpio_WritePin (
 uint32_t port,
 uint32_t pin,
 uint32_t pinStatus)
```

Write a value to a pin(0/1)

**4.3.2.6 Function: Gpio\_WritePin**

## Parameters

|                  |                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>      | The port you want to configure <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to configure</li> </ul>                                                                |
| <i>pin</i>       | The pin you want to configure <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\</li> </ul>                                 |
| <i>pinStatus</i> | The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> <li>GPIO_PIN_SET : Sets the pin value to 1</li> <li>GPIO_PIN_RESET : Resets the pin value to 0</li> </ul> |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.4 Gpio.h File Reference

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

### Data Structures

- struct [gpio\\_t](#)  
*The GPIO Type contains GPIO configurations.*

### Macros

- #define GPIO\_PIN\_SET 0
- #define GPIO\_PIN\_RESET !GPIO\_PIN\_SET
- #define GPIO\_PIN\_0 0x0001
- #define GPIO\_PIN\_1 0x0002
- #define GPIO\_PIN\_2 0x0004
- #define GPIO\_PIN\_3 0x0008
- #define GPIO\_PIN\_4 0x0010
- #define GPIO\_PIN\_5 0x0020
- #define GPIO\_PIN\_6 0x0040
- #define GPIO\_PIN\_7 0x0080
- #define GPIO\_PIN\_8 0x0100
- #define GPIO\_PIN\_9 0x0200
- #define GPIO\_PIN\_10 0x0400
- #define GPIO\_PIN\_11 0x0800
- #define GPIO\_PIN\_12 0x1000
- #define GPIO\_PIN\_13 0x2000
- #define GPIO\_PIN\_14 0x4000
- #define GPIO\_PIN\_15 0x8000
- #define GPIO\_PIN\_ALL 0xFFFF
- #define GPIO\_SPEED\_10\_MHZ 0x01
- #define GPIO\_SPEED\_02\_MHZ 0x02

- `#define GPIO_SPEED_50_MHZ 0x03`
- `#define GPIO_MODE_GP_OUTPUT_PP 0x00`
- `#define GPIO_MODE_GP_OUTPUT_OD 0x04`
- `#define GPIO_MODE_AF_OUTPUT_PP 0x08`
- `#define GPIO_MODE_AF_OUTPUT_OD 0x0C`
- `#define GPIO_MODE_INPUT_ANALOG 0x10`
- `#define GPIO_MODE_INPUT_FLOATING 0x14`
- `#define GPIO_MODE_INPUT_PULL_DOWN 0x18`
- `#define GPIO_MODE_INPUT_PULL_UP 0x28`
- `#define GPIO_PORTA (uint32_t)0x40010800`
- `#define GPIO_PORTB (uint32_t)0x40010C00`
- `#define GPIO_PORTC (uint32_t)0x40011000`
- `#define GPIO_PORTD (uint32_t)0x40011400`
- `#define GPIO_PORTE (uint32_t)0x40011800`
- `#define GPIO_PORTF (uint32_t)0x40011C00`
- `#define GPIO_PORTG (uint32_t)0x40012000`

## Functions

- Std\_ReturnType [Gpio\\_InitPins](#) ([gpio\\_t](#) \*gpio)  
*Initializes pins mode and speed for a specific port.*
- Std\_ReturnType [Gpio\\_WritePin](#) (uint32\_t port, uint32\_t pin, uint32\_t pinStatus)  
*Write a value to a pin(0/1)*
- Std\_ReturnType [Gpio\\_ReadPin](#) (uint32\_t port, uint32\_t pin, uint8\_t \*state)  
*Reads a value to a pin(0/1)*

### 4.4.1 Detailed Description

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

#### Author

Mark Attia

#### Date

February 6, 2020

### 4.4.2 Function Documentation

#### 4.4.2.1 Gpio\_InitPins()

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

Initializes pins mode and speed for a specific port.

#### 4.4.2.2 Function: Gpio\_InitPins



## Parameters

|             |                                                          |
|-------------|----------------------------------------------------------|
| <i>gpio</i> | An object of type <a href="#">gpio_t</a> 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

**4.4.2.3 Function: Gpio\_InitPins**

## Parameters

|             |                                                          |
|-------------|----------------------------------------------------------|
| <i>gpio</i> | An object of type <a href="#">gpio_t</a> 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

**4.4.2.4 Gpio\_ReadPin()**

```
Std_ReturnType Gpio_ReadPin (
 uint32_t port,
 uint32_t pin,
 uint8_t * state)
```

Reads a value to a pin(0/1)

**4.4.2.5 Function: Gpio\_ReadPin**

## Parameters

|              |                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>  | The port you want to read from <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to read from</li> </ul>                       |
| <i>pin</i>   | The pin you want to read <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\</li> </ul>  |
| <i>state</i> | To return a status in <ul style="list-style-type: none"> <li>GPIO_PIN_SET : The pin is set to 1</li> <li>GPIO_PIN_RESET : The pin is set to 0</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.4.2.6 Function: Gpio\_ReadPin****Parameters**

|              |                                                                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>  | The port you want to read from <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to read from</li> </ul>                       |
| <i>pin</i>   | The pin you want to read <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to read //You can OR more than one pin\</li> </ul>  |
| <i>state</i> | To return a status in <ul style="list-style-type: none"> <li>GPIO_PIN_SET : The pin is set to 1</li> <li>GPIO_PIN_RESET : The pin is set to 0</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.4.2.7 Gpio\_WritePin()**

```
Std_ReturnType Gpio_WritePin (
 uint32_t port,
 uint32_t pin,
 uint32_t pinStatus)
```

Write a value to a pin(0/1)

**4.4.2.8 Function: Gpio\_WritePin****Parameters**

|                  |                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>      | The port you want to configure <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to configure</li> </ul>                                                                |
| <i>pin</i>       | The pin you want to configure <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\</li> </ul>                                 |
| <i>pinStatus</i> | The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> <li>GPIO_PIN_SET : Sets the pin value to 1</li> <li>GPIO_PIN_RESET : Resets the pin value to 0</li> </ul> |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.4.2.9 Function: Gpio\_WritePin

## Parameters

|                  |                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>port</i>      | The port you want to configure <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to configure</li> </ul>                                                                |
| <i>pin</i>       | The pin you want to configure <ul style="list-style-type: none"> <li>GPIO_PIN_X : The pin number you want to configure //You can OR more than one pin\</li> </ul>                                 |
| <i>pinStatus</i> | The status of the pins (GPIO_PIN_SET/GPIO_PIN_RESET) <ul style="list-style-type: none"> <li>GPIO_PIN_SET : Sets the pin value to 1</li> <li>GPIO_PIN_RESET : Resets the pin value to 0</li> </ul> |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.5 HRcc.h File Reference

This is the user interface for the RCC Handler.

## Functions

- Std\_ReturnType [HRcc\\_SystemClockInit](#) (void)  
*This function initializes the system clock.*
- Std\_ReturnType [HRcc\\_EnPortClock](#) (uint32\_t port)  
*This function initializes the clock for a specific GPIO port.*

## 4.5.1 Detailed Description

This is the user interface for the RCC Handler.

This is implementation for the RCC Handler.

## Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com))

## Version

0.1

## Date

2020-03-24

## Copyright

Copyright (c) 2020

## 4.5.2 Function Documentation

### 4.5.2.1 HRcc\_EnPortClock()

```
Std_ReturnType HRcc_EnPortClock (
 uint32_t port)
```

This function initializes the clock for a specific GPIO port.

#### Parameters

|             |                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------|
| <i>port</i> | The GPIO port <ul style="list-style-type: none"> <li>GPIO_PORTX : The pin number you want to configure</li> </ul> |
|-------------|-------------------------------------------------------------------------------------------------------------------|

#### Returns

Std\_ReturnType

E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

### 4.5.2.2 HRcc\_SystemClockInit()

```
Std_ReturnType HRcc_SystemClockInit (
 void)
```

This function initializes the system clock.

#### Returns

Std\_ReturnType E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.6 Nvic.c File Reference

This file is an implementation for the NVIC driver for the arm cortex m3.

```
#include "Std_Types.h"
#include "Nvic.h"
#include "Nvic_Cfg.h"
```

### Data Structures

- struct [nvic\\_t](#)

*The NVIC Registers.*

## Macros

- `#define NVIC_BASE_ADDRESS 0xE000E100`
- `#define AIRC *((volatile uint32_t*)0xE000ED0C) /* Application interrupt and reset control register */`
- `#define AIRC_LOCK 0x05FA0000 /* Application interrupt and reset control register Lock */`
- `#define AIRC_LOCK_CLR 0x0000FFFF /* Application interrupt and reset control register Lock Clear Mask */`
- `#define AIRC_SYS_RST 0x00000004`
- `#define AIRC_GROUP_CLR 0x0000F8FF`
- `#define NVIC_GROUP_CHECK 0xFFFFFFFF8`
- `#define NVIC_NON_IMPLEMENTED_PRI (8 - NVIC_GROUP_SIZE - NVIC_SUBGROUP_SIZE)`
- `#define NVIC_0_BIT_MASK 0b0`
- `#define NVIC_1_BIT_MASK 0b1`
- `#define NVIC_2_BIT_MASK 0b11`
- `#define NVIC_3_BIT_MASK 0b111`
- `#define NVIC_4_BIT_MASK 0b1111`
- `#define NVIC_5_BIT_MASK 0b11111`
- `#define NVIC_6_BIT_MASK 0b111111`
- `#define NVIC_7_BIT_MASK 0b1111111`
- `#define NVIC_CONCAT_MASK(x) NVIC_CONCAT_MASK_HELP(x)`
- `#define NVIC_CONCAT_MASK_HELP(x) NVIC_###x###_BIT_MASK`
- `#define NVIC ((volatile nvic_t*)(NVIC_BASE_ADDRESS))`

## Functions

- Std\_ReturnType [Nvic\\_EnableInterrupt](#) (uint8\_t intNumber)  
*Enables a specific Interrupt.*
- Std\_ReturnType [Nvic\\_DisableInterrupt](#) (uint8\_t intNumber)  
*Disables a specific Interrupt.*
- Std\_ReturnType [Nvic\\_SetPending](#) (uint8\_t intNumber)  
*Sets the pending flag for a specific interrupt.*
- Std\_ReturnType [Nvic\\_ClearPending](#) (uint8\_t intNumber)  
*Clears the pending flag for a specific interrupt.*
- Std\_ReturnType [Nvic\\_IsInterruptActive](#) (uint8\_t \*activeState, uint8\_t intNumber)  
*Checks if the interrupt is active.*
- Std\_ReturnType [Nvic\\_SetSubpriority](#) (uint8\_t priority, uint8\_t intNumber)  
*Sets the subpriority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_GetSubpriority](#) (uint8\_t \*priority, uint8\_t intNumber)  
*Gets the subpriority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_SetGroupPriority](#) (uint8\_t priority, uint8\_t intNumber)  
*Sets the group priority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_GetGroupPriority](#) (uint8\_t \*priority, uint8\_t intNumber)  
*Gets the group priority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_ConfigGroupSize](#) (void)  
*Configure the group size.*
- Std\_ReturnType [Nvic\\_ResetSystem](#) (void)  
*Resets the microcontroller.*
- Std\_ReturnType [Nvic\\_EnablePeripheral](#) (void)  
*Enables the peripherals interrupts.*
- Std\_ReturnType [Nvic\\_DisablePeripheral](#) (void)  
*Disable the peripherals interrupts.*
- Std\_ReturnType [Nvic\\_SetFault](#) (void)

- Blocks all interrupts including hard fault.*
- Std\_ReturnType [Nvic\\_ClearFault](#) (void)  
*Returns from fault mode.*
- Std\_ReturnType [Nvic\\_FilterPriority](#) (uint8\_t pri)  
*Only allow interrupts over a certain priority.*
- Std\_ReturnType [Nvic\\_GenerateSoftwareInterrupt](#) (uint8\_t intNumber)  
*Generates a software interrupt (Atomic function to generate interrupt immediately)*

### 4.6.1 Detailed Description

This file is an implementation for the NVIC driver for the arm cortex m3.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com))

#### Version

0.1

#### Date

2020-02-29

#### Copyright

Copyright (c) 2020

### 4.6.2 Function Documentation

#### 4.6.2.1 Nvic\_ClearFault()

```
Std_ReturnType Nvic_ClearFault (
 void)
```

Returns from fault mode.

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.2 Nvic\_ClearPending()

```
Std_ReturnType Nvic_ClearPending (
 uint8_t intNumber)
```

Clears the pending flag for a specific interrupt.

## Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|-----------------------------------------------------------------------------------------------------------------|

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.3 Nvic\_ConfigGroupSize()**

```
Std_ReturnType Nvic_ConfigGroupSize (
 void)
```

Configure the group size.

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.4 Nvic\_DisableInterrupt()**

```
Std_ReturnType Nvic_DisableInterrupt (
 uint8_t intNumber)
```

Disables a specific Interrupt.

## Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|-----------------------------------------------------------------------------------------------------------------|

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.5 Nvic\_DisablePeripheral()**

```
Std_ReturnType Nvic_DisablePeripheral (
 void)
```

Disable the peripherals interrupts.

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.6 Nvic\_EnableInterrupt()**

```
Std_ReturnType Nvic_EnableInterrupt (
 uint8_t intNumber)
```

Enables a specific Interrupt.

**Parameters**

|                  |                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table<br><ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|--------------------------------------------------------------------------------------------------------------------|

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.7 Nvic\_EnablePeripheral()**

```
Std_ReturnType Nvic_EnablePeripheral (
 void)
```

Enables the peripherals interrupts.

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.8 Nvic\_FilterPriority()**

```
Std_ReturnType Nvic_FilterPriority (
 uint8_t pri)
```

Only allow interrupts over a certain priority.

**Parameters**

|            |                              |
|------------|------------------------------|
| <i>per</i> | the minimum priority allowed |
|------------|------------------------------|



**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.9 Nvic\_GenerateSoftwareInterrupt()**

```
Std_ReturnType Nvic_GenerateSoftwareInterrupt (
 uint8_t intNumber)
```

Generates a software interrupt (Atomic function to generate interrupt immediately)

**Parameters**

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table<br>• NVIC_IRQNUM_X |
|------------------|--------------------------------------------------------------------|

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.10 Nvic\_GetGroupPriority()**

```
Std_ReturnType Nvic_GetGroupPriority (
 uint8_t * priority,
 uint8_t intNumber)
```

Gets the group priority for aspecific interrupt.

**Parameters**

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to get                                       |
| <i>intNumber</i> | the number of the interrupt in the vector table<br>• NVIC_IRQNUM_X |

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.11 Nvic\_GetSubpriority()**

```
Std_ReturnType Nvic_GetSubpriority (
 uint8_t * priority,
 uint8_t intNumber)
```

Gets the subpriority for aspecific interrupt.

#### Parameters

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to get                                                                                      |
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"> <li>• NVIC_IRQNUM_X</li> </ul> |

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.12 Nvic\_IsInterruptActive()

```
Std_ReturnType Nvic_IsInterruptActive (
 uint8_t * activeState,
 uint8_t intNumber)
```

Checks if the interrupt is active.

#### Parameters

|                    |                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------|
| <i>activeState</i> | the active state of the interrupt <ul style="list-style-type: none"> <li>• NVIC_ACTIVE</li> <li>• NVIC_NOT_ACTIVE</li> </ul> |
| <i>intNumber</i>   | the number of the interrupt in the vector table <ul style="list-style-type: none"> <li>• NVIC_IRQNUM_X</li> </ul>            |

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.13 Nvic\_ResetSystem()

```
Std_ReturnType Nvic_ResetSystem (
 void)
```

Resets the microcontroller.

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.14 Nvic\_SetFault()

```
Std_ReturnType Nvic_SetFault (
 void)
```

Blocks all interrupts including hard fault.

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.15 Nvic\_SetGroupPriority()

```
Std_ReturnType Nvic_SetGroupPriority (
 uint8_t priority,
 uint8_t intNumber)
```

Sets the group priority for aspecific interrupt.

##### Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to set                                                                                    |
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.6.2.16 Nvic\_SetPending()

```
Std_ReturnType Nvic_SetPending (
 uint8_t intNumber)
```

Sets the pending flag for a specific interrupt.

##### Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|-----------------------------------------------------------------------------------------------------------------|

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.6.2.17 Nvic\_SetSubpriority()**

```
Std_ReturnType Nvic_SetSubpriority (
 uint8_t priority,
 uint8_t intNumber)
```

Sets the subpriority for aspecific interrupt.

**Parameters**

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to set                                                                                      |
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"> <li>• NVIC_IRQNUM_X</li> </ul> |

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7 Nvic.h File Reference**

This file is a user interface for the NVIC driver for the arm cortex m3.

**Macros**

- #define **NVIC\_ACTIVE** 0
- #define **NVIC\_NOT\_ACTIVE** !NVIC\_ACTIVE
- #define **NVIC\_IRQNUM\_WWDG** 0
- #define **NVIC\_IRQNUM\_PVD** 1
- #define **NVIC\_IRQNUM\_TAMPER** 2
- #define **NVIC\_IRQNUM\_RTC** 3
- #define **NVIC\_IRQNUM\_FLASH** 4
- #define **NVIC\_IRQNUM\_RCC** 5
- #define **NVIC\_IRQNUM\_EXTI0** 6
- #define **NVIC\_IRQNUM\_EXTI1** 7
- #define **NVIC\_IRQNUM\_EXTI2** 8
- #define **NVIC\_IRQNUM\_EXTI3** 9
- #define **NVIC\_IRQNUM\_EXTI4** 10
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL1** 11
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL2** 12
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL3** 13
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL4** 14
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL5** 15

- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL6** 16
- #define **NVIC\_IRQNUM\_DMA1\_CHANNEL7** 17
- #define **NVIC\_IRQNUM\_ADC1\_2** 18
- #define **NVIC\_IRQNUM\_USB\_HP\_CAN\_TX** 19
- #define **NVIC\_IRQNUM\_USB\_HP\_CAN\_RX0** 20
- #define **NVIC\_IRQNUM\_CAN\_RX1** 21
- #define **NVIC\_IRQNUM\_CAN\_SCE** 22
- #define **NVIC\_IRQNUM\_EXTI9\_5** 23
- #define **NVIC\_IRQNUM\_TIM1\_BRK** 24
- #define **NVIC\_IRQNUM\_TIM1\_UP** 25
- #define **NVIC\_IRQNUM\_TIM1\_TRG\_COM** 26
- #define **NVIC\_IRQNUM\_TIM1\_CC** 27
- #define **NVIC\_IRQNUM\_TIM2** 28
- #define **NVIC\_IRQNUM\_TIM3** 29
- #define **NVIC\_IRQNUM\_TIM4** 30
- #define **NVIC\_IRQNUM\_I2C1\_EV** 31
- #define **NVIC\_IRQNUM\_I2C1\_ER** 32
- #define **NVIC\_IRQNUM\_I2C2\_EV** 33
- #define **NVIC\_IRQNUM\_I2C2\_ER** 34
- #define **NVIC\_IRQNUM\_SPI1** 35
- #define **NVIC\_IRQNUM\_SPI2** 36
- #define **NVIC\_IRQNUM\_USART1** 37
- #define **NVIC\_IRQNUM\_USART2** 38
- #define **NVIC\_IRQNUM\_USART3** 39
- #define **NVIC\_IRQNUM\_EXTI15\_10** 40
- #define **NVIC\_IRQNUM\_RTC\_ALARM** 41
- #define **NVIC\_IRQNUM\_USB\_WAKE\_UP** 42
- #define **NVIC\_IRQNUM\_TIM8\_BRK** 43
- #define **NVIC\_IRQNUM\_TIM8\_UP** 44
- #define **NVIC\_IRQNUM\_TIM8\_TRG\_COM** 45
- #define **NVIC\_IRQNUM\_TIM8\_CC** 46
- #define **NVIC\_IRQNUM\_ADC3** 47
- #define **NVIC\_IRQNUM\_FSMC** 48
- #define **NVIC\_IRQNUM\_SDIO** 49
- #define **NVIC\_IRQNUM\_TIM5** 50
- #define **NVIC\_IRQNUM\_SPI3** 51
- #define **NVIC\_IRQNUM\_UART4** 52
- #define **NVIC\_IRQNUM\_UART5** 53
- #define **NVIC\_IRQNUM\_TIM6** 54
- #define **NVIC\_IRQNUM\_TIM7** 55
- #define **NVIC\_IRQNUM\_DMA2\_Channel1** 56
- #define **NVIC\_IRQNUM\_DMA2\_Channel2** 57
- #define **NVIC\_IRQNUM\_DMA2\_Channel3** 58
- #define **NVIC\_IRQNUM\_DMA2\_Channel4\_5** 59

## Functions

- Std\_ReturnType [Nvic\\_EnableInterrupt](#) (uint8\_t intNumber)  
*Enables a specific Interrupt.*
- Std\_ReturnType [Nvic\\_DisableInterrupt](#) (uint8\_t intNumber)  
*Disables a specific Interrupt.*
- Std\_ReturnType [Nvic\\_SetPending](#) (uint8\_t intNumber)  
*Sets the pending flag for a specific interrupt.*

- Std\_ReturnType [Nvic\\_ClearPending](#) (uint8\_t intNumber)  
*Clears the pending flag for a specific interrupt.*
- Std\_ReturnType [Nvic\\_IsInterruptActive](#) (uint8\_t \*activeState, uint8\_t intNumber)  
*Checks if the interrupt is active.*
- Std\_ReturnType [Nvic\\_SetSubpriority](#) (uint8\_t priority, uint8\_t intNumber)  
*Sets the subpriority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_GetSubpriority](#) (uint8\_t \*priority, uint8\_t intNumber)  
*Gets the subpriority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_SetGroupPriority](#) (uint8\_t priority, uint8\_t intNumber)  
*Sets the group priority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_GetGroupPriority](#) (uint8\_t \*priority, uint8\_t intNumber)  
*Gets the group priority for aspecific interrupt.*
- Std\_ReturnType [Nvic\\_ConfigGroupSize](#) (void)  
*Configure the group size.*
- Std\_ReturnType [Nvic\\_ResetSystem](#) (void)  
*Resets the microcontroller.*
- Std\_ReturnType [Nvic\\_EnablePeripheral](#) (void)  
*Enables the peripherals interrupts.*
- Std\_ReturnType [Nvic\\_DisablePeripheral](#) (void)  
*Disable the peripherals interrupts.*
- Std\_ReturnType [Nvic\\_SetFault](#) (void)  
*Blocks all interrupts including hard fault.*
- Std\_ReturnType [Nvic\\_ClearFault](#) (void)  
*Returns from fault mode.*
- Std\_ReturnType [Nvic\\_FilterPriority](#) (uint8\_t pri)  
*Only allow interrupts over a certain priority.*
- Std\_ReturnType [Nvic\\_GenerateSoftwareInterrupt](#) (uint8\_t intNumber)  
*Generates a software interrupt (Atomic function to generate interrupt immediately)*

### 4.7.1 Detailed Description

This file is a user interface for the NVIC driver for the arm cortex m3.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com))

#### Version

0.1

#### Date

2020-02-29

#### Copyright

Copyright (c) 2020

## 4.7.2 Macro Definition Documentation

### 4.7.2.1 NVIC\_IRQNUM\_WWDG

```
#define NVIC_IRQNUM_WWDG 0
```

Interrupt Number

## 4.7.3 Function Documentation

### 4.7.3.1 Nvic\_ClearFault()

```
Std_ReturnType Nvic_ClearFault (
 void)
```

Returns from fault mode.

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

### 4.7.3.2 Nvic\_ClearPending()

```
Std_ReturnType Nvic_ClearPending (
 uint8_t intNumber)
```

Clears the pending flag for a specific interrupt.

#### Parameters

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table<br>• NVIC_IRQNUM_X |
|------------------|--------------------------------------------------------------------|

#### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.3 Nvic\_ConfigGroupSize()

```
Std_ReturnType Nvic_ConfigGroupSize (
 void)
```

Configure the group size.

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.4 Nvic\_DisableInterrupt()

```
Std_ReturnType Nvic_DisableInterrupt (
 uint8_t intNumber)
```

Disables a specific Interrupt.

##### Parameters

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table<br>• NVIC_IRQNUM_X |
|------------------|--------------------------------------------------------------------|

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.5 Nvic\_DisablePeripheral()

```
Std_ReturnType Nvic_DisablePeripheral (
 void)
```

Disable the peripherals interrupts.

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.6 Nvic\_EnableInterrupt()

```
Std_ReturnType Nvic_EnableInterrupt (
 uint8_t intNumber)
```

Enables a specific Interrupt.



## Parameters

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"> <li>• NVIC_IRQNUM_X</li> </ul> |
|------------------|-------------------------------------------------------------------------------------------------------------------|

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.7 Nvic\_EnablePeripheral()**

```
Std_ReturnType Nvic_EnablePeripheral (
 void)
```

Enables the peripherals interrupts.

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.8 Nvic\_FilterPriority()**

```
Std_ReturnType Nvic_FilterPriority (
 uint8_t pri)
```

Only allow interrupts over a certain priority.

## Parameters

|            |                              |
|------------|------------------------------|
| <i>per</i> | the minimum priority allowed |
|------------|------------------------------|

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.9 Nvic\_GenerateSoftwareInterrupt()**

```
Std_ReturnType Nvic_GenerateSoftwareInterrupt (
 uint8_t intNumber)
```

Generates a software interrupt (Atomic function to generate interrupt immediately)

**Parameters**

|                  |                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table<br><ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|--------------------------------------------------------------------------------------------------------------------|

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.10 Nvic\_GetGroupPriority()**

```
Std_ReturnType Nvic_GetGroupPriority (
 uint8_t * priority,
 uint8_t intNumber)
```

Gets the group priority for aspecific interrupt.

**Parameters**

|                  |                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to get                                                                                       |
| <i>intNumber</i> | the number of the interrupt in the vector table<br><ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.11 Nvic\_GetSubpriority()**

```
Std_ReturnType Nvic_GetSubpriority (
 uint8_t * priority,
 uint8_t intNumber)
```

Gets the subpriority for aspecific interrupt.

**Parameters**

|                  |                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to get                                                                                       |
| <i>intNumber</i> | the number of the interrupt in the vector table<br><ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.12 Nvic\_IsInterruptActive()**

```
Std_ReturnType Nvic_IsInterruptActive (
 uint8_t * activeState,
 uint8_t intNumber)
```

Checks if the interrupt is active.

**Parameters**

|                    |                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>activeState</i> | the active state of the interrupt <ul style="list-style-type: none"><li>• NVIC_ACTIVE</li><li>• NVIC_NOT_ACTIVE</li></ul> |
| <i>intNumber</i>   | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul>           |

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.13 Nvic\_ResetSystem()**

```
Std_ReturnType Nvic_ResetSystem (
 void)
```

Resets the microcontroller.

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

**4.7.3.14 Nvic\_SetFault()**

```
Std_ReturnType Nvic_SetFault (
 void)
```

Blocks all interrupts including hard fault.

**Returns**

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.15 Nvic\_SetGroupPriority()

```
Std_ReturnType Nvic_SetGroupPriority (
 uint8_t priority,
 uint8_t intNumber)
```

Sets the group priority for aspecific interrupt.

##### Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to set                                                                                    |
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.16 Nvic\_SetPending()

```
Std_ReturnType Nvic_SetPending (
 uint8_t intNumber)
```

Sets the pending flag for a specific interrupt.

##### Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |
|------------------|-----------------------------------------------------------------------------------------------------------------|

##### Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

#### 4.7.3.17 Nvic\_SetSubpriority()

```
Std_ReturnType Nvic_SetSubpriority (
 uint8_t priority,
 uint8_t intNumber)
```

Sets the subpriority for aspecific interrupt.

## Parameters

|                  |                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>priority</i>  | the priority you want to set                                                                                    |
| <i>intNumber</i> | the number of the interrupt in the vector table <ul style="list-style-type: none"><li>• NVIC_IRQNUM_X</li></ul> |

## Returns

Std\_ReturnType E\_OK: If the function executed successfully E\_NOT\_OK: If the function failed to execute

## 4.8 Protocol.c File Reference

This is the implementation for a self designed software protocol.

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

### Data Structures

- struct [header\\_t](#)
- struct [packet\\_t](#)

### Macros

- #define **PROTOCOL\_HEADER\_SIZE** 8
- #define **PROTOCOL\_ACK** 0x55
- #define **PROTOCOL\_ACK\_SIZE** 1
- #define **PROTOCOL\_DATA\_KEY** 2
- #define **PROTOCOL\_ADDRESS\_KEY** 4
- #define **PROTOCOL\_EOT\_KEY** 6

### Functions

- Std\_ReturnType [Protocol\\_Send](#) (uint16\_t msgType, uint16\_t length, uint8\_t \*data)  
*Sends a packet.*
- Std\_ReturnType [Protocol\\_Receive](#) (uint16\_t \*msgType, uint16\_t \*length, uint8\_t \*data)  
*Sends a packet.*
- Std\_ReturnType [Protocol\\_SendAck](#) (void)  
*Sends Acknowledgement.*

### 4.8.1 Detailed Description

This is the implementation for a self designed software protocol.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com) )

#### Version

0.1

#### Date

2020-05-15

#### Copyright

Copyright (c) 2020

### 4.8.2 Function Documentation

#### 4.8.2.1 Protocol\_Receive()

```
Std_ReturnType Protocol_Receive (
 uint16_t * msgType,
 uint16_t * length,
 uint8_t * data)
```

Sends a packet.

#### Parameters

|                |                                         |
|----------------|-----------------------------------------|
| <i>msgType</i> | The type of the message<br>• PROTOCOL_x |
| <i>length</i>  | the length of the data in the message   |
| <i>data</i>    | the data to send                        |

#### Returns

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

#### 4.8.2.2 Protocol\_Send()

```
Std_ReturnType Protocol_Send (
 uint16_t msgType,
 uint16_t length,
 uint8_t * data)
```

Sends a packet.

##### Parameters

|                |                                             |
|----------------|---------------------------------------------|
| <i>msgType</i> | The type of the message<br><br>• PROTOCOL_x |
| <i>length</i>  | the length of the data in the message       |
| <i>data</i>    | the data to send                            |

##### Returns

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

#### 4.8.2.3 Protocol\_SendAck()

```
Std_ReturnType Protocol_SendAck (
 void)
```

Sends Acknowledgement.

##### Returns

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

## 4.9 Protocol.h File Reference

This is the user interface for the self designed software protocol.

### Macros

- #define **PROTOCOL\_DATA** 1
- #define **PROTOCOL\_ADDRESS** 3
- #define **PROTOCOL\_EOT** 5

## Functions

- Std\_ReturnType [Protocol\\_Send](#) (uint16\_t msgType, uint16\_t length, uint8\_t \*data)  
*Sends a packet.*
- Std\_ReturnType [Protocol\\_Receive](#) (uint16\_t \*msgType, uint16\_t \*length, uint8\_t \*data)  
*Sends a packet.*
- Std\_ReturnType [Protocol\\_SendAck](#) (void)  
*Sends Acknowledgement.*

### 4.9.1 Detailed Description

This is the user interface for the self designed software protocol.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com) )

#### Version

0.1

#### Date

2020-05-16

#### Copyright

Copyright (c) 2020

### 4.9.2 Function Documentation

#### 4.9.2.1 Protocol\_Receive()

```
Std_ReturnType Protocol_Receive (
 uint16_t * msgType,
 uint16_t * length,
 uint8_t * data)
```

Sends a packet.

#### Parameters

|                |                                                                                         |
|----------------|-----------------------------------------------------------------------------------------|
| <i>msgType</i> | The type of the message<br><ul style="list-style-type: none"><li>• PROTOCOL_x</li></ul> |
| <i>length</i>  | the length of the data in the message                                                   |
| <i>data</i>    | the data to send                                                                        |



**Returns**

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

**4.9.2.2 Protocol\_Send()**

```
Std_ReturnType Protocol_Send (
 uint16_t msgType,
 uint16_t length,
 uint8_t * data)
```

Sends a packet.

**Parameters**

|                |                                         |
|----------------|-----------------------------------------|
| <i>msgType</i> | The type of the message<br>• PROTOCOL_x |
| <i>length</i>  | the length of the data in the message   |
| <i>data</i>    | the data to send                        |

**Returns**

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

**4.9.2.3 Protocol\_SendAck()**

```
Std_ReturnType Protocol_SendAck (
 void)
```

Sends Acknowledgement.

**Returns**

Std\_ReturnType A Status E\_OK : If the function was executed successfully E\_NOT\_OK : If the function didn't execute successfully

**4.10 Rcc.c File Reference**

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

```
#include "Std_Types.h"
#include "RCC.h"
```

## Macros

- `#define RCC_BASE_ADDRESS 0x40021000`
- `#define RCC_CR *((volatile u32*)(RCC_BASE_ADDRESS + 0x00))`
- `#define RCC_CFGR *((volatile u32*)(RCC_BASE_ADDRESS + 0x04))`
- `#define RCC_CIR *((volatile u32*)(RCC_BASE_ADDRESS + 0x08))`
- `#define RCC_APB2RSTR *((volatile u32*)(RCC_BASE_ADDRESS + 0x0C))`
- `#define RCC_APB1RSTR *((volatile u32*)(RCC_BASE_ADDRESS + 0x10))`
- `#define RCC_AHBENR *((volatile u32*)(RCC_BASE_ADDRESS + 0x14))`
- `#define RCC_APB2ENR *((volatile u32*)(RCC_BASE_ADDRESS + 0x18))`
- `#define RCC_APB1ENR *((volatile u32*)(RCC_BASE_ADDRESS + 0x1C))`
- `#define RCC_BDCR *((volatile u32*)(RCC_BASE_ADDRESS + 0x20))`
- `#define RCC_CRS *((volatile u32*)(RCC_BASE_ADDRESS + 0x24))`
- `#define RCC_AHBRSTR *((volatile u32*)(RCC_BASE_ADDRESS + 0x28))`
- `#define RCC_CFGR2 *((volatile u32*)(RCC_BASE_ADDRESS + 0x2C))`
- `#define RCC_MCO_CLR 0xF8FFFFFF`
- `#define RCC_SYS_CLK_SELECT_CLR 0xFFFFFFF0`
- `#define RCC_PLL_MUL_CLR 0xFFC3FFFF`
- `#define RCC_PLL_SRC_CLR 0xFFFEFFFF`
- `#define RCC_SYS_CLK_STATUS 0x0000000C`

## Functions

- Std\_ReturnType [Rcc\\_SetClockState](#) (uint32\_t clock, uint8\_t state)  
*Choose a specific clock and changes its state (On / Off)*
- Std\_ReturnType [Rcc\\_IsClockReady](#) (uint32\_t clock, uint8\_t \*ready)  
*Checks if a specific clock is ready or not.*
- Std\_ReturnType [Rcc\\_SelectMcoClock](#) (uint32\_t clock)  
*Selects the clock on the mco pin.*
- Std\_ReturnType [Rcc\\_SetPrescaler](#) (uint32\_t clock, uint32\_t value)  
*Sets the prescaler value for a specific clock.*
- Std\_ReturnType [Rcc\\_SetPllMultiplier](#) (uint32\_t pll)  
*Sets the PLL Multiplication factor.*
- Std\_ReturnType [Rcc\\_SetPllSource](#) (uint32\_t source)  
*Chooses the PLL clock source.*
- Std\_ReturnType [Rcc\\_GetSystemClockStatus](#) (uint8\_t \*sysClk)  
*Which clock is working as system clock.*
- Std\_ReturnType [Rcc\\_SwitchSystemClock](#) (uint32\_t clock)  
*Switches the system clock (HSI / HSE / PLL)*
- Std\_ReturnType [Rcc\\_SetApb2PeriphClockState](#) (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the APB2 bus and changes its state (On / Off)*
- Std\_ReturnType [Rcc\\_ResetApb2Periph](#) (uint32\_t periph)  
*Resets a specific peripheral on the APB2 bus.*
- Std\_ReturnType [Rcc\\_SetApb1PeriphClockState](#) (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the APB1 bus and changes its state (On / Off)*
- Std\_ReturnType [Rcc\\_ResetApb1Periph](#) (uint32\_t periph)  
*Resets a specific peripheral on the APB1 bus.*
- Std\_ReturnType [Rcc\\_SetAhbPeriphClockState](#) (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the AHB bus and changes its state (On / Off)*
- Std\_ReturnType [Rcc\\_ResetAhbPeriph](#) (uint32\_t periph)  
*Resets a specific peripheral on the AHB bus.*

### 4.10.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

### 4.10.2 Function Documentation

#### 4.10.2.1 Rcc\_GetSystemClockStatus()

```
Std_ReturnType Rcc_GetSystemClockStatus (
 uint8_t * sysClk)
```

Which clock is working as system clock.

#### 4.10.2.2 Function: Rcc\_GetSystemClockStatus

##### Parameters

|                |                                                                                                                                                                                                                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>sysClk</i>  | Saves the clock that is working as system clock in <ul style="list-style-type: none"><li>RCC_HSI_SYS : High speed internal clock is used as system clock</li><li>RCC_HSE_SYS : High speed external clock is used as system clock</li><li>RCC_PLL_SYS : PLL clock is used as system clock</li></ul> |
| <i>returns</i> | A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly                                                                                                                                                                                         |

#### 4.10.2.3 Rcc\_IsClockReady()

```
Std_ReturnType Rcc_IsClockReady (
 uint32_t clock,
 uint8_t * ready)
```

Checks if a specific clock is ready or not.

#### 4.10.2.4 Function: Rcc\_IsReady

## Parameters

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to check for <ul style="list-style-type: none"> <li>• RCC_HSI_RDY: for the high speed internal clock</li> <li>• RCC_HSE_RDY: for the high speed external clock</li> <li>• RCC_PLL_RDY: for the PLL clock</li> </ul> |
| <i>ready</i> | Saves the ready state in <ul style="list-style-type: none"> <li>• RCC_IS_RDY : if the clock is ready</li> <li>• RCC_NOT_RDY : if the clock is not ready</li> </ul>                                                                     |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.10.2.5 Rcc\_ResetAhbPeriph()

```
Std_ReturnType Rcc_ResetAhbPeriph (
 uint32_t periph)
```

Resets a specific peripheral on the AHB bus.

## 4.10.2.6 Function: Rcc\_ResetAhbPeriph

## Parameters

|               |                                                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | The peripheral you want to reset <ul style="list-style-type: none"> <li>• RCC_OTGFS_RST: OTGFS reset</li> <li>• RCC_ETHMAC_RST: Ethernet MAC reset</li> </ul> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 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 Rcc\_ResetApb1Periph()

```
Std_ReturnType Rcc_ResetApb1Periph (
 uint32_t periph)
```

Resets a specific peripheral on the APB1 bus.

#### 4.10.2.8 Function: Rcc\_ResetApb1Periph

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_RST: Timer 2 reset</li> <li>• RCC_TIM3_RST: Timer 3 reset</li> <li>• RCC_TIM4_RST: Timer 4 reset</li> <li>• RCC_TIM5_RST: Timer 5 reset</li> <li>• RCC_TIM6_RST: Timer 6 reset</li> <li>• RCC_TIM7_RST: Timer 7 reset</li> <li>• RCC_TIM12_RST: Timer 12 reset</li> <li>• RCC_TIM13_RST: Timer 13 reset</li> <li>• RCC_TIM14_RST: Timer 14 reset</li> <li>• RCC_WWD_GEN_RST: Window watchdog reset</li> <li>• RCC_SPI2_RST: SPI 2 reset</li> <li>• RCC_SPI3_RST: SPI 3 reset</li> <li>• RCC_USART2_RST: USART 2 reset</li> <li>• RCC_USART3_RST: USART 3 reset</li> <li>• RCC_USART4_RST: USART 4 reset</li> <li>• RCC_USART5_RST: USART 5 reset</li> <li>• RCC_I2C1_RST: I2C 1 reset</li> <li>• RCC_I2C2_RST: I2C 2 reset</li> <li>• RCC_USB_RST: USB reset</li> <li>• RCC_CAN_RST: CAN reset</li> <li>• RCC_BKP_RST: Backup interface reset</li> <li>• RCC_PWR_RST: Power interface reset</li> <li>• RCC_DAC_RST: DAC interface reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## 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 Rcc\_ResetApb2Periph()

```
Std_ReturnType Rcc_ResetApb2Periph (
 uint32_t periph)
```

Resets a specific peripheral on the APB2 bus.

#### 4.10.2.10 Function: Rcc\_ResetApb2Periph

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_RST: Alternate function input output reset</li> <li>• RCC_IOPA_RST: Port A input output reset</li> <li>• RCC_IOPB_RST: Port B input output reset</li> <li>• RCC_IOPC_RST: Port C input output reset</li> <li>• RCC_IOPD_RST: Port D input output reset</li> <li>• RCC_IOPE_RST: Port E input output reset</li> <li>• RCC_IOPF_RST: Port F input output reset</li> <li>• RCC_IOPG_RST: Port G input output reset</li> <li>• RCC_ADC1_RST: ADC 1 reset</li> <li>• RCC_ADC2_RST: ADC 2 reset</li> <li>• RCC_TIM1_RST: Timer 1 reset</li> <li>• RCC_SPI1_RST: SPI 1 reset</li> <li>• RCC_TIM8_RST: Timer 8 reset</li> <li>• RCC_USART1_RST: USART 1 reset</li> <li>• RCC_ADC3_RST: ADC 3 reset</li> <li>• RCC_TIM9_RST: Timer 9 reset</li> <li>• RCC_TIM10_RSTN: Timer 10 reset</li> <li>• RCC_TIM11_RSTN: Timer 11 reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.10.2.11 Rcc\_SelectMcoClock()

```
Std_ReturnType Rcc_SelectMcoClock (
 uint32_t clock)
```

Selects the clock on the mco pin.

## 4.10.2.12 Function: Rcc\_SelectMcoClock

## Parameters

|              |                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_MCO_NO_CLK : No clock will be on MCO</li> <li>• RCC_MCO_SYS_CLK : Select system clock on the MCO</li> <li>• RCC_MCO_HSI_CLK : Select high speed internal clock on the MCO</li> <li>• RCC_MCO_HSE_CLK : Select high speed external clock on the MCO</li> <li>• RCC_MCO_PLL_CLK : Select PLL clock on the MCO</li> </ul> |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.10.2.13 Rcc\_SetAhbPeriphClockState()

```
Std_ReturnType Rcc_SetAhbPeriphClockState (
 uint32_t periph,
 uint8_t state)
```

Choose a specific peripheral on the AHB bus and changes its state (On / Off)

## 4.10.2.14 Function: Rcc\_SetAhbPeriphClockState

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | The peripheral clock you want to configure <ul style="list-style-type: none"> <li>• RCC_DMA1_CLK_EN: DMA 1 clock enable</li> <li>• RCC_DMA2_CLK_EN: DMA 2 clock enable</li> <li>• RCC_SRAM_CLK_EN: SRAM interface clock enable</li> <li>• RCC_FLITF_CLK_EN: FLITF clock enable</li> <li>• RCC_CRC_CLK_EN: CRC clock enable</li> <li>• RCC_OTGFS_CLK_EN: OTGFS clock enable</li> <li>• RCC_ETHMAC_CLK_EN: Ethernet MAC clock enable</li> <li>• RCC_ETHMACTX_CLK_EN: Ethernet MAC TX clock enable</li> <li>• RCC_ETHMACRX_CLK_EN: Ethernet MAC RX clock enable</li> </ul> |
| <i>state</i>  | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON: To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                     |



**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.10.2.15 Rcc\_SetApb1PeriphClockState()**

```
Std_ReturnType Rcc_SetApb1PeriphClockState (
 uint32_t periph,
 uint8_t state)
```

Choose a specific peripheral on the APB1 bus and changes its state (On / Off)

**4.10.2.16 Function: Rcc\_SetApb1PeriphClockState**

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_CLK_EN: Timer 2 clock enable</li> <li>• RCC_TIM3_CLK_EN: Timer 3 clock enable</li> <li>• RCC_TIM4_CLK_EN: Timer 4 clock enable</li> <li>• RCC_TIM5_CLK_EN: Timer 5 clock enable</li> <li>• RCC_TIM6_CLK_EN: Timer 6 clock enable</li> <li>• RCC_TIM7_CLK_EN: Timer 7 clock enable</li> <li>• RCC_TIM12_CLK_EN: Timer 12 clock enable</li> <li>• RCC_TIM13_CLK_EN: Timer 13 clock enable</li> <li>• RCC_TIM14_CLK_EN: Timer 14 clock enable</li> <li>• RCC_WWD_GEN_CLK_EN: Window watchdog clock enable</li> <li>• RCC_SPI2_CLK_EN: SPI 2 clock enable</li> <li>• RCC_SPI3_CLK_EN: SPI 3 clock enable</li> <li>• RCC_USART2_CLK_EN: USART 2 clock enable</li> <li>• RCC_USART3_CLK_EN: USART 3 clock enable</li> <li>• RCC_USART4_CLK_EN: USART 4 clock enable</li> <li>• RCC_USART5_CLK_EN: USART 5 clock enable</li> <li>• RCC_I2C1_CLK_EN: I2C 1 clock enable</li> <li>• RCC_I2C2_CLK_EN: I2C 2 clock enable</li> <li>• RCC_USB_CLK_EN: USB clock enable</li> <li>• RCC_CAN_CLK_EN: CAN clock enable</li> <li>• RCC_BKP_CLK_EN: Backup interface clock enable</li> <li>• RCC_PWR_CLK_EN: Power interface clock enable</li> <li>• RCC_DAC_CLK_EN: DAC interface clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.10.2.17 Rcc\_SetApb2PeriphClockState()

```
Std_ReturnType Rcc_SetApb2PeriphClockState (
```

```
uint32_t periph,
uint8_t state)
```

Choose a specific peripheral on the APB2 bus and changes its state (On / Off)

#### 4.10.2.18 Function: Rcc\_SetApb2PeriphClockState

##### Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_CLK_EN: Alternate function input output clock enable</li> <li>• RCC_IOPA_CLK_EN: Port A input output clock enable</li> <li>• RCC_IOPB_CLK_EN: Port B input output clock enable</li> <li>• RCC_IOPC_CLK_EN: Port C input output clock enable</li> <li>• RCC_IOPD_CLK_EN: Port D input output clock enable</li> <li>• RCC_IOPE_CLK_EN: Port E input output clock enable</li> <li>• RCC_IOPF_CLK_EN: Port F input output clock enable</li> <li>• RCC_IOPG_CLK_EN: Port G input output clock enable</li> <li>• RCC_ADC1_CLK_EN: ADC 1 clock enable</li> <li>• RCC_ADC2_CLK_EN: ADC 2 clock enable</li> <li>• RCC_TIM1_CLK_EN: Timer 1 clock enable</li> <li>• RCC_SPI1_CLK_EN: SPI 1 clock enable</li> <li>• RCC_TIM8_CLK_EN: Timer 8 clock enable</li> <li>• RCC_USART1_CLK_EN: USART 1 clock enable</li> <li>• RCC_ADC3_CLK_EN: ADC 3 clock enable</li> <li>• RCC_TIM9_CLK_EN: Timer 9 clock enable</li> <li>• RCC_TIM10_CLK_EN: Timer 10 clock enable</li> <li>• RCC_TIM11_CLK_EN: Timer 11 clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

##### Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

#### 4.10.2.19 Rcc\_SetClockState()

```
Std_ReturnType Rcc_SetClockState (
 uint32_t clock,
 uint8_t state)
```

Choose a specific clock and changes its state (On / Off)

#### 4.10.2.20 Function: Rcc\_SetClockState

##### Parameters

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_HSI_SET: for the high speed internal clock</li> <li>• RCC_HSE_SET: for the high speed external clock</li> <li>• RCC_PLL_SET: for the PLL clock</li> </ul> |
| <i>state</i> | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_CLK_ON : To set the clock on</li> <li>• RCC_CLK_OFF : To set the clock off</li> </ul>                                                                 |

##### Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

#### 4.10.2.21 Rcc\_SetPllMultiplier()

```
Std_ReturnType Rcc_SetPllMultiplier (
 uint32_t pll)
```

Sets the PLL Multiplication factor.

#### 4.10.2.22 Function: Rcc\_SetPllMultiplier

##### Parameters

|            |                                                                                                                                    |
|------------|------------------------------------------------------------------------------------------------------------------------------------|
| <i>pll</i> | : The PLL multiplication factor <ul style="list-style-type: none"> <li>• RCC_PLL_MUL_XX : Set PLL multiplication factor</li> </ul> |
|------------|------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.10.2.23 Rcc\_SetPllSource()**

```
Std_ReturnType Rcc_SetPllSource (
 uint32_t source)
```

Chooses the PLL clock source.

**4.10.2.24 Function: Rcc\_SetPllSource****Parameters**

|               |                                                                                                                                                                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>source</i> | : The PLL clock source <ul style="list-style-type: none"> <li>• RCC_PLL_SRC_HSI : Choose high speed internal clock / 2 as a PLL clock source</li> <li>• RCC_PLL_SRC_HSE : Choose high speed external clock as a PLL clock source</li> </ul> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.10.2.25 Rcc\_SetPrescaler()**

```
Std_ReturnType Rcc_SetPrescaler (
 uint32_t clock,
 uint32_t value)
```

Sets the prescaler value for a specific clock.

**4.10.2.26 Function: Rcc\_SetPrescaler****Parameters**

|              |                                                                                                                                                                                                                                                                                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_USB_PRE : For the USB prescaler</li> <li>• RCC_PLL_HSE_PRE : For the Pll prescaler</li> <li>• RCC_ADC_PRE : For the ADC prescaler</li> <li>• RCC_APB2_PRE : For the APB2 prescaler</li> <li>• RCC_APB1_PRE : For the APB1 prescaler</li> <li>• RCC_AHB_PRE : For the AHB prescaler</li> </ul> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Parameters**

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>value</i> | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_USB_PRE_1_5 : No USB prescaler value</li> <li>• RCC_USB_PRE_1_5 : USB prescaler 1.5</li> <li>• RCC_PLL_HSE_PRE_X : PLL Prescaler value using high speed external clock</li> <li>• RCC_ADC_PRE_X : ADC Prescaler value</li> <li>• RCC_APB2_PRE_XX : APB2 prescaler value</li> <li>• RCC_APB1_PRE_XX : APB1 prescaler value</li> <li>• RCC_AHB_PRE_XXX : AHB prescaler value</li> </ul> |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.10.2.27 Rcc\_SwitchSystemClock()**

```
Std_ReturnType Rcc_SwitchSystemClock (
 uint32_t clock)
```

Switches the system clock (HSI / HSE / PLL)

**4.10.2.28 Function: Rcc\_SwitchSystemClock****Parameters**

|              |                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | : The PLL clock source <ul style="list-style-type: none"> <li>• RCC_SYS_CLK_SELECT_HSI : Select high speed internal clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_HSE : Select high speed external clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_PLL : Select PLL clock as system clock</li> </ul> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11 Rcc.h File Reference**

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

## Macros

- `#define RCC_YES 4`
- `#define RCC_NO 5`
- `#define RCC_CLK_ON 6`
- `#define RCC_CLK_OFF 7`
- `#define RCC_IS_RDY 0`
- `#define RCC_NOT_RDY !RCC_IS_RDY`
- `#define RCC_HSI_SET 0x00000001`
- `#define RCC_HSE_SET 0x00010000`
- `#define RCC_PLL_SET 0x01000000`
- `#define RCC_HSI_RDY 0xFFFFFFFF`
- `#define RCC_HSE_RDY 0xFFFDFFFF`
- `#define RCC_PLL_RDY 0xFDFFFFFF`
- `#define RCC_MCO_NO_CLK 0x00000000`
- `#define RCC_MCO_SYS_CLK 0x04000000`
- `#define RCC_MCO_HSI_CLK 0x05000000`
- `#define RCC_MCO_HSE_CLK 0x06000000`
- `#define RCC_MCO_PLL_CLK 0x07000000`
- `#define RCC_PLL_SRC_HSI 0x00000000`
- `#define RCC_PLL_SRC_HSE 0x00010000`
- `#define RCC_USB_PRE 0xFFBFFFFF`
- `#define RCC_PLL_HSE_PRE 0xFFFDFFFF`
- `#define RCC_ADC_PRE 0xFFFF3FFF`
- `#define RCC_APB2_PRE 0xFFFFC7FF`
- `#define RCC_APB1_PRE 0xFFFFF8FF`
- `#define RCC_AHB_PRE 0xFFFFF0F`
- `#define RCC_USB_PRE_1_5 0x00000000`
- `#define RCC_USB_PRE_0 0x00400000`
- `#define RCC_PLL_MUL_02 0x00000000`
- `#define RCC_PLL_MUL_03 0x00040000`
- `#define RCC_PLL_MUL_04 0x00080000`
- `#define RCC_PLL_MUL_05 0x000C0000`
- `#define RCC_PLL_MUL_06 0x00100000`
- `#define RCC_PLL_MUL_07 0x00140000`
- `#define RCC_PLL_MUL_08 0x00180000`
- `#define RCC_PLL_MUL_09 0x001C0000`
- `#define RCC_PLL_MUL_10 0x00200000`
- `#define RCC_PLL_MUL_11 0x00240000`
- `#define RCC_PLL_MUL_12 0x00280000`
- `#define RCC_PLL_MUL_13 0x002C0000`
- `#define RCC_PLL_MUL_14 0x00300000`
- `#define RCC_PLL_MUL_15 0x00340000`
- `#define RCC_PLL_MUL_16 0x00380000`
- `#define RCC_PLL_HSE_PRE_0 0x00000000`
- `#define RCC_PLL_HSE_PRE_2 0x00020000`
- `#define RCC_ADC_PRE_2 0x00000000`
- `#define RCC_ADC_PRE_4 0x00004000`
- `#define RCC_ADC_PRE_6 0x00008000`
- `#define RCC_ADC_PRE_8 0x0000C000`
- `#define RCC_APB2_PRE_00 0x00000000`
- `#define RCC_APB2_PRE_02 0x00002000`
- `#define RCC_APB2_PRE_04 0x00002800`
- `#define RCC_APB2_PRE_08 0x00003000`
- `#define RCC_APB2_PRE_16 0x00003800`

- #define **RCC\_APB1\_PRE\_00** 0x00000000
- #define **RCC\_APB1\_PRE\_02** 0x00000400
- #define **RCC\_APB1\_PRE\_04** 0x00000500
- #define **RCC\_APB1\_PRE\_08** 0x00000600
- #define **RCC\_APB1\_PRE\_16** 0x00000700
- #define **RCC\_AHB\_PRE\_000** 0x00000000
- #define **RCC\_AHB\_PRE\_002** 0x00000080
- #define **RCC\_AHB\_PRE\_004** 0x00000090
- #define **RCC\_AHB\_PRE\_008** 0x000000A0
- #define **RCC\_AHB\_PRE\_016** 0x000000B0
- #define **RCC\_AHB\_PRE\_064** 0x000000C0
- #define **RCC\_AHB\_PRE\_128** 0x000000D0
- #define **RCC\_AHB\_PRE\_256** 0x000000E0
- #define **RCC\_AHB\_PRE\_512** 0x000000F0
- #define **RCC\_HSI\_SYS** 0x00000000
- #define **RCC\_HSE\_SYS** 0x00000001
- #define **RCC\_PLL\_SYS** 0x00000002
- #define **RCC\_SYS\_CLK\_SELECT\_HSI** 0x00000000
- #define **RCC\_SYS\_CLK\_SELECT\_HSE** 0x00000001
- #define **RCC\_SYS\_CLK\_SELECT\_PLL** 0x00000002
- #define **RCC\_PERIPH\_CLK\_ON** 0
- #define **RCC\_PERIPH\_CLK\_OFF** 1
- #define **RCC\_AFIO\_CLK\_EN** 0x00000001
- #define **RCC\_IOPA\_CLK\_EN** 0x00000004
- #define **RCC\_IOPB\_CLK\_EN** 0x00000008
- #define **RCC\_IOPC\_CLK\_EN** 0x00000010
- #define **RCC\_IOPD\_CLK\_EN** 0x00000020
- #define **RCC\_IOPE\_CLK\_EN** 0x00000040
- #define **RCC\_IOPF\_CLK\_EN** 0x00000080
- #define **RCC\_IOPG\_CLK\_EN** 0x00000100
- #define **RCC\_ADC1\_CLK\_EN** 0x00000200
- #define **RCC\_ADC2\_CLK\_EN** 0x00000400
- #define **RCC\_TIM1\_CLK\_EN** 0x00000800
- #define **RCC\_SPI1\_CLK\_EN** 0x00001000
- #define **RCC\_TIM8\_CLK\_EN** 0x00002000
- #define **RCC\_USART1\_CLK\_EN** 0x00004000
- #define **RCC\_ADC3\_CLK\_EN** 0x00008000
- #define **RCC\_TIM9\_CLK\_EN** 0x00080000
- #define **RCC\_TIM10\_CLK\_EN** 0x00100000
- #define **RCC\_TIM11\_CLK\_EN** 0x00200000
- #define **RCC\_AFIO\_RST** 0x00000001
- #define **RCC\_IOPA\_RST** 0x00000004
- #define **RCC\_IOPB\_RST** 0x00000008
- #define **RCC\_IOPC\_RST** 0x00000010
- #define **RCC\_IOPD\_RST** 0x00000020
- #define **RCC\_IOPE\_RST** 0x00000040
- #define **RCC\_IOPF\_RST** 0x00000080
- #define **RCC\_IOPG\_RST** 0x00000100
- #define **RCC\_ADC1\_RST** 0x00000200
- #define **RCC\_ADC2\_RST** 0x00000400
- #define **RCC\_TIM1\_RST** 0x00000800
- #define **RCC\_SPI1\_RST** 0x00001000
- #define **RCC\_TIM8\_RST** 0x00002000
- #define **RCC\_USART1\_RST** 0x00004000
- #define **RCC\_ADC3\_RST** 0x00008000



- #define **RCC\_TIM9\_RST** 0x00080000
- #define **RCC\_TIM10\_RST** 0x00100000
- #define **RCC\_TIM11\_RST** 0x00200000
- #define **RCC\_TIM2\_CLK\_EN** 0x00000001
- #define **RCC\_TIM3\_CLK\_EN** 0x00000002
- #define **RCC\_TIM4\_CLK\_EN** 0x00000004
- #define **RCC\_TIM5\_CLK\_EN** 0x00000008
- #define **RCC\_TIM6\_CLK\_EN** 0x00000010
- #define **RCC\_TIM7\_CLK\_EN** 0x00000020
- #define **RCC\_TIM12\_CLK\_EN** 0x00000040
- #define **RCC\_TIM13\_CLK\_EN** 0x00000080
- #define **RCC\_TIM14\_CLK\_EN** 0x00000100
- #define **RCC\_WWD\_GEN\_CLK\_EN** 0x00000800
- #define **RCC\_SPI2\_CLK\_EN** 0x00004000
- #define **RCC\_SPI3\_CLK\_EN** 0x00008000
- #define **RCC\_USART2\_CLK\_EN** 0x00020000
- #define **RCC\_USART3\_CLK\_EN** 0x00040000
- #define **RCC\_USART4\_CLK\_EN** 0x00080000
- #define **RCC\_USART5\_CLK\_EN** 0x00100000
- #define **RCC\_I2C1\_CLK\_EN** 0x00200000
- #define **RCC\_I2C2\_CLK\_EN** 0x00400000
- #define **RCC\_USB\_CLK\_EN** 0x00800000
- #define **RCC\_CAN\_CLK\_EN** 0x02000000
- #define **RCC\_BKP\_CLK\_EN** 0x08000000
- #define **RCC\_PWR\_CLK\_EN** 0x10000000
- #define **RCC\_DAC\_CLK\_EN** 0x20000000
- #define **RCC\_TIM2\_RST** 0x00000001
- #define **RCC\_TIM3\_RST** 0x00000002
- #define **RCC\_TIM4\_RST** 0x00000004
- #define **RCC\_TIM5\_RST** 0x00000008
- #define **RCC\_TIM6\_RST** 0x00000010
- #define **RCC\_TIM7\_RST** 0x00000020
- #define **RCC\_TIM12\_RST** 0x00000040
- #define **RCC\_TIM13\_RST** 0x00000080
- #define **RCC\_TIM14\_RST** 0x00000100
- #define **RCC\_WWD\_GEN\_RST** 0x00000800
- #define **RCC\_SPI2\_RST** 0x00004000
- #define **RCC\_SPI3\_RST** 0x00008000
- #define **RCC\_USART2\_RST** 0x00020000
- #define **RCC\_USART3\_RST** 0x00040000
- #define **RCC\_USART4\_RST** 0x00080000
- #define **RCC\_USART5\_RST** 0x00100000
- #define **RCC\_I2C1\_RST** 0x00200000
- #define **RCC\_I2C2\_RST** 0x00400000
- #define **RCC\_USB\_RST** 0x00800000
- #define **RCC\_CAN\_RST** 0x02000000
- #define **RCC\_BKP\_RST** 0x08000000
- #define **RCC\_PWR\_RST** 0x10000000
- #define **RCC\_DAC\_RST** 0x20000000
- #define **RCC\_DMA1\_CLK\_EN** 0x00000001
- #define **RCC\_DMA2\_CLK\_EN** 0x00000002
- #define **RCC\_SRAM\_CLK\_EN** 0x00000004
- #define **RCC\_FLITF\_CLK\_EN** 0x00000010
- #define **RCC\_CRC\_CLK\_EN** 0x00000040
- #define **RCC\_OTGFS\_CLK\_EN** 0x00001000

- #define **RCC\_ETHMAC\_CLK\_EN** 0x00004000
- #define **RCC\_ETHMACTX\_CLK\_EN** 0x00008000
- #define **RCC\_ETHMACRX\_CLK\_EN** 0x00010000
- #define **RCC\_OTGFS\_RST** 0x00001000
- #define **RCC\_ETHMAC\_RST** 0x00004000

## Functions

- Std\_ReturnType **Rcc\_SetClockState** (uint32\_t clock, uint8\_t state)  
*Choose a specific clock and changes its state (On / Off)*
- Std\_ReturnType **Rcc\_IsClockReady** (uint32\_t clock, uint8\_t \*ready)  
*Checks if a specific clock is ready or not.*
- Std\_ReturnType **Rcc\_SelectMcoClock** (uint32\_t clock)  
*Selects the clock on the mco pin.*
- Std\_ReturnType **Rcc\_SetPrescaler** (uint32\_t clock, uint32\_t value)  
*Sets the prescaler value for a specific clock.*
- Std\_ReturnType **Rcc\_SetPllMultiplier** (uint32\_t pll)  
*Sets the PLL Multiplication factor.*
- Std\_ReturnType **Rcc\_SetPllSource** (uint32\_t source)  
*Chooses the PLL clock source.*
- Std\_ReturnType **Rcc\_GetSystemClockStatus** (uint8\_t \*sysClk)  
*Which clock is working as system clock.*
- Std\_ReturnType **Rcc\_SwitchSystemClock** (uint32\_t clock)  
*Switches the system clock (HSI / HSE / PLL)*
- Std\_ReturnType **Rcc\_SetApb2PeriphClockState** (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the APB2 bus and changes its state (On / Off)*
- Std\_ReturnType **Rcc\_ResetApb2Periph** (uint32\_t periph)  
*Resets a specific peripheral on the APB2 bus.*
- Std\_ReturnType **Rcc\_SetApb1PeriphClockState** (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the APB1 bus and changes its state (On / Off)*
- Std\_ReturnType **Rcc\_ResetApb1Periph** (uint32\_t periph)  
*Resets a specific peripheral on the APB1 bus.*
- Std\_ReturnType **Rcc\_SetAhbPeriphClockState** (uint32\_t periph, uint8\_t state)  
*Choose a specific peripheral on the AHB bus and changes its state (On / Off)*
- Std\_ReturnType **Rcc\_ResetAhbPeriph** (uint32\_t periph)  
*Resets a specific peripheral on the AHB bus.*

### 4.11.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

## 4.11.2 Function Documentation

### 4.11.2.1 Rcc\_GetSystemClockStatus()

```
Std_ReturnType Rcc_GetSystemClockStatus (
 uint8_t * sysClk)
```

Which clock is working as system clock.

#### 4.11.2.2 Function: Rcc\_GetSystemClockStatus

##### Parameters

|                |                                                                                                                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>sysClk</i>  | Saves the clock that is working as system clock in <ul style="list-style-type: none"> <li>• RCC_HSI_SYS : High speed internal clock is used as system clock</li> <li>• RCC_HSE_SYS : High speed external clock is used as system clock</li> <li>• RCC_PLL_SYS : PLL clock is used as system clock</li> </ul> |
| <i>returns</i> | A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly                                                                                                                                                                                                   |

#### 4.11.2.3 Function: Rcc\_GetSystemClockStatus

##### Parameters

|                |                                                                                                                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>sysClk</i>  | Saves the clock that is working as system clock in <ul style="list-style-type: none"> <li>• RCC_HSI_SYS : High speed internal clock is used as system clock</li> <li>• RCC_HSE_SYS : High speed external clock is used as system clock</li> <li>• RCC_PLL_SYS : PLL clock is used as system clock</li> </ul> |
| <i>returns</i> | A status E_OK : if the function is executed correctly E_NOT_OK : if the function is not executed correctly                                                                                                                                                                                                   |

### 4.11.2.4 Rcc\_IsClockReady()

```
Std_ReturnType Rcc_IsClockReady (
 uint32_t clock,
 uint8_t * ready)
```

Checks if a specific clock is ready or not.

#### 4.11.2.5 Function: Rcc\_IsReady

**Parameters**

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to check for <ul style="list-style-type: none"> <li>• RCC_HSI_RDY: for the high speed internal clock</li> <li>• RCC_HSE_RDY: for the high speed external clock</li> <li>• RCC_PLL_RDY: for the PLL clock</li> </ul> |
| <i>ready</i> | Saves the ready state in <ul style="list-style-type: none"> <li>• RCC_IS_RDY : if the clock is ready</li> <li>• RCC_NOT_RDY : if the clock is not ready</li> </ul>                                                                     |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.6 Function: Rcc\_IsReady****Parameters**

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to check for <ul style="list-style-type: none"> <li>• RCC_HSI_RDY: for the high speed internal clock</li> <li>• RCC_HSE_RDY: for the high speed external clock</li> <li>• RCC_PLL_RDY: for the PLL clock</li> </ul> |
| <i>ready</i> | Saves the ready state in <ul style="list-style-type: none"> <li>• RCC_IS_RDY : if the clock is ready</li> <li>• RCC_NOT_RDY : if the clock is not ready</li> </ul>                                                                     |

**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 Rcc\_ResetAhbPeriph()**

```
Std_ReturnType Rcc_ResetAhbPeriph (
 uint32_t periph)
```

Resets a specific peripheral on the AHB bus.

**4.11.2.8 Function: Rcc\_ResetAhbPeriph**

## Parameters

|               |                                                                                                                                                            |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | The peripheral you want to reset <ul style="list-style-type: none"><li>• RCC_OTGFS_RST: OTGFS reset</li><li>• RCC_ETHMAC_RST: Ethernet MAC reset</li></ul> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.9 Function: Rcc\_ResetAhbPeriph**

## Parameters

|               |                                                                                                                                                            |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | The peripheral you want to reset <ul style="list-style-type: none"><li>• RCC_OTGFS_RST: OTGFS reset</li><li>• RCC_ETHMAC_RST: Ethernet MAC reset</li></ul> |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.10 Rcc\_ResetApb1Periph()**

```
Std_ReturnType Rcc_ResetApb1Periph (
 uint32_t periph)
```

Resets a specific peripheral on the APB1 bus.

**4.11.2.11 Function: Rcc\_ResetApb1Periph**

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_RST: Timer 2 reset</li> <li>• RCC_TIM3_RST: Timer 3 reset</li> <li>• RCC_TIM4_RST: Timer 4 reset</li> <li>• RCC_TIM5_RST: Timer 5 reset</li> <li>• RCC_TIM6_RST: Timer 6 reset</li> <li>• RCC_TIM7_RST: Timer 7 reset</li> <li>• RCC_TIM12_RST: Timer 12 reset</li> <li>• RCC_TIM13_RST: Timer 13 reset</li> <li>• RCC_TIM14_RST: Timer 14 reset</li> <li>• RCC_WWD_GEN_RST: Window watchdog reset</li> <li>• RCC_SPI2_RST: SPI 2 reset</li> <li>• RCC_SPI3_RST: SPI 3 reset</li> <li>• RCC_USART2_RST: USART 2 reset</li> <li>• RCC_USART3_RST: USART 3 reset</li> <li>• RCC_USART4_RST: USART 4 reset</li> <li>• RCC_USART5_RST: USART 5 reset</li> <li>• RCC_I2C1_RST: I2C 1 reset</li> <li>• RCC_I2C2_RST: I2C 2 reset</li> <li>• RCC_USB_RST: USB reset</li> <li>• RCC_CAN_RST: CAN reset</li> <li>• RCC_BKP_RST: Backup interface reset</li> <li>• RCC_PWR_RST: Power interface reset</li> <li>• RCC_DAC_RST: DAC interface reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.12 Function: Rcc\_ResetApb1Periph

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_RST: Timer 2 reset</li> <li>• RCC_TIM3_RST: Timer 3 reset</li> <li>• RCC_TIM4_RST: Timer 4 reset</li> <li>• RCC_TIM5_RST: Timer 5 reset</li> <li>• RCC_TIM6_RST: Timer 6 reset</li> <li>• RCC_TIM7_RST: Timer 7 reset</li> <li>• RCC_TIM12_RST: Timer 12 reset</li> <li>• RCC_TIM13_RST: Timer 13 reset</li> <li>• RCC_TIM14_RST: Timer 14 reset</li> <li>• RCC_WWD_GEN_RST: Window watchdog reset</li> <li>• RCC_SPI2_RST: SPI 2 reset</li> <li>• RCC_SPI3_RST: SPI 3 reset</li> <li>• RCC_USART2_RST: USART 2 reset</li> <li>• RCC_USART3_RST: USART 3 reset</li> <li>• RCC_USART4_RST: USART 4 reset</li> <li>• RCC_USART5_RST: USART 5 reset</li> <li>• RCC_I2C1_RST: I2C 1 reset</li> <li>• RCC_I2C2_RST: I2C 2 reset</li> <li>• RCC_USB_RST: USB reset</li> <li>• RCC_CAN_RST: CAN reset</li> <li>• RCC_BKP_RST: Backup interface reset</li> <li>• RCC_PWR_RST: Power interface reset</li> <li>• RCC_DAC_RST: DAC interface reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.13 Rcc\_ResetApb2Periph()

```
Std_ReturnType Rcc_ResetApb2Periph (
 uint32_t periph)
```

Resets a specific peripheral on the APB2 bus.

## 4.11.2.14 Function: Rcc\_ResetApb2Periph

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_RST: Alternate function input output reset</li> <li>• RCC_IOPA_RST: Port A input output reset</li> <li>• RCC_IOPB_RST: Port B input output reset</li> <li>• RCC_IOPC_RST: Port C input output reset</li> <li>• RCC_IOPD_RST: Port D input output reset</li> <li>• RCC_IOPE_RST: Port E input output reset</li> <li>• RCC_IOPF_RST: Port F input output reset</li> <li>• RCC_IOPG_RST: Port G input output reset</li> <li>• RCC_ADC1_RST: ADC 1 reset</li> <li>• RCC_ADC2_RST: ADC 2 reset</li> <li>• RCC_TIM1_RST: Timer 1 reset</li> <li>• RCC_SPI1_RST: SPI 1 reset</li> <li>• RCC_TIM8_RST: Timer 8 reset</li> <li>• RCC_USART1_RST: USART 1 reset</li> <li>• RCC_ADC3_RST: ADC 3 reset</li> <li>• RCC_TIM9_RST: Timer 9 reset</li> <li>• RCC_TIM10_RSTN: Timer 10 reset</li> <li>• RCC_TIM11_RSTN: Timer 11 reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.15 Function: Rcc\_ResetApb2Periph



## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral you want to reset</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_RST: Alternate function input output reset</li> <li>• RCC_IOPA_RST: Port A input output reset</li> <li>• RCC_IOPB_RST: Port B input output reset</li> <li>• RCC_IOPC_RST: Port C input output reset</li> <li>• RCC_IOPD_RST: Port D input output reset</li> <li>• RCC_IOPE_RST: Port E input output reset</li> <li>• RCC_IOPF_RST: Port F input output reset</li> <li>• RCC_IOPG_RST: Port G input output reset</li> <li>• RCC_ADC1_RST: ADC 1 reset</li> <li>• RCC_ADC2_RST: ADC 2 reset</li> <li>• RCC_TIM1_RST: Timer 1 reset</li> <li>• RCC_SPI1_RST: SPI 1 reset</li> <li>• RCC_TIM8_RST: Timer 8 reset</li> <li>• RCC_USART1_RST: USART 1 reset</li> <li>• RCC_ADC3_RST: ADC 3 reset</li> <li>• RCC_TIM9_RST: Timer 9 reset</li> <li>• RCC_TIM10_RSTN: Timer 10 reset</li> <li>• RCC_TIM11_RSTN: Timer 11 reset</li> </ul> |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.16 Rcc\_SelectMcoClock()

```
Std_ReturnType Rcc_SelectMcoClock (
 uint32_t clock)
```

Selects the clock on the mco pin.

## 4.11.2.17 Function: Rcc\_SelectMcoClock

**Parameters**

|              |                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_MCO_NO_CLK : No clock will be on MCO</li> <li>• RCC_MCO_SYS_CLK : Select system clock on the MCO</li> <li>• RCC_MCO_HSI_CLK : Select high speed internal clock on the MCO</li> <li>• RCC_MCO_HSE_CLK : Select high speed external clock on the MCO</li> <li>• RCC_MCO_PLL_CLK : Select PLL clock on the MCO</li> </ul> |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.18 Function: Rcc\_SelectMcoClock****Parameters**

|              |                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_MCO_NO_CLK : No clock will be on MCO</li> <li>• RCC_MCO_SYS_CLK : Select system clock on the MCO</li> <li>• RCC_MCO_HSI_CLK : Select high speed internal clock on the MCO</li> <li>• RCC_MCO_HSE_CLK : Select high speed external clock on the MCO</li> <li>• RCC_MCO_PLL_CLK : Select PLL clock on the MCO</li> </ul> |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.19 Rcc\_SetAhbPeriphClockState()**

```
Std_ReturnType Rcc_SetAhbPeriphClockState (
 uint32_t periph,
 uint8_t state)
```

Choose a specific peripheral on the AHB bus and changes its state (On / Off)

**4.11.2.20 Function: Rcc\_SetAhbPeriphClockState**

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_DMA1_CLK_EN: DMA 1 clock enable</li> <li>• RCC_DMA2_CLK_EN: DMA 2 clock enable</li> <li>• RCC_SRAM_CLK_EN: SRAM interface clock enable</li> <li>• RCC_FLITF_CLK_EN: FLITF clock enable</li> <li>• RCC_CRC_CLK_EN: CRC clock enable</li> <li>• RCC_OTGFS_CLK_EN: OTGFS clock enable</li> <li>• RCC_ETHMAC_CLK_EN: Ethernet MAC clock enable</li> <li>• RCC_ETHMACTX_CLK_EN: Ethernet MAC TX clock enable</li> <li>• RCC_ETHMACRX_CLK_EN: Ethernet MAC RX clock enable</li> </ul> |
| <i>state</i>  | <p>: The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON: To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                     |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.21 Function: Rcc\_SetAhbPeriphClockState

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_DMA1_CLK_EN: DMA 1 clock enable</li> <li>• RCC_DMA2_CLK_EN: DMA 2 clock enable</li> <li>• RCC_SRAM_CLK_EN: SRAM interface clock enable</li> <li>• RCC_FLITF_CLK_EN: FLITF clock enable</li> <li>• RCC_CRC_CLK_EN: CRC clock enable</li> <li>• RCC_OTGFS_CLK_EN: OTGFS clock enable</li> <li>• RCC_ETHMAC_CLK_EN: Ethernet MAC clock enable</li> <li>• RCC_ETHMACTX_CLK_EN: Ethernet MAC TX clock enable</li> <li>• RCC_ETHMACRX_CLK_EN: Ethernet MAC RX clock enable</li> </ul> |
| <i>state</i>  | <p>: The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON: To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                     |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.22 Rcc\_SetApb1PeriphClockState()**

```
Std_ReturnType Rcc_SetApb1PeriphClockState (
 uint32_t periph,
 uint8_t state)
```

Choose a specific peripheral on the APB1 bus and changes its state (On / Off)

**4.11.2.23 Function: Rcc\_SetApb1PeriphClockState**

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_CLK_EN: Timer 2 clock enable</li> <li>• RCC_TIM3_CLK_EN: Timer 3 clock enable</li> <li>• RCC_TIM4_CLK_EN: Timer 4 clock enable</li> <li>• RCC_TIM5_CLK_EN: Timer 5 clock enable</li> <li>• RCC_TIM6_CLK_EN: Timer 6 clock enable</li> <li>• RCC_TIM7_CLK_EN: Timer 7 clock enable</li> <li>• RCC_TIM12_CLK_EN: Timer 12 clock enable</li> <li>• RCC_TIM13_CLK_EN: Timer 13 clock enable</li> <li>• RCC_TIM14_CLK_EN: Timer 14 clock enable</li> <li>• RCC_WWD_GEN_CLK_EN: Window watchdog clock enable</li> <li>• RCC_SPI2_CLK_EN: SPI 2 clock enable</li> <li>• RCC_SPI3_CLK_EN: SPI 3 clock enable</li> <li>• RCC_USART2_CLK_EN: USART 2 clock enable</li> <li>• RCC_USART3_CLK_EN: USART 3 clock enable</li> <li>• RCC_USART4_CLK_EN: USART 4 clock enable</li> <li>• RCC_USART5_CLK_EN: USART 5 clock enable</li> <li>• RCC_I2C1_CLK_EN: I2C 1 clock enable</li> <li>• RCC_I2C2_CLK_EN: I2C 2 clock enable</li> <li>• RCC_USB_CLK_EN: USB clock enable</li> <li>• RCC_CAN_CLK_EN: CAN clock enable</li> <li>• RCC_BKP_CLK_EN: Backup interface clock enable</li> <li>• RCC_PWR_CLK_EN: Power interface clock enable</li> <li>• RCC_DAC_CLK_EN: DAC interface clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.24 Function: Rcc\_SetApb1PeriphClockState

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_TIM2_CLK_EN: Timer 2 clock enable</li> <li>• RCC_TIM3_CLK_EN: Timer 3 clock enable</li> <li>• RCC_TIM4_CLK_EN: Timer 4 clock enable</li> <li>• RCC_TIM5_CLK_EN: Timer 5 clock enable</li> <li>• RCC_TIM6_CLK_EN: Timer 6 clock enable</li> <li>• RCC_TIM7_CLK_EN: Timer 7 clock enable</li> <li>• RCC_TIM12_CLK_EN: Timer 12 clock enable</li> <li>• RCC_TIM13_CLK_EN: Timer 13 clock enable</li> <li>• RCC_TIM14_CLK_EN: Timer 14 clock enable</li> <li>• RCC_WWD_GEN_CLK_EN: Window watchdog clock enable</li> <li>• RCC_SPI2_CLK_EN: SPI 2 clock enable</li> <li>• RCC_SPI3_CLK_EN: SPI 3 clock enable</li> <li>• RCC_USART2_CLK_EN: USART 2 clock enable</li> <li>• RCC_USART3_CLK_EN: USART 3 clock enable</li> <li>• RCC_USART4_CLK_EN: USART 4 clock enable</li> <li>• RCC_USART5_CLK_EN: USART 5 clock enable</li> <li>• RCC_I2C1_CLK_EN: I2C 1 clock enable</li> <li>• RCC_I2C2_CLK_EN: I2C 2 clock enable</li> <li>• RCC_USB_CLK_EN: USB clock enable</li> <li>• RCC_CAN_CLK_EN: CAN clock enable</li> <li>• RCC_BKP_CLK_EN: Backup interface clock enable</li> <li>• RCC_PWR_CLK_EN: Power interface clock enable</li> <li>• RCC_DAC_CLK_EN: DAC interface clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.25 Rcc\_SetApb2PeriphClockState()

```
Std_ReturnType Rcc_SetApb2PeriphClockState (
```

```
uint32_t periph,
uint8_t state)
```

Choose a specific peripheral on the APB2 bus and changes its state (On / Off)

#### 4.11.2.26 Function: Rcc\_SetApb2PeriphClockState

##### Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_CLK_EN: Alternate function input output clock enable</li> <li>• RCC_IOPA_CLK_EN: Port A input output clock enable</li> <li>• RCC_IOPB_CLK_EN: Port B input output clock enable</li> <li>• RCC_IOPC_CLK_EN: Port C input output clock enable</li> <li>• RCC_IOPD_CLK_EN: Port D input output clock enable</li> <li>• RCC_IOPE_CLK_EN: Port E input output clock enable</li> <li>• RCC_IOPF_CLK_EN: Port F input output clock enable</li> <li>• RCC_IOPG_CLK_EN: Port G input output clock enable</li> <li>• RCC_ADC1_CLK_EN: ADC 1 clock enable</li> <li>• RCC_ADC2_CLK_EN: ADC 2 clock enable</li> <li>• RCC_TIM1_CLK_EN: Timer 1 clock enable</li> <li>• RCC_SPI1_CLK_EN: SPI 1 clock enable</li> <li>• RCC_TIM8_CLK_EN: Timer 8 clock enable</li> <li>• RCC_USART1_CLK_EN: USART 1 clock enable</li> <li>• RCC_ADC3_CLK_EN: ADC 3 clock enable</li> <li>• RCC_TIM9_CLK_EN: Timer 9 clock enable</li> <li>• RCC_TIM10_CLK_EN: Timer 10 clock enable</li> <li>• RCC_TIM11_CLK_EN: Timer 11 clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

##### Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

#### 4.11.2.27 Function: Rcc\_SetApb2PeriphClockState

## Parameters

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>periph</i> | <p>The peripheral clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_AFIO_CLK_EN: Alternate function input output clock enable</li> <li>• RCC_IOPA_CLK_EN: Port A input output clock enable</li> <li>• RCC_IOPB_CLK_EN: Port B input output clock enable</li> <li>• RCC_IOPC_CLK_EN: Port C input output clock enable</li> <li>• RCC_IOPD_CLK_EN: Port D input output clock enable</li> <li>• RCC_IOPE_CLK_EN: Port E input output clock enable</li> <li>• RCC_IOPF_CLK_EN: Port F input output clock enable</li> <li>• RCC_IOPG_CLK_EN: Port G input output clock enable</li> <li>• RCC_ADC1_CLK_EN: ADC 1 clock enable</li> <li>• RCC_ADC2_CLK_EN: ADC 2 clock enable</li> <li>• RCC_TIM1_CLK_EN: Timer 1 clock enable</li> <li>• RCC_SPI1_CLK_EN: SPI 1 clock enable</li> <li>• RCC_TIM8_CLK_EN: Timer 8 clock enable</li> <li>• RCC_USART1_CLK_EN: USART 1 clock enable</li> <li>• RCC_ADC3_CLK_EN: ADC 3 clock enable</li> <li>• RCC_TIM9_CLK_EN: Timer 9 clock enable</li> <li>• RCC_TIM10_CLK_EN: Timer 10 clock enable</li> <li>• RCC_TIM11_CLK_EN: Timer 11 clock enable</li> </ul> |
| <i>state</i>  | <p>The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_PERIPH_CLK_ON : To set the clock on</li> <li>• RCC_PERIPH_CLK_OFF : To set the clock off</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.28 Rcc\_SetClockState()

```
Std_ReturnType Rcc_SetClockState (
 uint32_t clock,
 uint8_t state)
```

Choose a specific clock and changes its state (On / Off)

## 4.11.2.29 Function: Rcc\_SetClockState



## Parameters

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_HSI_SET: for the high speed internal clock</li> <li>• RCC_HSE_SET: for the high speed external clock</li> <li>• RCC_PLL_SET: for the PLL clock</li> </ul> |
| <i>state</i> | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_CLK_ON : To set the clock on</li> <li>• RCC_CLK_OFF : To set the clock off</li> </ul>                                                                 |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.30 Function: Rcc\_SetClockState**

## Parameters

|              |                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_HSI_SET: for the high speed internal clock</li> <li>• RCC_HSE_SET: for the high speed external clock</li> <li>• RCC_PLL_SET: for the PLL clock</li> </ul> |
| <i>state</i> | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_CLK_ON : To set the clock on</li> <li>• RCC_CLK_OFF : To set the clock off</li> </ul>                                                                 |

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.31 Rcc\_SetPlIMultiplier()**

```
Std_ReturnType Rcc_SetPlIMultiplier (
 uint32_t pll)
```

Sets the PLL Multiplication factor.

**4.11.2.32 Function: Rcc\_SetPlIMultiplier**

**Parameters**

|            |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| <i>pll</i> | : The PLL multiplication factor                                                                    |
|            | <ul style="list-style-type: none"> <li>• RCC_PLL_MUL_XX : Set PLL multiplication factor</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.33 Function: Rcc\_SetPlIMultiplier****Parameters**

|            |                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------|
| <i>pll</i> | : The PLL multiplication factor                                                                    |
|            | <ul style="list-style-type: none"> <li>• RCC_PLL_MUL_XX : Set PLL multiplication factor</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.34 Rcc\_SetPlISource()**

```
Std_ReturnType Rcc_SetPlISource (
 uint32_t source)
```

Chooses the PLL clock source.

**4.11.2.35 Function: Rcc\_SetPlISource****Parameters**

|               |                                                                                                                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>source</i> | : The PLL clock source                                                                                                                                                                                               |
|               | <ul style="list-style-type: none"> <li>• RCC_PLL_SRC_HSI : Choose high speed internal clock / 2 as a PLL clock source</li> <li>• RCC_PLL_SRC_HSE : Choose high speed external clock as a PLL clock source</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.36 Function: Rcc\_SetPlISource**

## Parameters

|               |                                                                                                                                                                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>source</i> | : The PLL clock source <ul style="list-style-type: none"> <li>• RCC_PLL_SRC_HSI : Choose high speed internal clock / 2 as a PLL clock source</li> <li>• RCC_PLL_SRC_HSE : Choose high speed external clock as a PLL clock source</li> </ul> |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.37 Rcc\_SetPrescaler()

```
Std_ReturnType Rcc_SetPrescaler (
 uint32_t clock,
 uint32_t value)
```

Sets the prescaler value for a specific clock.

## 4.11.2.38 Function: Rcc\_SetPrescaler

## Parameters

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | The clock you want to configure <ul style="list-style-type: none"> <li>• RCC_USB_PRE : For the USB prescaler</li> <li>• RCC_PLL_HSE_PRE : For the Pll prescaler</li> <li>• RCC_ADC_PRE : For the ADC prescaler</li> <li>• RCC_APB2_PRE : For the APB2 prescaler</li> <li>• RCC_APB1_PRE : For the APB1 prescaler</li> <li>• RCC_AHB_PRE : For the AHB prescaler</li> </ul>                                                                                             |
| <i>value</i> | : The state of the clock (On / Off) <ul style="list-style-type: none"> <li>• RCC_USB_PRE_1_5 : No USB prescaler value</li> <li>• RCC_USB_PRE_1_5 : USB prescaler 1.5</li> <li>• RCC_PLL_HSE_PRE_X : PLL Prescaler value using high speed external clock</li> <li>• RCC_ADC_PRE_X : ADC Prescaler value</li> <li>• RCC_APB2_PRE_XX : APB2 prescaler value</li> <li>• RCC_APB1_PRE_XX : APB1 prescaler value</li> <li>• RCC_AHB_PRE_XXX : AHB prescaler value</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.39 Function: Rcc\_SetPrescaler****Parameters**

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | <p>The clock you want to configure</p> <ul style="list-style-type: none"> <li>• RCC_USB_PRE : For the USB prescaler</li> <li>• RCC_PLL_HSE_PRE : For the Pll prescaler</li> <li>• RCC_ADC_PRE : For the ADC prescaler</li> <li>• RCC_APB2_PRE : For the APB2 prescaler</li> <li>• RCC_APB1_PRE : For the APB1 prescaler</li> <li>• RCC_AHB_PRE : For the AHB prescaler</li> </ul>                                                                                             |
| <i>value</i> | <p>: The state of the clock (On / Off)</p> <ul style="list-style-type: none"> <li>• RCC_USB_PRE_1_5 : No USB prescaler value</li> <li>• RCC_USB_PRE_1_5 : USB prescaler 1.5</li> <li>• RCC_PLL_HSE_PRE_X : PLL Prescaler value using high speed external clock</li> <li>• RCC_ADC_PRE_X : ADC Prescaler value</li> <li>• RCC_APB2_PRE_XX : APB2 prescaler value</li> <li>• RCC_APB1_PRE_XX : APB1 prescaler value</li> <li>• RCC_AHB_PRE_XXX : AHB prescaler value</li> </ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.11.2.40 Rcc\_SwitchSystemClock()**

```
Std_ReturnType Rcc_SwitchSystemClock (
 uint32_t clock)
```

Switches the system clock (HSI / HSE / PLL)

**4.11.2.41 Function: Rcc\_SwitchSystemClock**

## Parameters

|              |                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | : The PLL clock source <ul style="list-style-type: none"> <li>• RCC_SYS_CLK_SELECT_HSI : Select high speed internal clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_HSE : Select high speed external clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_PLL : Select PLL clock as system clock</li> </ul> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.11.2.42 Function: Rcc\_SwitchSystemClock

## Parameters

|              |                                                                                                                                                                                                                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>clock</i> | : The PLL clock source <ul style="list-style-type: none"> <li>• RCC_SYS_CLK_SELECT_HSI : Select high speed internal clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_HSE : Select high speed external clock as system clock</li> <li>• RCC_SYS_CLK_SELECT_PLL : Select PLL clock as system clock</li> </ul> |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.12 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 "Hrcc.h"
#include "Switch.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]

### 4.12.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

### 4.12.2 Function Documentation

#### 4.12.2.1 Switch\_GetSwitchStatus()

```
Std_ReturnType Switch_GetSwitchStatus (
 uint8_t switchName,
 uint8_t * state)
```

Gets the status of the switch.

#### 4.12.2.2 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"><li>• SWITCH_PRESSED : if the switch is pressed</li><li>• SWITCH_NOT_PRESSED : if the switch is not pressed</li></ul> |

##### 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 Switch\_Init()

```
Std_ReturnType Switch_Init (
 void)
```

Initializes GPIOs for the Switches.

#### 4.12.2.4 Function: Switch\_Init

##### Returns

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

## 4.13 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](#)  
*The Switch pin layout.*

### Macros

- #define **SWITCH\_PRESSED** 0
- #define **SWITCH\_NOT\_PRESSED** !SWITCH\_PRESSED

### 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.*

#### 4.13.1 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

#### 4.13.2 Function Documentation

##### 4.13.2.1 Switch\_GetSwitchStatus()

```
Std_ReturnType Switch_GetSwitchStatus (
 uint8_t switchName,
 uint8_t * state)
```

Gets the status of the switch.

##### 4.13.2.2 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"><li>• SWITCH_PRESSED : if the switch is pressed</li><li>• SWITCH_NOT_PRESSED : if the switch is not pressed</li></ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.13.2.3 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"><li>• SWITCH_PRESSED : if the switch is pressed</li><li>• SWITCH_NOT_PRESSED : if the switch is not pressed</li></ul> |

**Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.13.2.4 Switch\_Init()**

```
Std_ReturnType Switch_Init (
 void)
```

Initializes GPIOs for the Switches.

**4.13.2.5 Function: Switch\_Init****Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly

**4.13.2.6 Function: Switch\_Init****Returns**

: A status E\_OK : if the function is executed correctly E\_NOT\_OK : if the function is not executed correctly



## 4.14 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_Cfg.h"
#include "Switch.h"
```

### Variables

- const [switch\\_t](#) **Switch\_switches** [SWITCH\_NUMBER\_OF\_SWITCHES]

#### 4.14.1 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

#### 4.14.2 Variable Documentation

##### 4.14.2.1 Switch\_switches

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

##### Initial value:

```
= {
 {GPIO_PIN_8, GPIO_PORTA, GPIO_PIN_RESET}
}
```

## 4.15 Switch\_Cfg.h File Reference

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

### Macros

- #define **SWITCH\_NUMBER\_OF\_SWITCHES** 1
- #define **BOOTLOADER\_SWITCH** 0

### 4.15.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

## 4.16 Uart.c File Reference

This is the implementation for the UART driver.

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

### Data Structures

- struct [uart\\_t](#)  
*The UART Registers.*
- struct [dataBuffer\\_t](#)  
*The Uart Data Buffer.*

### Macros

- #define **UART\_NUMBER\_OF\_MODULES** 3
- #define **UART\_INT\_NUMBER** 37
- #define **UART\_BUFFER\_IDLE** 0
- #define **UART\_BUFFER\_BUSY** 1
- #define **UART\_TXE\_CLR** 0xFFFFFFFF7F
- #define **UART\_TC\_CLR** 0xFFFFFFFFBF
- #define **UART\_RXNE\_CLR** 0xFFFFFFFFDF
- #define **UART\_PE\_CLR** 0xFFFFFFFFFE
- #define **UART\_DR\_CLR** 0xFFFFFE00
- #define **UART\_STOP\_CLR** 0xFFFFCFFF
- #define **UART\_TXEIE\_CLR** 0xFFFF7F
- #define **UART\_PS\_CLR** 0xFFFFDFF
- #define **UART\_M\_CLR** 0xFFFFEFFF
- #define **UART\_LBD\_CLR** 0xFFFFEFFF
- #define **UART\_LBDIE\_CLR** 0xFFFFFBF
- #define **UART\_TXE\_GET** 0x00000080
- #define **UART\_TC\_GET** 0x00000040
- #define **UART\_RXNE\_GET** 0x00000020
- #define **UART\_PE\_GET** 0x00000001

- #define **UART\_UE\_SET** 0x00002000
- #define **UART\_PCE\_SET** 0x00000400
- #define **UART\_PEIE\_SET** 0x00000100
- #define **UART\_TXEIE\_SET** 0x00000080
- #define **UART\_TCIE\_SET** 0x00000040
- #define **UART\_RXNEIE\_SET** 0x00000020
- #define **UART\_IDLEIE\_SET** 0x00000010
- #define **UART\_TE\_SET** 0x00000008
- #define **UART\_RE\_SET** 0x00000004
- #define **UART\_M\_SET** 0x00001000
- #define **UART\_LBD\_SET** 0x00000100
- #define **UART\_LBDIE\_SET** 0x00000040
- #define **UART\_DMAT\_SET** 0x00000080
- #define **UART\_DMAR\_SET** 0x00000040
- #define **UART\_SBK\_SET** 0x00000001
- #define **UART\_LINEN\_CLR** 0xFFFFBFFF
- #define **UART\_RTSE\_CLR** 0xFFFFFEFF
- #define **UART\_NO\_PRESCALER** 0x1
- #define **DMA\_DID\_NOT\_RECEIVE** 0
- #define **DMA\_RECEIVED** 1

## Functions

- Std\_ReturnType **Uart\_Init** (**Uart\_cfg\_t** \*cfgUart)  
*Initializes the UART.*
- Std\_ReturnType **Uart\_Send** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Sends data through the UART.*
- Std\_ReturnType **Uart\_Receive** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Receives data through the UART.*
- Std\_ReturnType **Uart\_SendSync** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Sends data through the UART synchronously.*
- Std\_ReturnType **Uart\_ReceiveSync** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Receives data through the UART synchronously.*
- Std\_ReturnType **Uart\_SetTxCb** (txCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when transmission is completed.*
- Std\_ReturnType **Uart\_SetRxCb** (rxCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when receive is completed.*
- Std\_ReturnType **Uart\_SetBreakCb** (brCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when break happens.*
- Std\_ReturnType **Uart\_SendBreak** (uint8\_t uartModule)  
*Sends a Lin break of 13 bit length.*
- void **USART1\_IRQHandler** (void)  
*The UART 1 Handler.*
- void **USART2\_IRQHandler** (void)  
*The UART 2 Handler.*
- void **USART3\_IRQHandler** (void)  
*The UART 3 Handler.*

## Variables

- const uint32\_t [Uart\\_Address](#) [UART\_NUMBER\_OF\_MODULES]  
*The Base Adresses of the UART module.*
- const volatile uint8\_t **Uart\_DmaTxChannelNumber** [UART\_NUMBER\_OF\_MODULES]
- const volatile uint8\_t **Uart\_DmaRxChannelNumber** [UART\_NUMBER\_OF\_MODULES]

### 4.16.1 Detailed Description

This is the implementation for the UART driver.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com) )

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.16.2 Function Documentation

#### 4.16.2.1 Uart\_Init()

```
Std_ReturnType Uart_Init (
 Uart_cfg_t * cfgUart)
```

Initializes the UART.

#### Parameters

|            |                         |
|------------|-------------------------|
| <i>cfg</i> | The Uart Configurations |
|------------|-------------------------|

#### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.16.2.2 Uart\_Receive()

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

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                                                                                       |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

##### 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

#### 4.16.2.3 Uart\_ReceiveSync()

```
Std_ReturnType Uart_ReceiveSync (
 uint8_t * data,
 uint16_t length,
 uint8_t uartModule)
```

Receives data through the UART synchronously.

##### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The buffer to receive data in                                                                                         |
| <i>length</i>     | the length of the data in bytes                                                                                       |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

##### 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

#### 4.16.2.4 Uart\_Send()

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

Sends data through the UART.

##### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The data to send                                                                                                          |
| <i>length</i>     | the length of the data in bytes                                                                                           |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |

##### 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

#### 4.16.2.5 Uart\_SendBreak()

```
Std_ReturnType Uart_SendBreak (
 uint8_t uartModule)
```

Sends a Lin break of 13 bit length.

##### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|

##### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.16.2.6 Uart\_SendSync()

```
Std_ReturnType Uart_SendSync (
 uint8_t * data,
 uint16_t length,
 uint8_t uartModule)
```

Sends data through the UART synchronously.

##### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The data to send                                                                                                      |
| <i>length</i>     | the length of the data in bytes                                                                                       |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

##### 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

#### 4.16.2.7 Uart\_SetBreakCb()

```
Std_ReturnType Uart_SetBreakCb (
 brCb_t func,
 uint8_t uartModule)
```

Sets the callback function that will be called when break happens.

##### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                 |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

**Returns**

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

**4.16.2.8 Uart\_SetRxCb()**

```
Std_ReturnType Uart_SetRxCb (
 rxCb_t func,
 uint8_t uartModule)
```

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

**Parameters**

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                     |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |

**Returns**

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

**4.16.2.9 Uart\_SetTxCb()**

```
Std_ReturnType Uart_SetTxCb (
 txCb_t func,
 uint8_t uartModule)
```

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

**Parameters**

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                     |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |



#### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

##### 4.16.2.10 USART1\_IRQHandler()

```
void USART1_IRQHandler (
 void)
```

The UART 1 Handler.

##### 4.16.2.11 USART2\_IRQHandler()

```
void USART2_IRQHandler (
 void)
```

The UART 2 Handler.

##### 4.16.2.12 USART3\_IRQHandler()

```
void USART3_IRQHandler (
 void)
```

The UART 3 Handler.

### 4.16.3 Variable Documentation

#### 4.16.3.1 Uart\_Address

```
const uint32_t Uart_Address[UART_NUMBER_OF_MODULES]
```

##### Initial value:

```
= {
 0x40013800,
 0x40004400,
 0x40004800
}
```

The Base Adresses of the UART module.

#### 4.16.3.2 Uart\_DmaRxChannelNumber

```
const volatile uint8_t Uart_DmaRxChannelNumber[UART_NUMBER_OF_MODULES]
```

**Initial value:**

```
=
{
 DMA_CH_5,
 DMA_CH_6,
 DMA_CH_3
}
```

#### 4.16.3.3 Uart\_DmaTxChannelNumber

```
const volatile uint8_t Uart_DmaTxChannelNumber[UART_NUMBER_OF_MODULES]
```

**Initial value:**

```
=
{
 DMA_CH_4,
 DMA_CH_7,
 DMA_CH_2
}
```

## 4.17 Uart.h File Reference

This is the user interface for the UART driver.

### Data Structures

- struct [Uart\\_cfg\\_t](#)

### Macros

- `#define UART1 0`
- `#define UART2 1`
- `#define UART3 2`
- `#define UART_ODD_PARITY 0x00000200`
- `#define UART_EVEN_PARITY 0x00000000`
- `#define UART_NO_PARITY 0xFFFFFBFF`
- `#define UART_STOP_ONE_BIT 0x00000000`
- `#define UART_STOP_TWO_BITS 0x00003000`
- `#define UART_FLOW_CONTROL_EN 0x00000100`
- `#define UART_FLOW_CONTROL_DIS 0x00000000`
- `#define UART_LIN_EN 0x00004000`
- `#define UART_LIN_DIS 0x00000000`
- `#define UART_INTERRUPT_DIS 0`
- `#define UART_INTERRUPT_TXE 1`
- `#define UART_INTERRUPT_TC 2`
- `#define UART_INTERRUPT_RXNE 4`
- `#define UART_INTERRUPT_LBD 8`
- `#define UART_MODE_ASYNC 0`
- `#define UART_MODE_DMA 1`

## Typedefs

- typedef void(\* **txCb\_t**) (uint8\_t)
- typedef void(\* **rxCb\_t**) (uint8\_t)
- typedef void(\* **brCb\_t**) (uint8\_t)

## Functions

- Std\_ReturnType **Uart\_Init** (**Uart\_cfg\_t** \*cfgUart)  
*Initializes the UART.*
- Std\_ReturnType **Uart\_Send** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Sends data through the UART.*
- Std\_ReturnType **Uart\_SendBreak** (uint8\_t uartModule)  
*Sends a Lin break of 13 bit length.*
- Std\_ReturnType **Uart\_Receive** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Receives data through the UART.*
- Std\_ReturnType **Uart\_SendSync** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Sends data through the UART synchronously.*
- Std\_ReturnType **Uart\_ReceiveSync** (uint8\_t \*data, uint16\_t length, uint8\_t uartModule)  
*Receives data through the UART synchronously.*
- Std\_ReturnType **Uart\_SetTxCb** (txCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when transmission is completed.*
- Std\_ReturnType **Uart\_SetRxCb** (rxCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when receive is completed.*
- Std\_ReturnType **Uart\_SetBreakCb** (brCb\_t func, uint8\_t uartModule)  
*Sets the callback function that will be called when break happens.*

### 4.17.1 Detailed Description

This is the user interface for the UART driver.

#### Author

Mark Attia ( [markjosephattia@gmail.com](mailto:markjosephattia@gmail.com) )

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.17.2 Function Documentation

#### 4.17.2.1 Uart\_Init()

```
Std_ReturnType Uart_Init (
 Uart_cfg_t * cfgUart)
```

Initializes the UART.

## Parameters

|                |                         |
|----------------|-------------------------|
| <i>cfgUart</i> | The Uart Configurations |
|----------------|-------------------------|

## Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

## Parameters

|            |                         |
|------------|-------------------------|
| <i>cfg</i> | The Uart Configurations |
|------------|-------------------------|

## Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.17.2.2 Uart\_Receive()

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

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                                                                                           |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |

## 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

### 4.17.2.3 Uart\_ReceiveSync()

```
Std_ReturnType Uart_ReceiveSync (
 uint8_t * data,
 uint16_t length,
 uint8_t uartModule)
```

Receives data through the UART synchronously.

#### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The buffer to receive data in                                                                                         |
| <i>length</i>     | the length of the data in bytes                                                                                       |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

#### 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

### 4.17.2.4 Uart\_Send()

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

Sends data through the UART.

#### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The data to send                                                                                                      |
| <i>length</i>     | the length of the data in bytes                                                                                       |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

#### 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

#### 4.17.2.5 Uart\_SendBreak()

```
Std_ReturnType Uart_SendBreak (
 uint8_t uartModule)
```

Sends a Lin break of 13 bit length.

##### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|

##### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.17.2.6 Uart\_SendSync()

```
Std_ReturnType Uart_SendSync (
 uint8_t * data,
 uint16_t length,
 uint8_t uartModule)
```

Sends data through the UART synchronously.

##### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>data</i>       | The data to send                                                                                                          |
| <i>length</i>     | the length of the data in bytes                                                                                           |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |

##### 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

#### 4.17.2.7 Uart\_SetBreakCb()

```
Std_ReturnType Uart_SetBreakCb (
 brCb_t func,
 uint8_t uartModule)
```

Sets the callback function that will be called when break happens.

##### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                 |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

##### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.17.2.8 Uart\_SetRxCb()

```
Std_ReturnType Uart_SetRxCb (
 rxCb_t func,
 uint8_t uartModule)
```

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

##### Parameters

|                   |                                                                                                                       |
|-------------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                 |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"><li>• UART1</li><li>• UART2</li><li>• UART3</li></ul> |

##### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

#### 4.17.2.9 Uart\_SetTxCb()

```
Std_ReturnType Uart_SetTxCb (
 txCb_t func,
 uint8_t uartModule)
```

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

##### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>func</i>       | the callback function                                                                                                     |
| <i>uartModule</i> | the module number of the UART <ul style="list-style-type: none"> <li>• UART1</li> <li>• UART2</li> <li>• UART3</li> </ul> |

##### Returns

Std\_ReturnType A Status E\_OK: If the function executed successfully E\_NOT\_OK: If the did not execute successfully

## 4.18 Uart\_Cfg.h File Reference

Those are the user configurations for the Uart Driver.

### Macros

- `#define UART_MODE UART_MODE_ASYNC`

#### 4.18.1 Detailed Description

Those are the user configurations for the Uart Driver.

##### Author

Mark Attia (markjosephattia@gmailcom)

##### Version

0.1

##### Date

2020-04-04

##### Copyright

Copyright (c) 2020



# Index

Bootloader.c, [9](#)

dataBuffer\_t, [5](#)

Fpec.h, [10](#)

    Fpec\_ErasePage, [11](#)

    Fpec\_Lock, [11](#)

    Fpec\_MassErase, [11](#)

    Fpec\_Unlock, [11](#)

    Fpec\_WriteBlock, [12](#)

    Fpec\_WriteHalfWord, [12](#)

Fpec\_ErasePage

    Fpec.h, [11](#)

Fpec\_Lock

    Fpec.h, [11](#)

Fpec\_MassErase

    Fpec.h, [11](#)

Fpec\_t, [5](#)

Fpec\_Unlock

    Fpec.h, [11](#)

Fpec\_WriteBlock

    Fpec.h, [12](#)

Fpec\_WriteHalfWord

    Fpec.h, [12](#)

Gpio.c, [13](#)

    Gpio\_InitPins, [13](#)

    Gpio\_ReadPin, [14](#)

    Gpio\_WritePin, [14](#)

Gpio.h, [15](#)

    Gpio\_InitPins, [16](#)

    Gpio\_ReadPin, [17](#)

    Gpio\_WritePin, [18](#)

Gpio\_InitPins

    Gpio.c, [13](#)

    Gpio.h, [16](#)

Gpio\_ReadPin

    Gpio.c, [14](#)

    Gpio.h, [17](#)

gpio\_t, [6](#)

Gpio\_WritePin

    Gpio.c, [14](#)

    Gpio.h, [18](#)

gpioReg\_t, [6](#)

header\_t, [6](#)

HRcc.h, [19](#)

    HRcc\_EnPortClock, [20](#)

    HRcc\_SystemClockInit, [20](#)

HRcc\_EnPortClock

HRcc.h, [20](#)

HRcc\_SystemClockInit

HRcc.h, [20](#)

Nvic.c, [20](#)

    Nvic\_ClearFault, [22](#)

    Nvic\_ClearPending, [22](#)

    Nvic\_ConfigGroupSize, [23](#)

    Nvic\_DisableInterrupt, [23](#)

    Nvic\_DisablePeripheral, [23](#)

    Nvic\_EnableInterrupt, [24](#)

    Nvic\_EnablePeripheral, [24](#)

    Nvic\_FilterPriority, [24](#)

    Nvic\_GenerateSoftwareInterrupt, [25](#)

    Nvic\_GetGroupPriority, [25](#)

    Nvic\_GetSubpriority, [25](#)

    Nvic\_IsInterruptActive, [26](#)

    Nvic\_ResetSystem, [26](#)

    Nvic\_SetFault, [26](#)

    Nvic\_SetGroupPriority, [27](#)

    Nvic\_SetPending, [27](#)

    Nvic\_SetSubpriority, [28](#)

Nvic.h, [28](#)

    Nvic\_ClearFault, [31](#)

    Nvic\_ClearPending, [31](#)

    Nvic\_ConfigGroupSize, [31](#)

    Nvic\_DisableInterrupt, [32](#)

    Nvic\_DisablePeripheral, [32](#)

    Nvic\_EnableInterrupt, [32](#)

    Nvic\_EnablePeripheral, [33](#)

    Nvic\_FilterPriority, [33](#)

    Nvic\_GenerateSoftwareInterrupt, [33](#)

    Nvic\_GetGroupPriority, [34](#)

    Nvic\_GetSubpriority, [34](#)

    NVIC\_IRQNUM\_WWDG, [31](#)

    Nvic\_IsInterruptActive, [35](#)

    Nvic\_ResetSystem, [35](#)

    Nvic\_SetFault, [35](#)

    Nvic\_SetGroupPriority, [35](#)

    Nvic\_SetPending, [36](#)

    Nvic\_SetSubpriority, [36](#)

Nvic\_ClearFault

    Nvic.c, [22](#)

    Nvic.h, [31](#)

Nvic\_ClearPending

    Nvic.c, [22](#)

    Nvic.h, [31](#)

Nvic\_ConfigGroupSize

    Nvic.c, [23](#)

    Nvic.h, [31](#)

- Nvic\_DisableInterrupt
  - Nvic.c, [23](#)
  - Nvic.h, [32](#)
- Nvic\_DisablePeripheral
  - Nvic.c, [23](#)
  - Nvic.h, [32](#)
- Nvic\_EnableInterrupt
  - Nvic.c, [24](#)
  - Nvic.h, [32](#)
- Nvic\_EnablePeripheral
  - Nvic.c, [24](#)
  - Nvic.h, [33](#)
- Nvic\_FilterPriority
  - Nvic.c, [24](#)
  - Nvic.h, [33](#)
- Nvic\_GenerateSoftwareInterrupt
  - Nvic.c, [25](#)
  - Nvic.h, [33](#)
- Nvic\_GetGroupPriority
  - Nvic.c, [25](#)
  - Nvic.h, [34](#)
- Nvic\_GetSubpriority
  - Nvic.c, [25](#)
  - Nvic.h, [34](#)
- NVIC\_IRQNUM\_WWDG
  - Nvic.h, [31](#)
- Nvic\_IsInterruptActive
  - Nvic.c, [26](#)
  - Nvic.h, [35](#)
- Nvic\_ResetSystem
  - Nvic.c, [26](#)
  - Nvic.h, [35](#)
- Nvic\_SetFault
  - Nvic.c, [26](#)
  - Nvic.h, [35](#)
- Nvic\_SetGroupPriority
  - Nvic.c, [27](#)
  - Nvic.h, [35](#)
- Nvic\_SetPending
  - Nvic.c, [27](#)
  - Nvic.h, [36](#)
- Nvic\_SetSubpriority
  - Nvic.c, [28](#)
  - Nvic.h, [36](#)
- nvic\_t, [7](#)
- packet\_t, [7](#)
- Protocol.c, [37](#)
  - Protocol\_Receive, [38](#)
  - Protocol\_Send, [38](#)
  - Protocol\_SendAck, [39](#)
- Protocol.h, [39](#)
  - Protocol\_Receive, [40](#)
  - Protocol\_Send, [41](#)
  - Protocol\_SendAck, [41](#)
- Protocol\_Receive
  - Protocol.c, [38](#)
  - Protocol.h, [40](#)
- Protocol\_Send
  - Protocol.c, [38](#)
  - Protocol.h, [41](#)
- Protocol\_SendAck
  - Protocol.c, [39](#)
  - Protocol.h, [41](#)
- Rcc.c, [41](#)
  - Rcc\_GetSystemClockStatus, [43](#)
  - Rcc\_IsClockReady, [43](#)
  - Rcc\_ResetAhbPeriph, [44](#)
  - Rcc\_ResetApb1Periph, [44](#)
  - Rcc\_ResetApb2Periph, [46](#)
  - Rcc\_SelectMcoClock, [47](#)
  - Rcc\_SetAhbPeriphClockState, [48](#)
  - Rcc\_SetApb1PeriphClockState, [49](#)
  - Rcc\_SetApb2PeriphClockState, [50](#)
  - Rcc\_SetClockState, [51](#)
  - Rcc\_SetPLIMultiplier, [52](#)
  - Rcc\_SetPLISource, [52](#)
  - Rcc\_SetPrescaler, [53](#)
  - Rcc\_SwitchSystemClock, [54](#)
- Rcc.h, [54](#)
  - Rcc\_GetSystemClockStatus, [59](#)
  - Rcc\_IsClockReady, [59](#)
  - Rcc\_ResetAhbPeriph, [60](#)
  - Rcc\_ResetApb1Periph, [61](#)
  - Rcc\_ResetApb2Periph, [63](#)
  - Rcc\_SelectMcoClock, [65](#)
  - Rcc\_SetAhbPeriphClockState, [66](#)
  - Rcc\_SetApb1PeriphClockState, [68](#)
  - Rcc\_SetApb2PeriphClockState, [70](#)
  - Rcc\_SetClockState, [72](#)
  - Rcc\_SetPLIMultiplier, [73](#)
  - Rcc\_SetPLISource, [74](#)
  - Rcc\_SetPrescaler, [75](#)
  - Rcc\_SwitchSystemClock, [76](#)
- Rcc\_GetSystemClockStatus
  - Rcc.c, [43](#)
  - Rcc.h, [59](#)
- Rcc\_IsClockReady
  - Rcc.c, [43](#)
  - Rcc.h, [59](#)
- Rcc\_ResetAhbPeriph
  - Rcc.c, [44](#)
  - Rcc.h, [60](#)
- Rcc\_ResetApb1Periph
  - Rcc.c, [44](#)
  - Rcc.h, [61](#)
- Rcc\_ResetApb2Periph
  - Rcc.c, [46](#)
  - Rcc.h, [63](#)
- Rcc\_SelectMcoClock
  - Rcc.c, [47](#)
  - Rcc.h, [65](#)
- Rcc\_SetAhbPeriphClockState
  - Rcc.c, [48](#)
  - Rcc.h, [66](#)
- Rcc\_SetApb1PeriphClockState
  - Rcc.c, [49](#)

- Rcc.h, 68
- Rcc\_SetApb2PeriphClockState
  - Rcc.c, 50
  - Rcc.h, 70
- Rcc\_SetClockState
  - Rcc.c, 51
  - Rcc.h, 72
- Rcc\_SetPllMultiplier
  - Rcc.c, 52
  - Rcc.h, 73
- Rcc\_SetPllSource
  - Rcc.c, 52
  - Rcc.h, 74
- Rcc\_SetPrescaler
  - Rcc.c, 53
  - Rcc.h, 75
- Rcc\_SwitchSystemClock
  - Rcc.c, 54
  - Rcc.h, 76
- Switch.c, 77
  - Switch\_GetSwitchStatus, 78
  - Switch\_Init, 78
- Switch.h, 79
  - Switch\_GetSwitchStatus, 79
  - Switch\_Init, 80
- Switch\_Cfg.c, 81
  - Switch\_switches, 81
- Switch\_Cfg.h, 81
- Switch\_GetSwitchStatus
  - Switch.c, 78
  - Switch.h, 79
- Switch\_Init
  - Switch.c, 78
  - Switch.h, 80
- Switch\_switches
  - Switch\_Cfg.c, 81
- switch\_t, 7
- Uart.c, 82
  - Uart\_Address, 89
  - Uart\_DmaRxChannelNumber, 89
  - Uart\_DmaTxChannelNumber, 90
  - Uart\_Init, 84
  - Uart\_Receive, 84
  - Uart\_ReceiveSync, 85
  - Uart\_Send, 86
  - Uart\_SendBreak, 86
  - Uart\_SendSync, 87
  - Uart\_SetBreakCb, 87
  - Uart\_SetRxCb, 88
  - Uart\_SetTxCb, 88
  - USART1\_IRQHandler, 89
  - USART2\_IRQHandler, 89
  - USART3\_IRQHandler, 89
- Uart.h, 90
  - Uart\_Init, 91
  - Uart\_Receive, 92
  - Uart\_ReceiveSync, 92
  - Uart\_Send, 93
  - Uart\_SendBreak, 94
  - Uart\_SendSync, 94
  - Uart\_SetBreakCb, 95
  - Uart\_SetRxCb, 95
  - Uart\_SetTxCb, 95
- Uart\_Address
  - Uart.c, 89
- Uart\_Cfg.h, 96
- Uart\_cfg\_t, 8
- Uart\_DmaRxChannelNumber
  - Uart.c, 89
- Uart\_DmaTxChannelNumber
  - Uart.c, 90
- Uart\_Init
  - Uart.c, 84
  - Uart.h, 91
- Uart\_Receive
  - Uart.c, 84
  - Uart.h, 92
- Uart\_ReceiveSync
  - Uart.c, 85
  - Uart.h, 92
- Uart\_Send
  - Uart.c, 86
  - Uart.h, 93
- Uart\_SendBreak
  - Uart.c, 86
  - Uart.h, 94
- Uart\_SendSync
  - Uart.c, 87
  - Uart.h, 94
- Uart\_SetBreakCb
  - Uart.c, 87
  - Uart.h, 95
- Uart\_SetRxCb
  - Uart.c, 88
  - Uart.h, 95
- Uart\_SetTxCb
  - Uart.c, 88
  - Uart.h, 95
- uart\_t, 8
- USART1\_IRQHandler
  - Uart.c, 89
- USART2\_IRQHandler
  - Uart.c, 89
- USART3\_IRQHandler
  - Uart.c, 89