

## Two Counters

Generated by Doxygen 1.8.17



<b>1 Data Structure Index</b>	<b>1</b>
1.1 Data Structures . . . . .	1
<b>2 File Index</b>	<b>3</b>
2.1 File List . . . . .	3
<b>3 Data Structure Documentation</b>	<b>5</b>
3.1 clcd_t Struct Reference . . . . .	5
3.2 dataBuffer_t Struct Reference . . . . .	5
3.3 frame_t Union Reference . . . . .	6
3.3.1 Detailed Description . . . . .	6
3.4 gpio_t Struct Reference . . . . .	6
3.5 hUartConfig_t Struct Reference . . . . .	6
3.6 led_t Struct Reference . . . . .	7
3.7 NVIC_regMap Struct Reference . . . . .	7
3.8 RCC_regMap Struct Reference . . . . .	7
3.9 switch_t Struct Reference . . . . .	8
3.10 SysTask Struct Reference . . . . .	8
3.11 SYSTICK_regMap Struct Reference . . . . .	8
3.12 Task Struct Reference . . . . .	8
3.13 uart_t Struct Reference . . . . .	9
<b>4 File Documentation</b>	<b>11</b>
4.1 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/App.h File Reference	11
4.1.1 Detailed Description . . . . .	11
4.1.2 Function Documentation . . . . .	12
4.1.2.1 APP_init() . . . . .	12
4.1.2.2 APP_receiveFcn() . . . . .	12
4.1.2.3 APP_sendTask() . . . . .	12
4.2 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/CLcd.h File Reference	12
4.2.1 Detailed Description . . . . .	13
4.2.2 Function Documentation . . . . .	14
4.2.2.1 CLcd_ClearDisplay() . . . . .	14
4.2.2.2 CLcd_ConfigCursor() . . . . .	14
4.2.2.3 CLcd_ConfigDisplay() . . . . .	14
4.2.2.4 CLcd_GotoXY() . . . . .	15
4.2.2.5 CLcd_Init() . . . . .	15
4.2.2.6 CLcd_SetDoneNotification() . . . . .	16
4.2.2.7 CLcd_Task() . . . . .	16
4.2.2.8 CLcd_WriteString() . . . . .	16
4.3 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Gpio.h File Reference	17
4.3.1 Detailed Description . . . . .	18
4.3.2 Function Documentation . . . . .	18

4.3.2.1 Gpio_InitPins()	18
4.3.2.2 Function: Gpio_InitPins	18
4.3.2.3 Function: Gpio_InitPins	18
4.3.2.4 Gpio_ReadPin()	19
4.3.2.5 Function: Gpio_ReadPin	19
4.3.2.6 Function: Gpio_ReadPin	19
4.3.2.7 Gpio_WritePin()	20
4.3.2.8 Function: Gpio_WritePin	20
4.3.2.9 Function: Gpio_WritePin	20
4.4 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HRcc.h File Reference	20
4.4.1 Detailed Description	21
4.4.2 Function Documentation	21
4.4.2.1 HRcc_EnPortClock()	21
4.4.2.2 HRcc_SystemClockInit()	22
4.5 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUart.h File Reference	22
4.5.1 Detailed Description	23
4.5.2 Function Documentation	23
4.5.2.1 HUart_Config()	23
4.5.2.2 HUart_Init()	24
4.5.2.3 HUart_Receive()	24
4.5.2.4 HUart_Send()	24
4.5.2.5 HUart_SetModule()	25
4.5.2.6 HUart_SetRxCb()	25
4.5.2.7 HUart_SetTxCb()	25
4.6 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUart_Cfg.h File Reference	26
4.6.1 Detailed Description	26
4.7 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led.h File Reference	27
4.7.1 Detailed Description	27
4.7.2 Function Documentation	27
4.7.2.1 Led_Init()	28
4.7.2.2 Function: Led_Init	28
4.7.2.3 Function: Led_Init	28
4.7.2.4 Led_SetLedOff()	28
4.7.2.5 Function: Led_SetLedOff	28
4.7.2.6 Function: Led_SetLedOff	28
4.7.2.7 Led_SetLedOn()	29
4.7.2.8 Function: Led_SetLedOn	29
4.7.2.9 Function: Led_SetLedOn	29
4.7.2.10 Led_SetLedStatus()	29
4.7.2.11 Function: Led_SetLedStatus	29

4.7.2.12 Function: Led_SetLedStatus . . . . .	30
4.8 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led_Cfg.h File Reference . . . . .	30
4.8.1 Detailed Description . . . . .	30
4.9 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/NVIC.h File Reference	30
4.9.1 Detailed Description . . . . .	32
4.9.2 Macro Definition Documentation . . . . .	33
4.9.2.1 NVIC_DISABLE . . . . .	33
4.9.2.2 NVIC_IRQNUM_WWDG . . . . .	33
4.9.2.3 NVIC_RESET . . . . .	33
4.9.3 Function Documentation . . . . .	33
4.9.3.1 NVIC_configurePriority() . . . . .	33
4.9.3.2 NVIC_controlAllPeripheral() . . . . .	33
4.9.3.3 NVIC_controlFault() . . . . .	35
4.9.3.4 NVIC_controlInterrupt() . . . . .	35
4.9.3.5 NVIC_controlPendingFlag() . . . . .	35
4.9.3.6 NVIC_filterInterrupts() . . . . .	36
4.9.3.7 NVIC_getActiveFlagStatus() . . . . .	36
4.9.3.8 NVIC_getPriority() . . . . .	36
4.10 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/RCC.h File Reference . . . . .	37
4.10.1 Detailed Description . . . . .	39
4.10.2 Function Documentation . . . . .	39
4.10.2.1 RCC_configureMCO() . . . . .	40
4.10.2.2 RCC_configurePLL() . . . . .	40
4.10.2.3 RCC_configurePrescalers() . . . . .	40
4.10.2.4 RCC_controlAHBPeripheral() . . . . .	41
4.10.2.5 RCC_controlAPB1Peripheral() . . . . .	41
4.10.2.6 RCC_controlAPB2Peripheral() . . . . .	42
4.10.2.7 RCC_selectSystemClock() . . . . .	42
4.10.2.8 RCC_setClockState() . . . . .	42
4.11 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED1.h File Reference . . . . .	43
4.11.1 Detailed Description . . . . .	43
4.11.2 Function Documentation . . . . .	43
4.11.2.1 SCHED_createTask() . . . . .	43
4.11.2.2 SCHED_init() . . . . .	44
4.11.2.3 SCHED_start() . . . . .	44
4.12 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED_CONF.h File Reference . . . . .	44
4.12.1 Detailed Description . . . . .	44
4.13 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch.h File Reference . . . . .	45

4.13.1 Detailed Description . . . . .	45
4.13.2 Function Documentation . . . . .	45
4.13.2.1 Switch_GetSwitchStatus() . . . . .	45
4.13.2.2 Function: Switch_GetSwitchStatus . . . . .	45
4.13.2.3 Function: Switch_GetSwitchStatus . . . . .	46
4.13.2.4 Switch_Init() . . . . .	46
4.13.2.5 Function: Switch_Init . . . . .	46
4.13.2.6 Function: Switch_Init . . . . .	46
4.13.2.7 Switch_Task() . . . . .	46
4.14 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch_Cfg.h File Reference . . . . .	47
4.14.1 Detailed Description . . . . .	47
4.15 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK.h File Reference . . . . .	47
4.15.1 Detailed Description . . . . .	48
4.15.2 Function Documentation . . . . .	48
4.15.2.1 SYSTICK_init() . . . . .	48
4.15.2.2 SYSTICK_setCallbackFcn() . . . . .	48
4.15.2.3 SYSTICK_setTime() . . . . .	49
4.15.2.4 SYSTICK_start() . . . . .	49
4.15.2.5 SYSTICK_stop() . . . . .	49
4.16 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK_CONF.h File Reference . . . . .	49
4.16.1 Detailed Description . . . . .	50
4.17 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Uart.h File Reference . . . . .	50
4.17.1 Detailed Description . . . . .	51
4.17.2 Function Documentation . . . . .	51
4.17.2.1 Uart_Init() . . . . .	51
4.17.2.2 Uart_Receive() . . . . .	52
4.17.2.3 Uart_Send() . . . . .	52
4.17.2.4 Uart_SetRxCb() . . . . .	53
4.17.2.5 Uart_SetTxCb() . . . . .	53
4.18 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/App.c File Reference . . . . .	54
4.18.1 Detailed Description . . . . .	55
4.18.2 Function Documentation . . . . .	55
4.18.2.1 APP_init() . . . . .	55
4.18.2.2 APP_receiveFcn() . . . . .	55
4.18.2.3 APP_sendTask() . . . . .	56
4.19 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Cld.c File Reference . . . . .	56
4.19.1 Detailed Description . . . . .	57
4.19.2 Function Documentation . . . . .	57
4.19.2.1 CLcd_ClearDisplay() . . . . .	58

4.19.2.2 CLcd_ConfigCursor()	58
4.19.2.3 CLcd_ConfigDisplay()	58
4.19.2.4 CLcd_GotoXY()	59
4.19.2.5 CLcd_Init()	59
4.19.2.6 CLcd_SetDoneNotification()	59
4.19.2.7 CLcd_Task()	60
4.19.2.8 CLcd_WriteString()	60
4.20 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/CLcd_Cfg.c File Reference	60
4.20.1 Detailed Description	61
4.20.2 Variable Documentation	61
4.20.2.1 CLcd_clcd	61
4.21 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Gpio.c File Reference	61
4.21.1 Detailed Description	62
4.21.2 Function Documentation	62
4.21.2.1 Gpio_InitPins()	62
4.21.2.2 Function: Gpio_InitPins	62
4.21.2.3 Gpio_ReadPin()	63
4.21.2.4 Function: Gpio_ReadPin	63
4.21.2.5 Gpio_WritePin()	63
4.21.2.6 Function: Gpio_WritePin	63
4.22 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/HUart.c File Reference	64
4.22.1 Detailed Description	65
4.22.2 Function Documentation	65
4.22.2.1 HUart_Config()	65
4.22.2.2 HUart_Init()	66
4.22.2.3 HUart_Receive()	66
4.22.2.4 HUart_Send()	67
4.22.2.5 HUart_SetModule()	67
4.22.2.6 HUart_SetRxCb()	67
4.22.2.7 HUart_SetTxCb()	68
4.23 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Led.c File Reference	68
4.23.1 Detailed Description	69
4.23.2 Function Documentation	69
4.23.2.1 Led_Init()	69
4.23.2.2 Function: Led_Init	69
4.23.2.3 Led_SetLedOff()	69
4.23.2.4 Function: Led_SetLedOff	69
4.23.2.5 Led_SetLedOn()	70
4.23.2.6 Function: Led_SetLedOn	70
4.23.2.7 Led_SetLedStatus()	70
4.23.2.8 Function: Led_SetLedStatus	70

4.24 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Led_Cfg.c File Reference . . . . .	71
4.24.1 Detailed Description . . . . .	71
4.24.2 Variable Documentation . . . . .	71
4.24.2.1 Led_ leds . . . . .	72
4.25 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/main.c File Reference . . . . .	72
4.25.1 Detailed Description . . . . .	72
4.26 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/NVIC.c File Reference . . . . .	73
4.26.1 Detailed Description . . . . .	74
4.26.2 Function Documentation . . . . .	74
4.26.2.1 NVIC_configurePriority() . . . . .	74
4.26.2.2 NVIC_controlAllPeripheral() . . . . .	74
4.26.2.3 NVIC_controlFault() . . . . .	75
4.26.2.4 NVIC_controlInterrupt() . . . . .	75
4.26.2.5 NVIC_controlPendingFlag() . . . . .	75
4.26.2.6 NVIC_filterInterrupts() . . . . .	76
4.26.2.7 NVIC_getActiveFlagStatus() . . . . .	76
4.26.2.8 NVIC_getPriority() . . . . .	76
4.27 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/SCHED.c File Reference . . . . .	77
4.27.1 Detailed Description . . . . .	77
4.27.2 Function Documentation . . . . .	77
4.27.2.1 SCHED_createTask() . . . . .	77
4.27.2.2 SCHED_init() . . . . .	78
4.27.2.3 SCHED_start() . . . . .	78
4.28 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Switch.c File Reference . . . . .	78
4.28.1 Detailed Description . . . . .	79
4.28.2 Function Documentation . . . . .	79
4.28.2.1 Switch_GetSwitchStatus() . . . . .	79
4.28.2.2 Function: Switch_GetSwitchStatus . . . . .	79
4.28.2.3 Switch_Init() . . . . .	79
4.28.2.4 Function: Switch_Init . . . . .	80
4.28.2.5 Switch_Task() . . . . .	80
4.29 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Switch_Cfg.c File Reference . . . . .	80
4.29.1 Detailed Description . . . . .	80
4.29.2 Variable Documentation . . . . .	80
4.29.2.1 Switch_switches . . . . .	81
4.30 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/SYSTICK.c File Reference . . . . .	81
4.30.1 Detailed Description . . . . .	82
4.30.2 Function Documentation . . . . .	82
4.30.2.1 SysTick_Handler() . . . . .	82
4.30.2.2 SYSTICK_init() . . . . .	82



---

4.30.2.3 SYSTICK_setCallbackFcn() . . . . .	82
4.30.2.4 SYSTICK_setTime() . . . . .	83
4.30.2.5 SYSTICK_start() . . . . .	83
4.30.2.6 SYSTICK_stop() . . . . .	83
4.31 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Uart.c File Reference	83
4.31.1 Detailed Description . . . . .	85
4.31.2 Function Documentation . . . . .	85
4.31.2.1 UART4_IRQHandler() . . . . .	85
4.31.2.2 UART5_IRQHandler() . . . . .	86
4.31.2.3 Uart_Init() . . . . .	86
4.31.2.4 Uart_Receive() . . . . .	86
4.31.2.5 Uart_Send() . . . . .	87
4.31.2.6 Uart_SetRxCb() . . . . .	87
4.31.2.7 Uart_SetTxCb() . . . . .	88
4.31.2.8 USART1_IRQHandler() . . . . .	88
4.31.2.9 USART2_IRQHandler() . . . . .	88
4.31.2.10 USART3_IRQHandler() . . . . .	88
4.31.3 Variable Documentation . . . . .	88
4.31.3.1 Uart_Address . . . . .	89
4.32 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Std_Types.h File Reference	89
4.32.1 Detailed Description . . . . .	90
<b>Index</b>	<b>91</b>



# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<b>clcd_t</b> . . . . .	5
<b>dataBuffer_t</b> . . . . .	5
<b>frame_t</b> . . . . .	
This is the frame type of size 4 byte . . . . .	6
<b>gpio_t</b> . . . . .	6
<b>hUartConfig_t</b> . . . . .	6
<b>led_t</b> . . . . .	7
<b>NVIC_regMap</b> . . . . .	7
<b>RCC_regMap</b> . . . . .	7
<b>switch_t</b> . . . . .	8
<b>SysTask</b> . . . . .	8
<b>SYSTICK_regMap</b> . . . . .	8
<b>Task</b> . . . . .	8
<b>uart_t</b> . . . . .	9



## Chapter 2

# File Index

### 2.1 File List

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

C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/ <b>Std_Types.h</b>	
Those are the standard types used in the drivers . . . . .	89
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>App.h</b>	
This is the user interface for the two counters application . . . . .	11
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>CLcd.h</b>	
This file is the user interface for the Character LCD Driver . . . . .	12
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Gpio.h</b>	
This file is to be used as an interface for the user of GPIO driver . . . . .	17
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>HRcc.h</b>	
This is the user interface for the RCC Handler . . . . .	20
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>HUart.h</b>	
This is the user interface for the uart handler . . . . .	22
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>HUart_Cfg.h</b>	
These are the user's configurations for the HUART driver . . . . .	26
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Led.h</b>	
This file is to be used as an interface for the user of the Led Handler . . . . .	27
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Led_Cfg.h</b>	
This file is to be given to the user to configure the Led Handler . . . . .	30
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>NVIC.h</b>	
This is the user interface for the NVIC driver . . . . .	30
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>RCC.h</b>	
This is the user interface for the RCC Driver . . . . .	37
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>SCHED1.h</b>	
This is the user interface for the scheduler . . . . .	43
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>SCHED_CONF.h</b>	
Those are the configurations for the Scheduler Driver . . . . .	44
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Switch.h</b>	
This file is to be used as an interface for the user of the Switch Handler . . . . .	45
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Switch_Cfg.h</b>	
This file is to be given to the user to configure the Switch Handler . . . . .	47
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>SYSTICK.h</b>	
This is the user interface for the Systick Driver . . . . .	47
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>SYSTICK_CONF.h</b>	
Those are the configurations for the Systick Driver . . . . .	49
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ <b>Uart.h</b>	
This is the user interface for the UART driver . . . . .	50

C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>App.c</b>	
This is an application for testing the UART and the LCD drivers . . . . .	54
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Clcd.c</b>	
This file contains the implementation for the Character LCD Driver . . . . .	56
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>CLcd_Cfg.c</b>	
The user's configurations . . . . .	60
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Gpio.c</b>	
This file is to be used as an implementation of the GPIO driver . . . . .	61
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>HUart.c</b>	
This is the implementation for the UART handler . . . . .	64
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Led.c</b>	
This file is to be used as an implementation for the Led Handler . . . . .	68
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Led_Cfg.c</b>	
Those are the User's configurations for the LED Driver . . . . .	71
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>main.c</b>	
Here is the implementation for the main function fo the application and also the tasks . . . . .	72
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>NVIC.c</b>	
This is the implementation for the NVIC Driver . . . . .	73
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>SCHED.c</b>	
This is the implementation of the scheduler . . . . .	77
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Switch.c</b>	
This file is to be used as an implementation for the Switch Handler . . . . .	78
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Switch_Cfg.c</b>	
This file is to be used as an implementation of the configurations the user configured in the <b>Switch_Cfg.h</b> (p. 47) . . . . .	80
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>SYSTICK.c</b>	
This is the SysTick driver implementation . . . . .	81
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ <b>Uart.c</b>	
This is the implementation for the UART driver . . . . .	83

## Chapter 3

# Data Structure Documentation

### 3.1 clcd\_t Struct Reference

#### Data Fields

- uint32\_t **enPin**
- uint32\_t **enPort**
- uint32\_t **rwPin**
- uint32\_t **rwPort**
- uint32\_t **rsPin**
- uint32\_t **rsPort**
- uint32\_t **dPin** [CLCD\_NUMBER\_OF\_DATA\_PINS]
- uint32\_t **dPort** [CLCD\_NUMBER\_OF\_DATA\_PINS]

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ **CLcd.h**

### 3.2 dataBuffer\_t Struct Reference

#### Data Fields

- uint8\_t \* **ptr**
- uint32\_t **pos**
- uint32\_t **size**
- uint8\_t **state**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **Uart.c**

### 3.3 frame\_t Union Reference

This is the frame type of size 4 byte.

#### Data Fields

- uint8\_t **data** [4]
- uint32\_t **fullFrame**

#### 3.3.1 Detailed Description

This is the frame type of size 4 byte.

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **App.c**

### 3.4 gpio\_t Struct Reference

#### Data Fields

- uint32\_t **pins**
- uint32\_t **speed**
- uint32\_t **mode**
- uint32\_t **port**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ **Gpio.h**

### 3.5 hUartConfig\_t Struct Reference

#### Data Fields

- uint32\_t **baudRate**
- uint32\_t **stopBits**
- uint32\_t **parity**
- uint32\_t **flowControl**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **HUart.c**



## 3.6 led\_t Struct Reference

### Data Fields

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

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ **Led.h**

## 3.7 NVIC\_regMap Struct Reference

### Data Fields

- u32 **ISER** [3]
- u32 **RESERVED0** [29]
- u32 **ICER** [3]
- u32 **RESERVED1** [29]
- u32 **ISPR** [3]
- u32 **RESERVED2** [29]
- u32 **ICPR** [3]
- u32 **RESERVED3** [29]
- u32 **IABR** [3]
- u32 **RESERVED4** [29]
- u32 **IPR** [21]

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **NVIC.c**

## 3.8 RCC\_regMap Struct Reference

### Data Fields

- u32 **RCC\_CR**
- u32 **RCC\_CFGR**
- u32 **RCC\_CIR**
- u32 **RCC\_APB2RSTR**
- u32 **RCC\_APB1RSTR**
- u32 **RCC\_AHBENR**
- u32 **RCC\_APB2ENR**
- u32 **RCC\_APB1ENR**
- u32 **RCC\_BDCR**
- u32 **RCC\_CSR**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/RCC.c

### 3.9 switch\_t Struct Reference

#### Data Fields

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

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ **Switch.h**

### 3.10 SysTask Struct Reference

#### Data Fields

- Task \* **appTask**
- u32 **RemainToExec**
- u32 **periodicTimeTicks**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **SCHED.c**

### 3.11 SYSTICK\_regMap Struct Reference

#### Data Fields

- u32 **CTRL**
- u32 **LOAD**
- u32 **VAL**
- u32 **CALIB**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **SYSTICK.c**

### 3.12 Task Struct Reference

#### Data Fields

- taskRunnable **runnable**
- u32 **periodicTime**
- u32 **priority**

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/ **SCHED1.h**

## 3.13 uart\_t Struct Reference

### Data Fields

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

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

- C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/ **Uart.c**



## Chapter 4

# File Documentation

### 4.1 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/App.h File Reference

This is the user interface for the two counters application.

#### Functions

- Std\_ReturnType **APP\_init** (void)  
*This is the initialization for the two counter application.*
- void **APP\_sendTask** (void)  
*The free running task that comes every 1 milli second.*
- void **APP\_receiveFcn** (void)  
*The receive function that will be called after each received frame.*

#### 4.1.1 Detailed Description

This is the user interface for the two counters application.

##### Author

Mariam Mohammed

##### Version

0.1

##### Date

2020-03-29

##### Copyright

Copyright (c) 2020

## 4.1.2 Function Documentation

### 4.1.2.1 APP\_init()

```
Std_ReturnType APP_init (
    void )
```

This is the initialization for the two counter application.

#### Returns

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

### 4.1.2.2 APP\_receiveFcn()

```
void APP_receiveFcn (
    void )
```

The receive function that will be called after each received frame.

### 4.1.2.3 APP\_sendTask()

```
void APP_sendTask (
    void )
```

The free running task that comes every 1 milli second.

## 4.2 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/CLcd.h File Reference

This file is the user interface for the Character LCD Driver.

### Data Structures

- struct **clcd\_t**

## Macros

- `#define CLCD_NUMBER_OF_DATA_PINS 4`
- `#define CLCD_TWO_LINES 0x8`
- `#define CLCD_ONE_LINE 0x0`
- `#define CLCD_DISP_ON 0x4`
- `#define CLCD_DISP_OFF 0x0`
- `#define CLCD_CURSOR_ON 0x2`
- `#define CLCD_CURSOR_OFF 0x0`
- `#define CLCD_BLINKING_ON 0x1`
- `#define CLCD_BLINKING_OFF 0x0`

## Typedefs

- `typedef void(* lcdCb_t) (void)`

## Functions

- Std\_ReturnType **CLcd\_Init** (uint8\_t nLines, uint8\_t cursor, uint8\_t blink)  
*The Character LCD initialization.*
- Std\_ReturnType **CLcd\_WriteString** (uint8\_t \*str, uint8\_t x, uint8\_t y)  
*Writes a string on a specific location on the lcd display.*
- Std\_ReturnType **CLcd\_ClearDisplay** (void)  
*Clears the display.*
- Std\_ReturnType **CLcd\_GotoXY** (uint8\_t x, uint8\_t y)  
*jumps to a specific location on the lcd display*
- Std\_ReturnType **CLcd\_ConfigCursor** (uint8\_t cursor, uint8\_t blink)  
*Configures the cursor options.*
- Std\_ReturnType **CLcd\_ConfigDisplay** (uint8\_t disp)  
*Sets the display on and off.*
- Std\_ReturnType **CLcd\_SetDoneNotification** (lcdCb\_t cb)  
*Sets the callback function executed when done.*
- void **CLcd\_Task** (void)  
*The running task that have to come every 1 milli second.*

### 4.2.1 Detailed Description

This file is the user interface for the Character LCD Driver.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

## 4.2.2 Function Documentation

### 4.2.2.1 CLcd\_ClearDisplay()

```
Std_ReturnType CLcd_ClearDisplay (
    void )
```

Clears the display.

#### Returns

Std\_ReturnType E\_OK : If the clear operation started successfully E\_NOT\_OK : If the clear operation is not able to start right now

### 4.2.2.2 CLcd\_ConfigCursor()

```
Std_ReturnType CLcd_ConfigCursor (
    uint8_t cursor,
    uint8_t blink )
```

Configures the cursor options.

#### Parameters

<i>cursor</i>	The State of the cursor (Visible or not) CLCD_CURSOR_ON CLCD_CURSOR_OFF
<i>blink</i>	The blinking option (no/off) CLCD_BLINKING_ON CLCD_BLINKING_OFF

#### Returns

Std\_ReturnType E\_OK : If the configuration started successfully E\_NOT\_OK : If the configuration is not able to start right now

### 4.2.2.3 CLcd\_ConfigDisplay()

```
Std_ReturnType CLcd_ConfigDisplay (
    uint8_t disp )
```

Sets the display on and off.

#### Parameters

<i>disp</i>	the display state CLCD_DISP_ON CLCD_DISP_OFF
-------------	--



**Returns**

Std\_ReturnType E\_OK : If the configuration started successfully E\_NOT\_OK : If the configuration is not able to start right now

**4.2.2.4 CLcd\_GotoXY()**

```
Std_ReturnType CLcd_GotoXY (
    uint8_t x,
    uint8_t y )
```

jumps to a specific location on the lcd display

**Parameters**

<i>x</i>	the location on the x-axis
<i>y</i>	the location on the y-axis

**Returns**

Std\_ReturnType E\_OK : If the goto operation started successfully E\_NOT\_OK : If the goto operation is not able to start right now

**4.2.2.5 CLcd\_Init()**

```
Std_ReturnType CLcd_Init (
    uint8_t nLines,
    uint8_t cursor,
    uint8_t blink )
```

The Character LCD initialization.

**Parameters**

<i>nLines</i>	The number of lines on display CLCD_TWO_LINES : Two lines display CLCD_ONE_LINE : One line display
<i>cursor</i>	The State of the cursor (Visible or not) CLCD_CURSOR_ON CLCD_CURSOR_OFF
<i>blink</i>	The blinking option (no/off) CLCD_BLINKING_ON CLCD_BLINKING_OFF

**Returns**

Std\_ReturnType E\_OK : If the initialization started successfully E\_NOT\_OK : If the initialization is not able to start right now

#### 4.2.2.6 CLcd\_SetDoneNotification()

```
Std_ReturnType CLcd_SetDoneNotification (
    lcdCb_t cb )
```

Sets the callback function executed when done.

##### Parameters

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

##### Returns

Std\_ReturnType

#### 4.2.2.7 CLcd\_Task()

```
void CLcd_Task (
    void )
```

The running task that have to come every 1 milli second.

#### 4.2.2.8 CLcd\_WriteString()

```
Std_ReturnType CLcd_WriteString (
    uint8_t * str,
    uint8_t x,
    uint8_t y )
```

Writes a string on a specific location on the lcd display.

##### Parameters

<i>str</i>	the string to write
<i>x</i>	the location on the x-axis
<i>y</i>	the location on the y-axis

##### Returns

Std\_ReturnType E\_OK : If the writing started successfully E\_NOT\_OK : If the write operation is not able to start right now

## 4.3 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Gpio.h File Reference

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

### Data Structures

- struct `gpio_t`

### Macros

- `#define GPIO_PIN_SET 0`
- `#define GPIO_PIN_RESET !GPIO_PIN_SET`
- `#define GPIO_PIN_0 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)*

### 4.3.1 Detailed Description

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

#### Author

Mark Attia

#### Date

February 6, 2020

### 4.3.2 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 <b>gpio_t</b> (p. 6) 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 Function: Gpio\_InitPins

### Parameters

<i>gpio</i>	An object of type <b>gpio_t</b> (p. 6) 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

#### 4.3.2.4 Gpio\_ReadPin()

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

Reads a value to a pin(0/1)

#### 4.3.2.5 Function: Gpio\_ReadPin

### Parameters

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

### Returns

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

#### 4.3.2.6 Function: Gpio\_ReadPin

### Parameters

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

### Returns

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

#### 4.3.2.7 Gpio\_WritePin()

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

Write a value to a pin(0/1)

#### 4.3.2.8 Function: Gpio\_WritePin

##### Parameters

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

##### Returns

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

#### 4.3.2.9 Function: Gpio\_WritePin

##### Parameters

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

##### Returns

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

## 4.4 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/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.*

### 4.4.1 Detailed Description

This is the user interface for the RCC Handler.

This is implementation for the RCC Handler.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-24

#### Copyright

Copyright (c) 2020

### 4.4.2 Function Documentation

#### 4.4.2.1 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 GPIO_PORTX : The pin number you want to configure
-------------	---

#### Returns

Std\_ReturnType

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

#### 4.4.2.2 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

## 4.5 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUart.h File Reference

This is the user interface for the uart handler.

### Macros

- #define **HUART\_MODULE\_1** 0
- #define **HUART\_MODULE\_2** 1
- #define **HUART\_MODULE\_3** 2
- #define **HUART\_MODULE\_4** 3
- #define **HUART\_MODULE\_5** 4
- #define **HUART\_ODD\_PARITY** 0x00000200
- #define **HUART\_EVEN\_PARITY** 0x00000000
- #define **HUART\_NO\_PARITY** 0xFFFFFBFF
- #define **HUART\_STOP\_ONE\_BIT** 0x00000000
- #define **HUART\_STOP\_TWO\_BITS** 0x00003000
- #define **HUART\_FLOW\_CONTROL\_EN** 0x00000100
- #define **HUART\_FLOW\_CONTROL\_DIS** 0x00000000

### Typedefs

- typedef void(\* **hUartTxCb\_t**) (void)
- typedef void(\* **hUartRxCb\_t**) (void)

### Functions

- Std\_ReturnType **HUart\_Init** (void)  
*Initializes the UART Module.*
- Std\_ReturnType **HUart\_Config** (uint32\_t baudRate, uint32\_t stopBits, uint32\_t parity, uint32\_t flowControl)  
*Sets configurations for the UART module \*The UART must be initialized after setting configurations to apply the changes.*
- Std\_ReturnType **HUart\_SetModule** (uint8\_t uartModule)  
*Sets the module that you will be using.*
- Std\_ReturnType **HUart\_Send** (uint8\_t \*data, uint16\_t length)  
*Sends data through the UART.*
- Std\_ReturnType **HUart\_Receive** (uint8\_t \*data, uint16\_t length)  
*Receives data through the UART.*
- Std\_ReturnType **HUart\_SetRxCb** (hUartRxCb\_t func)  
*Sets the callback function that will be called when receive is completed.*
- Std\_ReturnType **HUart\_SetTxCb** (hUartTxCb\_t func)  
*Sets the callback function that will be called when transmission is completed.*



### 4.5.1 Detailed Description

This is the user interface for the uart handler.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-29

#### Copyright

Copyright (c) 2020

### 4.5.2 Function Documentation

#### 4.5.2.1 HUart\_Config()

```
Std_ReturnType HUart_Config (
    uint32_t baudRate,
    uint32_t stopBits,
    uint32_t parity,
    uint32_t flowControl )
```

Sets configurations for the UART module \*The UART must be initialized after setting configurations to apply the changes.

#### Parameters

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits HUART_ONE_STOP_BIT HUART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission HUART_ODD_PARITY HUART_EVEN_PARITY HUART_NO_PARITY
<i>flowControl</i>	the flow control HUART_FLOW_CONTROL_EN HUART_FLOW_CONTROL_DIS

#### Returns

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

#### 4.5.2.2 HUart\_Init()

```
Std_ReturnType HUart_Init (
    void )
```

Initializes the UART Module.

##### Returns

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

#### 4.5.2.3 HUart\_Receive()

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

Receives data through the UART.

##### Parameters

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

##### Returns

Std\_ReturnType A Status E\_OK: If the driver is ready to receive E\_NOT\_OK: If the driver can't receive data right now

#### 4.5.2.4 HUart\_Send()

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

Sends data through the UART.

##### Parameters

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

#### Returns

Std\_ReturnType A Status E\_OK: If the driver is ready to send E\_NOT\_OK: If the driver can't send data right now

#### 4.5.2.5 HUart\_SetModule()

```
Std_ReturnType HUart_SetModule (
    uint8_t uartModule )
```

Sets the module that you will be using.

#### Parameters

<i>uartModule</i>	The UART module HUART_MODULE_1 HUART_MODULE_2 HUART_MODULE_3 HUART_MODULE_4 HUART_MODULE_5
-------------------	--

#### Returns

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

#### 4.5.2.6 HUart\_SetRxCb()

```
Std_ReturnType HUart_SetRxCb (
    hUartRxCb_t func )
```

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

#### Parameters

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

#### Returns

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

#### 4.5.2.7 HUart\_SetTxCb()

```
Std_ReturnType HUart_SetTxCb (
    hUartTxCb_t func )
```

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

#### Parameters

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

#### Returns

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

## 4.6 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUart\_Cfg.h File Reference

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

### Macros

- #define **HUART\_SYSTEM\_CLK** 8000000
- #define **HUART\_DEFAULT\_BAUDRATE** 9600
- #define **HUART\_DEFAULT\_STOP\_BITS** HUART\_STOP\_ONE\_BIT
- #define **HUART\_DEFAULT\_PARITY** HUART\_NO\_PARITY
- #define **HUART\_DEFAULT\_FLOW\_CONTROL** HUART\_FLOW\_CONTROL\_DIS
- #define **HUART\_DEFAULT\_MODULE** HUART\_MODULE\_1

### 4.6.1 Detailed Description

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

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-27

#### Copyright

Copyright (c) 2020

## 4.7 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led.h File Reference

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

### Data Structures

- struct **led\_t**

### Macros

- #define **LED\_ON** 0
- #define **LED\_OFF** !LED\_ON

### Functions

- Std\_ReturnType **Led\_Init** (void)  
*Initializes GPIOs for the LEDs.*
- Std\_ReturnType **Led\_SetLedOn** (uint8\_t ledName)  
*Sets the Led on.*
- Std\_ReturnType **Led\_SetLedOff** (uint8\_t ledName)  
*Sets the Led off.*
- Std\_ReturnType **Led\_SetLedStatus** (uint8\_t ledName, uint8\_t status)  
*Sets the Led off.*

#### 4.7.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

#### 4.7.2 Function Documentation

#### 4.7.2.1 Led\_Init()

```
Std_ReturnType Led_Init (
    void )
```

Initializes GPIOs for the LEDs.

#### 4.7.2.2 Function: Led\_Init

##### Returns

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

#### 4.7.2.3 Function: Led\_Init

##### Returns

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

#### 4.7.2.4 Led\_SetLedOff()

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

Sets the Led off.

#### 4.7.2.5 Function: Led\_SetLedOff

##### Parameters

<i>ledName</i>	The name of the LED
----------------	---------------------

##### Returns

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

#### 4.7.2.6 Function: Led\_SetLedOff

##### Parameters

<i>ledName</i>	The name of the LED
----------------	---------------------

##### Returns

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

#### 4.7.2.7 Led\_SetLedOn()

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

Sets the Led on.

#### 4.7.2.8 Function: Led\_SetLedOn

##### Parameters

<i>ledName</i>	The name of the LED
----------------	---------------------

##### Returns

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

#### 4.7.2.9 Function: Led\_SetLedOn

##### Parameters

<i>ledName</i>	The name of the LED
----------------	---------------------

##### Returns

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

#### 4.7.2.10 Led\_SetLedStatus()

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

Sets the Led off.

#### 4.7.2.11 Function: Led\_SetLedStatus

##### Parameters

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

**Returns**

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

**4.7.2.12 Function: Led\_SetLedStatus****Parameters**

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

**Returns**

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

## 4.8 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led\_Cfg.h File Reference

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

**Macros**

- #define LED\_NUMBER\_OF\_LEDS 1
- #define LED\_1 0
- #define LED\_2 1
- #define LED\_3 2

**4.8.1 Detailed Description**

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

**Author**

Mark Attia

**Date**

January 22, 2020

## 4.9 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/NVIC.h File Reference

This is the user interface for the NVIC driver.



## Macros

- #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
- `#define NVIC_DISABLE` 0
- `#define NVIC_ENABLE` 1
- `#define NVIC_RESET` 0
- `#define NVIC_SET` 1

## Functions

- void **NVIC\_controlInterrupt** (u8 interruptNum, u8 status)  
*Sets and resets the interrupts.*
- void **NVIC\_controlPendingFlag** (u8 interruptNum, u8 val)  
*Sets and resets The pending flag.*
- u8 **NVIC\_getActiveFlagStatus** (u8 interruptNum)  
*Gets the active flag state.*
- void **NVIC\_configurePriority** (u8 interruptNum, u8 priority)  
*Configures the periority of the interrupt.*
- u8 **NVIC\_getPriority** (u8 interruptNum)  
*Gets the priority of the interrupt.*
- void **NVIC\_controlAllPeripheral** (u8 status)  
*Controls All of the prephirals.*
- void **NVIC\_controlFault** (u8 status)  
*Controls The Fault flag.*
- void **NVIC\_filterInterrupts** (u8 priority)  
*Filters the interrupt.*

### 4.9.1 Detailed Description

This is the user interface for the NVIC driver.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-29

#### Copyright

Copyright (c) 2020

## 4.9.2 Macro Definition Documentation

### 4.9.2.1 NVIC\_DISABLE

```
#define NVIC_DISABLE 0
```

status

### 4.9.2.2 NVIC\_IRQNUM\_WWDG

```
#define NVIC_IRQNUM_WWDG 0
```

interruptNum

### 4.9.2.3 NVIC\_RESET

```
#define NVIC_RESET 0
```

val

## 4.9.3 Function Documentation

### 4.9.3.1 NVIC\_configurePriority()

```
void NVIC_configurePriority (
    u8 interruptNum,
    u8 priority )
```

Configures the periority of the interrupt.

#### Parameters

<i>interruptNum</i>	the number of the interrupt
<i>priority</i>	The periority

### 4.9.3.2 NVIC\_controlAllPeripheral()

```
void NVIC_controlAllPeripheral (
    u8 status )
```

Controls All of the prepirals.

## Parameters

<i>status</i>	NVIC_ENABLE NVIC_DISABLE
---------------	--------------------------

**4.9.3.3 NVIC\_controlFault()**

```
void NVIC_controlFault (
    u8 status )
```

Controls The Fault flag.

## Parameters

<i>status</i>	NVIC_ENABLE NVIC_DISABLE
---------------	--------------------------

**4.9.3.4 NVIC\_controlInterrupt()**

```
void NVIC_controlInterrupt (
    u8 interruptNum,
    u8 status )
```

Sets and resets the interrupts.

## Parameters

<i>interruptNum</i>	The Interrupt number
<i>status</i>	The state NVIC_DISABLE NVIC_ENABLE

**4.9.3.5 NVIC\_controlPendingFlag()**

```
void NVIC_controlPendingFlag (
    u8 interruptNum,
    u8 val )
```

Sets and resets The pending flag.

## Parameters

<i>interruptNum</i>	The Interrupt number
<i>val</i>	the value to be set NVIC_RESET NVIC_SET

#### 4.9.3.6 NVIC\_filterInterrupts()

```
void NVIC_filterInterrupts (
    u8 priority )
```

Filters the interrupt.

##### Parameters

<i>priority</i>	the priority of the interrupt
-----------------	-------------------------------

#### 4.9.3.7 NVIC\_getActiveFlagStatus()

```
u8 NVIC_getActiveFlagStatus (
    u8 interruptNum )
```

Gets the active flag state.

##### Parameters

<i>interruptNum</i>	the number of the interrupt
---------------------	-----------------------------

##### Returns

u8

#### 4.9.3.8 NVIC\_getPriority()

```
u8 NVIC_getPriority (
    u8 interruptNum )
```

Gets the priority of the interrupt.

##### Parameters

<i>interruptNum</i>	the number of the interrupt
---------------------	-----------------------------

##### Returns

u8

## 4.10 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/RCC.h File Reference

This is the user interface for the RCC Driver.

### Macros

- `#define ENABLE 1`
- `#define DISABLE 0`
- `#define RCC_DMA1 0x00000001`
- `#define RCC_DMA2 0x00000002`
- `#define RCC_SRAM 0x00000004`
- `#define RCC_FLITF 0x00000010`
- `#define RCC_CRC 0x00000040`
- `#define RCC_FSMC 0x00000100`
- `#define RCC_SDIO 0x00000400`
- `#define RCC_AFIO 0x00000001`
- `#define RCC_GPIOA 0x00000004`
- `#define RCC_GPIOB 0x00000008`
- `#define RCC_GPIOC 0x00000010`
- `#define RCC_GPIOD 0x00000020`
- `#define RCC_GPIOE 0x00000040`
- `#define RCC_GPIOF 0x00000080`
- `#define RCC_GPIOG 0x00000100`
- `#define RCC_ADC1 0x00000200`
- `#define RCC_ADC2 0x00000400`
- `#define RCC_TIM1 0x00000800`
- `#define RCC_SPI1 0x00001000`
- `#define RCC_TIM8 0x00002000`
- `#define RCC_USART1 0x00004000`
- `#define RCC_ADC3 0x00008000`
- `#define RCC_TIM9 0x00080000`
- `#define RCC_TIM10 0x00100000`
- `#define RCC_TIM11 0x00200000`
- `#define RCC_TIM2 0x00000001`
- `#define RCC_TIM3 0x00000002`
- `#define RCC_TIM4 0x00000004`
- `#define RCC_TIM5 0x00000008`
- `#define RCC_TIM6 0x00000010`
- `#define RCC_TIM7 0x00000020`
- `#define RCC_TIM12 0x00000040`
- `#define RCC_TIM13 0x00000080`
- `#define RCC_TIM14 0x00000100`
- `#define RCC_WWDG 0x00000800`
- `#define RCC_SPI2_I2S 0x00004000`
- `#define RCC_SPI3_I2S 0x00008000`
- `#define RCC_USART2 0x00020000`
- `#define RCC_USART3 0x00040000`
- `#define RCC_UART4 0x00080000`
- `#define RCC_UART5 0x00100000`
- `#define RCC_I2C1 0x00200000`

- `#define RCC_I2C2 0x00400000`
- `#define RCC_USB 0x00800000`
- `#define RCC_CAN 0x02000000`
- `#define RCC_BKP 0x08000000`
- `#define RCC_PWR 0x10000000`
- `#define RCC_DAC 0x20000000`
- `#define RCC_sysClk_HSI 0x00000000`
- `#define RCC_sysClk_HSE 0x00000001`
- `#define RCC_sysClk_PLL 0x00000002`
- `#define RCC_OFF 0`
- `#define RCC_ON 1`
- `#define RCC_HSI_ON 0x00000001`
- `#define RCC_HSE_ON 0x00010000`
- `#define RCC_PLL_ON 0x01000000`
- `#define RCC_PLLSRC_HSI 0x00000000`
- `#define RCC_PLLSRC_HSE 0x00010000`
- `#define RCC_PLLSRC_HSE_DIV_2 0x00030000`
- `#define RCC_PLLMUL_SPEED_2 0x00000000`
- `#define RCC_PLLMUL_SPEED_3 0x00040000`
- `#define RCC_PLLMUL_SPEED_4 0x00080000`
- `#define RCC_PLLMUL_SPEED_5 0x000C0000`
- `#define RCC_PLLMUL_SPEED_6 0x00100000`
- `#define RCC_PLLMUL_SPEED_7 0x00140000`
- `#define RCC_PLLMUL_SPEED_8 0x00180000`
- `#define RCC_PLLMUL_SPEED_9 0x001C0000`
- `#define RCC_PLLMUL_SPEED_10 0x00200000`
- `#define RCC_PLLMUL_SPEED_11 0x00240000`
- `#define RCC_PLLMUL_SPEED_12 0x00280000`
- `#define RCC_PLLMUL_SPEED_13 0x002C0000`
- `#define RCC_PLLMUL_SPEED_14 0x00300000`
- `#define RCC_PLLMUL_SPEED_15 0x00340000`
- `#define RCC_PLLMUL_SPEED_16 0x00380000`
- `#define RCC_USB_PRESCALER 0x00400000`
- `#define RCC_ADC_PRESCALER 0x0000C000`
- `#define RCC_APB2_PRESCALER 0x00003800`
- `#define RCC_APB1_PRESCALER 0x00000700`
- `#define RCC_AHB_PRESCALER 0x000000F0`
- `#define RCC_USB_DIVIDED 0x00000000`
- `#define RCC_USB_NDIVIDED 0x00400000`
- `#define RCC_ADC_DIV_2 0x00000000`
- `#define RCC_ADC_DIV_4 0x00004000`
- `#define RCC_ADC_DIV_6 0x00008000`
- `#define RCC_ADC_DIV_8 0x0000C000`
- `#define RCC_APB2_NDIVIDED 0x00000000`
- `#define RCC_APB2_DIV_2 0x00002000`
- `#define RCC_APB2_DIV_4 0x00002800`
- `#define RCC_APB2_DIV_8 0x00003000`
- `#define RCC_APB2_DIV_16 0x00003800`
- `#define RCC_APB1_NDIVIDED 0x00000000`
- `#define RCC_APB1_DIV_2 0x00000400`
- `#define RCC_APB1_DIV_4 0x00000500`
- `#define RCC_APB1_DIV_8 0x00000600`
- `#define RCC_APB1_DIV_16 0x00000700`
- `#define RCC_AHB_NDIVIDED 0x00000000`
- `#define RCC_AHB_DIV_2 0x00000080`



- `#define RCC_AHB_DIV_4 0x00000090`
- `#define RCC_AHB_DIV_8 0x000000A0`
- `#define RCC_AHB_DIV_16 0x000000B0`
- `#define RCC_AHB_DIV_64 0x000000C0`
- `#define RCC_AHB_DIV_128 0x000000D0`
- `#define RCC_AHB_DIV_256 0x000000E0`
- `#define RCC_AHB_DIV_512 0x000000F0`
- `#define RCC_MCO_NOSRC 0x00000000`
- `#define RCC_MCO_SYSCLK 0x04000000`
- `#define RCC_MCO_HSI 0x05000000`
- `#define RCC_MCO_HSE 0x06000000`
- `#define RCC_MCO_PLL 0x07000000`

## Functions

- void **RCC\_controlAHBPeripheral** (u32 peripheralNum, u32 status)
- void **RCC\_controlAPB2Peripheral** (u32 peripheralNum, u32 status)
- void **RCC\_controlAPB1Peripheral** (u32 peripheralNum, u32 status)
- void **RCC\_selectSystemClock** (u32 sysClkNum)
- void **RCC\_setClockState** (u32 clkNum, u32 status)
- void **RCC\_configurePLL** (u32 pllSrc, u32 speedMul)
- void **RCC\_configurePrescalers** (u32 target, u32 preValue)
- void **RCC\_configureMCO** (u32 clkNum)

### 4.10.1 Detailed Description

This is the user interface for the RCC Driver.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-28

#### Copyright

Copyright (c) 2020

### 4.10.2 Function Documentation

#### 4.10.2.1 RCC\_configureMCO()

```
void RCC_configureMCO (
    u32 clkNum )
```

Function Name: RCC\_configureMCO Usage: configure MCO source Function Arguments: u32 clkNum - takes one of these values RCC\_MCO\_NOSRC RCC\_MCO\_SYSCLK RCC\_MCO\_HSI  
RCC\_MCO\_HSE  
RCC\_MCO\_PLL

#### 4.10.2.2 RCC\_configurePLL()

```
void RCC_configurePLL (
    u32 pllSrc,
    u32 speedMul )
```

Function Name: RCC\_configurePLL Usage: configure PLL source & speed Function Arguments: u32 pllSrc - takes one of these values RCC\_PLLSRC\_HSI  
RCC\_PLLSRC\_HSE  
RCC\_PLLSRC\_HSE\_DIV\_2

u32 speedMul - takes one of these values RCC\_PLLMUL\_SPEED\_2 RCC\_PLLMUL\_SPEED\_3 RCC\_PLLMUL\_SPEED\_4 RCC\_PLLMUL\_SPEED\_5 RCC\_PLLMUL\_SPEED\_6 RCC\_PLLMUL\_SPEED\_7 RCC\_PLLMUL\_SPEED\_8 RCC\_PLLMUL\_SPEED\_9 RCC\_PLLMUL\_SPEED\_10 RCC\_PLLMUL\_SPEED\_11 RCC\_PLLMUL\_SPEED\_12 RCC\_PLLMUL\_SPEED\_13 RCC\_PLLMUL\_SPEED\_14 RCC\_PLLMUL\_SPEED\_15 RCC\_PLLMUL\_SPEED\_16

#### 4.10.2.3 RCC\_configurePrescalers()

```
void RCC_configurePrescalers (
    u32 target,
    u32 preValue )
```

Function Name: RCC\_configurePrescalers Usage: configure prescalers for a specific target Function Arguments: u32 target - takes one of these values RCC\_USB\_PRESCALER RCC\_ADC\_PRESCALER RCC\_APB2\_PRESCALER RCC\_APB1\_PRESCALER RCC\_AHB\_PRESCALER

u32 preValue - takes one of these values RCC\_USB\_DIVIDED  
RCC\_USB\_NDIVIDED RCC\_ADC\_DIV\_2  
RCC\_ADC\_DIV\_4  
RCC\_ADC\_DIV\_6  
RCC\_ADC\_DIV\_6  
RCC\_APB2\_NDIVIDED RCC\_APB2\_DIV\_2  
RCC\_APB2\_DIV\_4  
RCC\_APB2\_DIV\_8  
RCC\_APB2\_DIV\_16  
RCC\_APB1\_NDIVIDED RCC\_APB1\_DIV\_2  
RCC\_APB1\_DIV\_4  
RCC\_APB1\_DIV\_8  
RCC\_APB1\_DIV\_16  
RCC\_AHB\_NDIVIDED RCC\_AHB\_DIV\_2  
RCC\_AHB\_DIV\_4  
RCC\_AHB\_DIV\_8

RCC\_AHB\_DIV\_16  
 RCC\_AHB\_DIV\_64  
 RCC\_AHB\_DIV\_128 RCC\_AHB\_DIV\_256 RCC\_AHB\_DIV\_512

Function Name: RCC\_configurePrescalers Usage: configure prescalers for a specific target Function Arguments: u32 target - takes one of these values RCC\_USB\_PRESCALER RCC\_ADC\_PRESCALER RCC\_APB2\_PRESCALER RCC\_APB1\_PRESCALER RCC\_AHB\_PRESCALER

u32 preValue - takes one of these values RCC\_USB\_DIVIDED  
 RCC\_USB\_NDIVIDED RCC\_ADC\_DIV\_2  
 RCC\_ADC\_DIV\_4  
 RCC\_ADC\_DIV\_6  
 RCC\_ADC\_DIV\_6  
 RCC\_APB2\_NDIVIDED RCC\_APB2\_DIV\_2  
 RCC\_APB2\_DIV\_4  
 RCC\_APB2\_DIV\_8  
 RCC\_APB2\_DIV\_16  
 RCC\_APB1\_NDIVIDED RCC\_APB1\_DIV\_2  
 RCC\_APB1\_DIV\_4  
 RCC\_APB1\_DIV\_8  
 RCC\_APB1\_DIV\_16

#### 4.10.2.4 RCC\_controlAHBPeripheral()

```
void RCC_controlAHBPeripheral (
    u32 peripheralNum,
    u32 status )
```

Function Name: RCC\_controlAHBPeripheral Usage: Disable/Enable peripherals on AHB Function Arguments: u32 peripheralNum - takes one of these values RCC\_DMA1 RCC\_DMA2 RCC\_SRAM RCC\_FLITF RCC\_CRC RCC\_FSMC RCC\_SDIO

u32 status - takes one of these values ENABLE DISABLE

#### 4.10.2.5 RCC\_controlAPB1Peripheral()

```
void RCC_controlAPB1Peripheral (
    u32 peripheralNum,
    u32 status )
```

Function Name: RCC\_controlAPB1Peripheral Usage: Disable/Enable peripherals on APB1 Function Arguments: u32 peripheralNum - takes one of these values RCC\_TIM2

RCC\_TIM3  
 RCC\_TIM4  
 RCC\_TIM5  
 RCC\_TIM6  
 RCC\_TIM7  
 RCC\_TIM12  
 RCC\_TIM13  
 RCC\_TIM14  
 RCC\_WWDG  
 RCC\_SPI2\_I2S RCC\_SPI3\_I2S RCC\_USART2  
 RCC\_USART3  
 RCC\_UART4

RCC\_UART5  
RCC\_I2C1  
RCC\_I2C2  
RCC\_USB  
RCC\_CAN  
RCC\_BKP  
RCC\_PWR  
RCC\_DAC

u32 status - takes one of these values ENABLE DISABLE

#### 4.10.2.6 RCC\_controlAPB2Peripheral()

```
void RCC_controlAPB2Peripheral (
    u32 peripheralNum,
    u32 status )
```

Function Name: RCC\_controlAPB2Peripheral Usage: Disable/Enable peripherals on APB2 Function Arguments:

u32 peripheralNum - takes one of these values RCC\_AFIO

RCC\_GPIOA RCC\_GPIOB RCC\_GPIOC RCC\_GPIOD RCC\_GPIOE RCC\_GPIOF RCC\_GPIOG RCC\_ADC1  
RCC\_ADC2

RCC\_TIM1

RCC\_SPI1

RCC\_TIM8

RCC\_USART1 RCC\_ADC3

RCC\_TIM9

RCC\_TIM10 RCC\_TIM11

u32 status - takes one of these values ENABLE DISABLE

#### 4.10.2.7 RCC\_selectSystemClock()

```
void RCC_selectSystemClock (
    u32 sysClkNum )
```

Function Name: RCC\_selectSystemClock Usage: Select clock source for the system Function Arguments: u32

sysClkNum - takes one of these values RCC\_sysClk\_HSI RCC\_sysClk\_HSE RCC\_sysClk\_PLL

#### 4.10.2.8 RCC\_setClockState()

```
void RCC_setClockState (
    u32 clkNum,
    u32 status )
```

Function Name: RCC\_setClockState Usage: RCC\_ON/RCC\_OFF a clock source Function Arguments: u32

clkNum - takes one of these values RCC\_HSI\_ON RCC\_HSE\_ON RCC\_PLL\_ON

u32 status - takes one of these values RCC\_ON RCC\_OFF

## 4.11 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED1.h File Reference

This is the user interface for the scheduler.

### Data Structures

- struct **Task**

### Typedefs

- typedef void(\* **taskRunnable**) (void)

### Functions

- void **SCHED\_init** (void)  
*The initialization function.*
- void **SCHED\_createTask** ( **Task** \*appTask)  
*This function creates a task dynamically in the run time.*
- void **SCHED\_start** (void)  
*Starts The running scheduel.*

#### 4.11.1 Detailed Description

This is the user interface for the scheduler.

##### Author

Mariam Mohammed

##### Version

0.1

##### Date

2020-03-29

##### Copyright

Copyright (c) 2020

#### 4.11.2 Function Documentation

##### 4.11.2.1 SCHED\_createTask()

```
void SCHED_createTask (
    Task * appTask )
```

This function creates a task dynamically in the run time.

## Parameters

<i>appTask</i>	This is the application task desired to create
----------------	--

**4.11.2.2 SCHED\_init()**

```
void SCHED_init (
    void )
```

The initialization function.

**4.11.2.3 SCHED\_start()**

```
void SCHED_start (
    void )
```

Starts The running scheduel.

## 4.12 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED\_CONF.h File Reference

Those are the configurations for the Scheduler Driver.

**Macros**

- `#define SCHED_MAX_TASK_NUM 3`
- `#define SCHED_AHB_PREVAL RCC_AHB_NDIVIDED`
- `#define SCHED_AHB_CLOCK 8000000`
- `#define SCHED_TICK_TIME_US 1000`

**4.12.1 Detailed Description**

Those are the configurations for the Scheduler Driver.

**Author**

Mariam Mohammed

**Version**

0.1

**Date**

2020-03-29

**Copyright**

Copyright (c) 2020

## 4.13 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch.h File Reference

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

### Data Structures

- struct **switch\_t**

### 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.*
- void **Switch\_Task** (void)  
*The running task of the switch driver to get the state of all of the switches.*

#### 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 SWITCH_PRESSED : if the switch is pressed SWITCH_NOT_PRESSED : if the switch is not pressed

**Returns**

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

**4.13.2.3 Function: Switch\_GetSwitchStatus****Parameters**

<i>switchName</i>	The name of the Switch
<i>state</i>	Save the status of the switch in SWITCH_PRESSED : if the switch is pressed SWITCH_NOT_PRESSED : if the switch is not pressed

**Returns**

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

**4.13.2.4 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

**4.13.2.7 Switch\_Task()**

```
void Switch_Task (  
    void )
```

The running task of the switch driver to get the state of all of the switches.



## 4.14 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch\_Cfg.h File Reference

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

### Macros

- `#define SWITCH_USE_RTOS`
- `#define SWITCH_NUMBER_OF_SWITCHES 1`
- `#define SWITCH_1 0`
- `#define SWITCH_2 1`
- `#define SWITCH_3 2`

#### 4.14.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

## 4.15 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK.h File Reference

This is the user interface for the Systick Driver.

### Macros

- `#define SYSTICK_CLKSRC_AHB_DIV_8 0x00000000`
- `#define SYSTICK_CLKSRC_AHB 0x00000004`

### Typedefs

- `typedef void(* SYSTICK_cbF) (void)`

## Functions

- void **SYSTICK\_init** (void)  
*The initialization of the SysTick.*
- void **SYSTICK\_start** (void)  
*Starts the SysTick.*
- void **SYSTICK\_stop** (void)  
*Stops the timer.*
- void **SYSTICK\_setTime** (u32 time, u32 AHB\_clockHz)  
*Sets the timer for a specific time.*
- void **SYSTICK\_setCallbackFcn** (SYSTICK\_cbF cbF)  
*Sets the callback function.*

### 4.15.1 Detailed Description

This is the user interface for the SysTick Driver.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-29

#### Copyright

Copyright (c) 2020

### 4.15.2 Function Documentation

#### 4.15.2.1 SYSTICK\_init()

```
void SYSTICK_init (  
    void )
```

The initialization of the SysTick.

#### 4.15.2.2 SYSTICK\_setCallbackFcn()

```
void SYSTICK_setCallbackFcn (  
    SYSTICK_cbF cbF )
```

Sets the callback function.

## Parameters

<i>cbF</i>	the function to set
------------	---------------------

#### 4.15.2.3 SYSTICK\_setTime()

```
void SYSTICK_setTime (
    u32 time,
    u32 AHB_clockHz )
```

Sets the timer for a specific time.

## Parameters

<i>time</i>	the time in milli seconds
<i>AHB_clockHz</i>	the AHB clock in Kilo Hz

#### 4.15.2.4 SYSTICK\_start()

```
void SYSTICK_start (
    void )
```

Starts the SysTick.

#### 4.15.2.5 SYSTICK\_stop()

```
void SYSTICK_stop (
    void )
```

Stops the timer.

## 4.16 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK\_CONF.h File Reference

Those are the configurations for the Sysstick Driver.

### Macros

- `#define SYSTICK_CLKSRC_PRE SYSTICK_CLKSRC_AHB`

### 4.16.1 Detailed Description

Those are the configurations for the Systick Driver.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-29

#### Copyright

Copyright (c) 2020

## 4.17 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Uart.h File Reference

This is the user interface for the UART driver.

### Macros

- `#define UART1 0`
- `#define UART2 1`
- `#define UART3 2`
- `#define UART4 3`
- `#define UART5 4`
- `#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`

### Typedefs

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

## Functions

- Std\_ReturnType **Uart\_Init** (uint32\_t baudRate, uint32\_t stopBits, uint32\_t parity, uint32\_t flowControl, uint32\_t sysClk, uint8\_t uartModule)  
*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\_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.*

### 4.17.1 Detailed Description

This is the user interface for the UART driver.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.17.2 Function Documentation

#### 4.17.2.1 Uart\_Init()

```
Std_ReturnType Uart_Init (
    uint32_t baudRate,
    uint32_t stopBits,
    uint32_t parity,
    uint32_t flowControl,
    uint32_t sysClk,
    uint8_t uartModule )
```

Initializes the UART.

**Parameters**

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits UART_ONE_STOP_BIT UART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission UART_ODD_PARITY UART_EVEN_PARITY UART_NO_PARITY
<i>flowControl</i>	the flow control UART_FLOW_CONTROL_EN UART_FLOW_CONTROL_DIS
<i>sysClk</i>	the clock of the system
<i>uartModule</i>	the module number of the UART UART1 UART2 UART3 UART4 UART5

**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 UART1 UART2 UART3 UART4 UART5

**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\_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 UART1 UART2 UART3 UART4 UART5

## 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

#### 4.17.2.4 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 UART1 UART2 UART3 UART4 UART5

## Returns

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

#### 4.17.2.5 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 UART1 UART2 UART3 UART4 UART5

### Returns

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

## 4.18 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/App.c File Reference

This is an application for testing the UART and the LCD drivers.

```
#include "Std_Types.h"
#include <stdlib.h>
#include "stdio.h"
#include "HUART_Cfg.h"
#include "HUART.h"
#include "Clcd.h"
#include "Switch_Cfg.h"
#include "Switch.h"
#include "Led_Cfg.h"
#include "Led.h"
#include "App.h"
```

### Data Structures

- union **frame\_t**

*This is the frame type of size 4 byte.*

### Functions

- Std\_ReturnType **APP\_init** (void)

*This is the initialization for the two counter application.*

- void **APP\_sendTask** (void)

*The free running task that comes every 1 milli second.*

- void **APP\_receiveFcn** (void)

*The receive function that will be called after each received frame.*

### Variables

- **frame\_t** recFrame
- **frame\_t** sendFrame



### 4.18.1 Detailed Description

This is an application for testing the UART and the LCD drivers.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-28

#### Copyright

Copyright (c) 2020

### 4.18.2 Function Documentation

#### 4.18.2.1 APP\_init()

```
Std_ReturnType APP_init (  
    void )
```

This is the initialization for the two counter application.

#### Returns

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

#### 4.18.2.2 APP\_receiveFcn()

```
void APP_receiveFcn (  
    void )
```

The receive function that will be called after each received frame.

### 4.18.2.3 APP\_sendTask()

```
void APP_sendTask (
    void )
```

The free running task that comes every 1 milli second.

## 4.19 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Clcd.c File Reference

This file contains the implementation for the Character LCD Driver.

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

### Macros

- `#define CLCD_INITIALIZED 0`
- `#define CLCD_NOT_INITIALIZED 1`
- `#define CLCD_EMPTY_CMD 0x0`
- `#define CLCD_INIT_CONST 0x3`
- `#define CLCD_FUNC_SET 0x2`
- `#define CLCD_CLEAR_DISP 0x1`
- `#define CLCD_INC 0x6`
- `#define CLCD_DDRAM 0x80`
- `#define CLCD_SECOND_LINE 0x40`
- `#define CLCD_DISP_SETTING 0x8`
- `#define CLCD_CONFIG_DISP_CLR 0xF7`

### Enumerations

- `enum initState_t { hardwareInit_s, specialCaseFunctionSet_s, functionSet_s, display_s, clear_s, entry_s }`
- `enum writeState_t { setAddress_s, writeData_s }`
- `enum process_t { init_p, write_p, clear_p, goto_p, setup_p, idle_p }`
- `enum enable_t { low_s, high_s }`

## Functions

- Std\_ReturnType **CLcd\_Init** (uint8\_t nLines, uint8\_t cursor, uint8\_t blink)  
*The Character LCD initialization.*
- Std\_ReturnType **CLcd\_WriteString** (uint8\_t \*str, uint8\_t x, uint8\_t y)  
*Writes a string on a specific location on the lcd display.*
- Std\_ReturnType **CLcd\_ClearDisplay** (void)  
*Clears the display.*
- Std\_ReturnType **CLcd\_GotoXY** (uint8\_t x, uint8\_t y)  
*jumps to a specific location on the lcd display*
- Std\_ReturnType **CLcd\_ConfigCursor** (uint8\_t cursor, uint8\_t blink)  
*Configures the cursor options.*
- Std\_ReturnType **CLcd\_ConfigDisplay** (uint8\_t disp)  
*Sets the display on and off.*
- Std\_ReturnType **CLcd\_SetDoneNotification** (lcdCb\_t cb)  
*Sets the callback function executed when done.*
- void **CLcd\_Task** (void)  
*The running task that have to come every 1 milli second.*

## Variables

- const clcd\_t **CLcd\_clcd**

### 4.19.1 Detailed Description

This file contains the implementation for the Character LCD Driver.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.19.2 Function Documentation

#### 4.19.2.1 CLcd\_ClearDisplay()

```
Std_ReturnType CLcd_ClearDisplay (
    void )
```

Clears the display.

##### Returns

Std\_ReturnType E\_OK : If the clear operation started successfully E\_NOT\_OK : If the clear operation is not able to start right now

#### 4.19.2.2 CLcd\_ConfigCursor()

```
Std_ReturnType CLcd_ConfigCursor (
    uint8_t cursor,
    uint8_t blink )
```

Configures the cursor options.

##### Parameters

<i>cursor</i>	The State of the cursor (Visible or not) CLCD_CURSOR_ON CLCD_CURSOR_OFF
<i>blink</i>	The blinking option (no/off) CLCD_BLINKING_ON CLCD_BLINKING_OFF

##### Returns

Std\_ReturnType E\_OK : If the configuration started successfully E\_NOT\_OK : If the configuration is not able to start right now

#### 4.19.2.3 CLcd\_ConfigDisplay()

```
Std_ReturnType CLcd_ConfigDisplay (
    uint8_t disp )
```

Sets the display on and off.

##### Parameters

<i>disp</i>	the display state CLCD_DISP_ON CLCD_DISP_OFF
-------------	--

##### Returns

Std\_ReturnType E\_OK : If the configuration started successfully E\_NOT\_OK : If the configuration is not able to start right now

#### 4.19.2.4 CLcd\_GotoXY()

```
Std_ReturnType CLcd_GotoXY (
    uint8_t x,
    uint8_t y )
```

jumps to a specific location on the lcd display

##### Parameters

<i>x</i>	the location on the x-axis
<i>y</i>	the location on the y-axis

##### Returns

Std\_ReturnType E\_OK : If the goto operation started successfully E\_NOT\_OK : If the goto operation is not able to start right now

#### 4.19.2.5 CLcd\_Init()

```
Std_ReturnType CLcd_Init (
    uint8_t nLines,
    uint8_t cursor,
    uint8_t blink )
```

The Character LCD initialization.

##### Parameters

<i>nLines</i>	The number of lines on display CLCD_TWO_LINES : Two lines display CLCD_ONE_LINE : One line display
<i>cursor</i>	The State of the cursor (Visible or not) CLCD_CURSOR_ON CLCD_CURSOR_OFF
<i>blink</i>	The blinking option (no/off) CLCD_BLINKING_ON CLCD_BLINKING_OFF

##### Returns

Std\_ReturnType E\_OK : If the initialization started successfully E\_NOT\_OK : If the initialization is not able to start right now

#### 4.19.2.6 CLcd\_SetDoneNotification()

```
Std_ReturnType CLcd_SetDoneNotification (
    lcdCb_t cb )
```

Sets the callback function executed when done.

**Parameters**

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

**Returns**

Std\_ReturnType

**4.19.2.7 CLcd\_Task()**

```
void CLcd_Task (
    void )
```

The running task that have to come every 1 milli second.

**4.19.2.8 CLcd\_WriteString()**

```
Std_ReturnType CLcd_WriteString (
    uint8_t * str,
    uint8_t x,
    uint8_t y )
```

Writes a string on a specific location on the lcd display.

**Parameters**

<i>str</i>	the string to write
<i>x</i>	the location on the x-axis
<i>y</i>	the location on the y-axis

**Returns**

Std\_ReturnType E\_OK : If the writing started successfully E\_NOT\_OK : If the write operation is not able to start right now

## 4.20 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/CLcd\_Cfg.c File Reference

The user's configurations.

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

## Variables

- `const clcd_t CLcd_clcd`

### 4.20.1 Detailed Description

The user's configurations.

#### Author

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

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.20.2 Variable Documentation

#### 4.20.2.1 CLcd\_clcd

```
const clcd_t CLcd_clcd
```

##### Initial value:

```
= {  
    .enPin = GPIO_PIN_2,  
    .enPort = GPIO_PORTA,  
    .rwPin = GPIO_PIN_1,  
    .rwPort = GPIO_PORTA,  
    .rsPin = GPIO_PIN_0,  
    .rsPort = GPIO_PORTA,  
    .dPin = {GPIO_PIN_3, GPIO_PIN_4, GPIO_PIN_5, GPIO_PIN_6},  
    .dPort = {GPIO_PORTA, GPIO_PORTA, GPIO_PORTA, GPIO_PORTA}  
}
```

## 4.21 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Gpio.c File Reference

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

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

## Macros

- `#define GPIO_CR 0x00`
- `#define GPIO_IDR 0x08`
- `#define GPIO_ODR 0x0C`
- `#define GPIO_BSR 0x10`
- `#define GPIO_BRR 0x14`
- `#define GPIO_LCK 0x18`
- `#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)*

### 4.21.1 Detailed Description

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

#### Author

Mark Attia

#### Date

February 6, 2020

### 4.21.2 Function Documentation

#### 4.21.2.1 Gpio\_InitPins()

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

Initializes pins mode and speed for a specific port.

#### 4.21.2.2 Function: Gpio\_InitPins



**Parameters**

<i>gpio</i>	An object of type <b>gpio_t</b> (p. 6) 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

**4.21.2.3 Gpio\_ReadPin()**

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

Reads a value to a pin(0/1)

**4.21.2.4 Function: Gpio\_ReadPin****Parameters**

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

**Returns**

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

**4.21.2.5 Gpio\_WritePin()**

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

Write a value to a pin(0/1)

**4.21.2.6 Function: Gpio\_WritePin**

**Parameters**

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

**Returns**

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

## 4.22 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/HUart.c File Reference

This is the implementation for the UART handler.

```
#include "Std_Types.h"
#include "Uart.h"
#include "HUart_Cfg.h"
#include "HUart.h"
#include "NVIC.h"
#include "RCC.h"
#include "Gpio.h"
```

**Data Structures**

- struct **hUartConfig\_t**

**Macros**

- #define **HUART\_DEFAULT\_MODULE** HUART\_MODULE\_1
- #define **UART\_NUMBER\_OF\_MODULES** 5
- #define **HUART\_NOT\_INITIALIZED** 1
- #define **HUART\_INITIALIZED** 0
- #define **HUART\_NOT\_CONFIGURED** 0
- #define **HUART\_CONFIGURED** 1

## Functions

- Std\_ReturnType **HUart\_Init** (void)  
*Initializes the UART Module.*
- Std\_ReturnType **HUart\_Config** (uint32\_t baudRate, uint32\_t stopBits, uint32\_t parity, uint32\_t flowControl)  
*Sets configurations for the UART module \*The UART must be initialized after setting configurations to apply the changes.*
- Std\_ReturnType **HUart\_SetModule** (uint8\_t uartModule)  
*Sets the module that you will be using.*
- Std\_ReturnType **HUart\_Send** (uint8\_t \*data, uint16\_t length)  
*Sends data through the UART.*
- Std\_ReturnType **HUart\_Receive** (uint8\_t \*data, uint16\_t length)  
*Receives data through the UART.*
- Std\_ReturnType **HUart\_SetRxCb** (hUartRxCb\_t func)  
*Sets the callback function that will be called when receive is completed.*
- Std\_ReturnType **HUart\_SetTxCb** (hUartTxCb\_t func)  
*Sets the callback function that will be called when transmission is completed.*

### 4.22.1 Detailed Description

This is the implementation for the UART handler.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-29

#### Copyright

Copyright (c) 2020

### 4.22.2 Function Documentation

#### 4.22.2.1 HUart\_Config()

```
Std_ReturnType HUart_Config (  
    uint32_t baudRate,  
    uint32_t stopBits,  
    uint32_t parity,  
    uint32_t flowControl )
```

Sets configurations for the UART module \*The UART must be initialized after setting configurations to apply the changes.

**Parameters**

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits HUART_ONE_STOP_BIT HUART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission HUART_ODD_PARITY HUART_EVEN_PARITY HUART_NO_PARITY
<i>flowControl</i>	the flow control HUART_FLOW_CONTROL_EN HUART_FLOW_CONTROL_DIS

**Returns**

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

**4.22.2.2 HUart\_Init()**

```
Std_ReturnType HUart_Init (
    void )
```

Initializes the UART Module.

**Returns**

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

**4.22.2.3 HUart\_Receive()**

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

Receives data through the UART.

**Parameters**

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

**Returns**

Std\_ReturnType A Status E\_OK: If the driver is ready to receive E\_NOT\_OK: If the driver can't receive data right now

#### 4.22.2.4 HUart\_Send()

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

Sends data through the UART.

##### Parameters

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

##### Returns

Std\_ReturnType A Status E\_OK: If the driver is ready to send E\_NOT\_OK: If the driver can't send data right now

#### 4.22.2.5 HUart\_SetModule()

```
Std_ReturnType HUart_SetModule (
    uint8_t uartModule )
```

Sets the module that you will be using.

##### Parameters

<i>uartModule</i>	The UART module HUART_MODULE_1 HUART_MODULE_2 HUART_MODULE_3 HUART_MODULE_4 HUART_MODULE_5
-------------------	--

##### Returns

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

#### 4.22.2.6 HUart\_SetRxCb()

```
Std_ReturnType HUart_SetRxCb (
    hUartRxCb_t func )
```

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

##### Parameters

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

**Returns**

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

**4.22.2.7 HUART\_SetTxCb()**

```
Std_ReturnType HUART_SetTxCb (
    hUartTxCb_t func )
```

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

**Parameters**

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

**Returns**

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

## 4.23 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Led.c File Reference

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

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

**Functions**

- Std\_ReturnType **Led\_Init** (void)  
*Initializes GPIOs for the LEDs.*
- Std\_ReturnType **Led\_SetLedOn** (uint8\_t ledName)  
*Sets the Led on.*
- Std\_ReturnType **Led\_SetLedOff** (uint8\_t ledName)  
*Sets the Led off.*
- Std\_ReturnType **Led\_SetLedStatus** (uint8\_t ledName, uint8\_t status)  
*Sets the Led off.*

## Variables

- const **led\_t** **Led\_leds** [LED\_NUMBER\_OF\_LEDS]

### 4.23.1 Detailed Description

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

#### Author

Mark Attia

#### Date

January 22, 2020

### 4.23.2 Function Documentation

#### 4.23.2.1 Led\_Init()

```
Std_ReturnType Led_Init (  
    void )
```

Initializes GPIOs for the LEDs.

#### 4.23.2.2 Function: Led\_Init

##### Returns

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

#### 4.23.2.3 Led\_SetLedOff()

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

Sets the Led off.

#### 4.23.2.4 Function: Led\_SetLedOff

**Parameters**

<i>ledName</i>	The name of the LED
----------------	---------------------

**Returns**

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

**4.23.2.5 Led\_SetLedOn()**

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

Sets the Led on.

**4.23.2.6 Function: Led\_SetLedOn****Parameters**

<i>ledName</i>	The name of the LED
----------------	---------------------

**Returns**

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

**4.23.2.7 Led\_SetLedStatus()**

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

Sets the Led off.

**4.23.2.8 Function: Led\_SetLedStatus****Parameters**

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



#### Returns

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

## 4.24 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Led\_Cfg.c File Reference

Those are the User's configurations for the LED Driver.

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

### Variables

- const **led\_t** **Led\_leds** [LED\_NUMBER\_OF\_LEDS]

#### 4.24.1 Detailed Description

Those are the User's configurations for the LED Driver.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-28

#### Copyright

Copyright (c) 2020

#### 4.24.2 Variable Documentation

#### 4.24.2.1 Led\_leds

```
const led_t Led_leds[LED_NUMBER_OF_LEDS]
```

**Initial value:**

```
= {  
    {GPIO_PIN_13, GPIO_PORTC, GPIO_PIN_RESET}  
}
```

### 4.25 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/main.c File Reference

Here is the implementation for the main function for the application and also the tasks.

```
#include "Std_Types.h"  
#include "SCHED1.h"  
#include "HRcc.h"  
#include "CLcd.h"  
#include "App.h"  
#include "Switch.h"
```

#### Functions

- void **main** (void)

#### Variables

- **Task t1** = { **APP\_sendTask**, 1000, 2}
- **Task t2** = { **CLcd\_Task**, 1000, 1}
- **Task t3** = { **Switch\_Task**, 1000, 0}

#### 4.25.1 Detailed Description

Here is the implementation for the main function for the application and also the tasks.

**Author**

Mariam Mohammed

**Version**

0.1

**Date**

2020-03-28

**Copyright**

Copyright (c) 2020

## 4.26 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/NVIC.c File Reference

This is the implementation for the NVIC Driver.

```
#include "Std_Types.h"
#include "NVIC.h"
```

### Data Structures

- struct **NVIC\_regMap**

### Macros

- #define **NVIC\_BASE\_ADDRESS** 0xE000E100
- #define **NVIC\_peripheral** ((volatile **NVIC\_regMap** \*) NVIC\_BASE\_ADDRESS)
- #define **NVIC\_IPR\_SETMASK** 0x000000ff

### Functions

- void **NVIC\_controlInterrupt** (u8 interruptNum, u8 status)  
*Sets and resets the interrupts.*
- void **NVIC\_controlPendingFlag** (u8 interruptNum, u8 val)  
*Sets and resets The pending flag.*
- u8 **NVIC\_getActiveFlagStatus** (u8 interruptNum)  
*Gets the active flag state.*
- void **NVIC\_configurePriority** (u8 interruptNum, u8 priority)  
*Configures the periority of the interrupt.*
- u8 **NVIC\_getPriority** (u8 interruptNum)  
*Gets the priority of the interrupt.*
- void **NVIC\_controlAllPeripheral** (u8 status)  
*Controls All of the prephirals.*
- void **NVIC\_controlFault** (u8 status)  
*Controls The Fault flag.*
- void **NVIC\_filterInterrupts** (u8 priority)  
*Filters the interrupt.*

### 4.26.1 Detailed Description

This is the implementation for the NVIC Driver.

#### Author

Mariam Mohammed

#### Version

0.1

#### Date

2020-03-28

#### Copyright

Copyright (c) 2020

### 4.26.2 Function Documentation

#### 4.26.2.1 NVIC\_configurePriority()

```
void NVIC_configurePriority (
    u8 interruptNum,
    u8 priority )
```

Configures the periority of the interrupt.

#### Parameters

<i>interruptNum</i>	the number of the interrupt
<i>priority</i>	The periority

#### 4.26.2.2 NVIC\_controlAllPeripheral()

```
void NVIC_controlAllPeripheral (
    u8 status )
```

Controls All of the prephirals.

## Parameters

<i>status</i>	NVIC_ENABLE NVIC_DISABLE
---------------	--------------------------

**4.26.2.3 NVIC\_controlFault()**

```
void NVIC_controlFault (
    u8 status )
```

Controls The Fault flag.

## Parameters

<i>status</i>	NVIC_ENABLE NVIC_DISABLE
---------------	--------------------------

**4.26.2.4 NVIC\_controlInterrupt()**

```
void NVIC_controlInterrupt (
    u8 interruptNum,
    u8 status )
```

Sets and resets the interrupts.

## Parameters

<i>interruptNum</i>	The Interrupt number
<i>status</i>	The state NVIC_DISABLE NVIC_ENABLE

**4.26.2.5 NVIC\_controlPendingFlag()**

```
void NVIC_controlPendingFlag (
    u8 interruptNum,
    u8 val )
```

Sets and resets The pending flag.

## Parameters

<i>interruptNum</i>	The Interrupt number
<i>val</i>	the value to be set NVIC_RESET NVIC_SET

#### 4.26.2.6 NVIC\_filterInterrupts()

```
void NVIC_filterInterrupts (
    u8 priority )
```

Filters the interrupt.

##### Parameters

<i>priority</i>	the priority of the interrupt
-----------------	-------------------------------

#### 4.26.2.7 NVIC\_getActiveFlagStatus()

```
u8 NVIC_getActiveFlagStatus (
    u8 interruptNum )
```

Gets the active flag state.

##### Parameters

<i>interruptNum</i>	the number of the interrupt
---------------------	-----------------------------

##### Returns

u8

#### 4.26.2.8 NVIC\_getPriority()

```
u8 NVIC_getPriority (
    u8 interruptNum )
```

Gets the priority of the interrupt.

##### Parameters

<i>interruptNum</i>	the number of the interrupt
---------------------	-----------------------------

##### Returns

u8

## 4.27 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/SCHED.c File Reference

This is the implementation of the scheduler.

```
#include "Std_Types.h"
#include "RCC.h"
#include "SYSTICK.h"
#include "SCHED1.h"
#include "SCHED_CONF.h"
```

### Data Structures

- struct **SysTask**

### Functions

- void **SCHED\_init** (void)  
*The initialization function.*
- void **SCHED\_createTask** ( **Task** \*appTask)  
*This function creates a task dynamically in the run time.*
- void **SCHED\_start** (void)  
*Starts The running scheduel.*

#### 4.27.1 Detailed Description

This is the implementation of the scheduler.

##### Author

Mariam Mohammed

##### Version

0.1

##### Date

2020-03-28

##### Copyright

Copyright (c) 2020

#### 4.27.2 Function Documentation

##### 4.27.2.1 SCHED\_createTask()

```
void SCHED_createTask (
    Task * appTask )
```

This function creates a task dynamically in the run time.

## Parameters

<i>appTask</i>	This is the application task desired to create
----------------	--

**4.27.2.2 SCHED\_init()**

```
void SCHED_init (
    void )
```

The initialization function.

**4.27.2.3 SCHED\_start()**

```
void SCHED_start (
    void )
```

Starts The running scheduel.

## 4.28 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/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_Cfg.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.*
- void **Switch\_Task** (void)  
*The running task of the switch driver to get the state of all of the switches.*

**Variables**

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



## **4.28.1 Detailed Description**

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

### **Author**

Mark Attia

### **Date**

January 22, 2020

## **4.28.2 Function Documentation**

### **4.28.2.1 Switch\_GetSwitchStatus()**

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

Gets the status of the switch.

### **4.28.2.2 Function: Switch\_GetSwitchStatus**

#### **Parameters**

<i>switchName</i>	The name of the Switch
<i>state</i>	Save the status of the switch in SWITCH_PRESSED : if the switch is pressed SWITCH_NOT_PRESSED : if the switch is not pressed

#### **Returns**

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

### **4.28.2.3 Switch\_Init()**

```
Std_ReturnType Switch_Init (
    void )
```

Initializes GPIOs for the Switches.

#### 4.28.2.4 Function: Switch\_Init

##### Returns

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

#### 4.28.2.5 Switch\_Task()

```
void Switch_Task (
    void )
```

The running task of the switch driver to get the state of all of the switches.

## 4.29 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/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** (p. 47).

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

### Variables

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

#### 4.29.1 Detailed Description

This file is to be used as an implementation of the configurations the user configured in the **Switch\_Cfg.h** (p. 47).

##### Author

Mark Attia

##### Date

January 22, 2020

#### 4.29.2 Variable Documentation

#### 4.29.2.1 Switch\_switches

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

**Initial value:**

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

## 4.30 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/SYSTICK.c File Reference

This is the SysTick driver implementation.

```
#include "Std_Types.h"  
#include "SYSTICK_CONF.h"  
#include "SYSTICK.h"
```

### Data Structures

- struct **SYSTICK\_regMap**

### Macros

- #define **SYSTICK\_BASE\_ADDRESS** 0xE000E010
- #define **SYSTICK\_peripheral** ((volatile **SYSTICK\_regMap** \*) SYSTICK\_BASE\_ADDRESS)
- #define **SYSTICK\_ENABLE\_SETMASK** 0x00000001
- #define **SYSTICK\_TICKINT\_SETMASK** 0x00000002
- #define **SYSTICK\_CLKSRC\_SETMASK** 0x00000004

### Functions

- void **SYSTICK\_init** (void)  
*The initialization of the SysTick.*
- void **SYSTICK\_start** (void)  
*Starts the SysTick.*
- void **SYSTICK\_stop** (void)  
*Stops the timer.*
- void **SYSTICK\_setTime** (u32 time, u32 AHB\_clockHz)  
*Sets the timer for a specific time.*
- void **SYSTICK\_setCallbackFcn** (SYSTICK\_cbF cbF)  
*Sets the callback function.*
- void **SysTick\_Handler** (void)  
*The SysTick Handler.*

### 4.30.1 Detailed Description

This is the SysTick driver implementation.

**Author**

Mariam Mohammed

**Version**

0.1

**Date**

2020-03-28

**Copyright**

Copyright (c) 2020

### 4.30.2 Function Documentation

#### 4.30.2.1 SysTick\_Handler()

```
void SysTick_Handler (
    void )
```

The SysTick Handler.

#### 4.30.2.2 SYSTICK\_init()

```
void SYSTICK_init (
    void )
```

The initialization of the SysTick.

#### 4.30.2.3 SYSTICK\_setCallbackFcn()

```
void SYSTICK_setCallbackFcn (
    SYSTICK_cbF cbF )
```

Sets the callback function.

## Parameters

<i>cbF</i>	the function to set
------------	---------------------

**4.30.2.4 SYSTICK\_setTime()**

```
void SYSTICK_setTime (
    u32 time,
    u32 AHB_clockHz )
```

Sets the timer for a specific time.

## Parameters

<i>time</i>	the time in milli seconds
<i>AHB_clockHz</i>	the AHB clock in Kilo Hz

**4.30.2.5 SYSTICK\_start()**

```
void SYSTICK_start (
    void )
```

Starts the Systick.

**4.30.2.6 SYSTICK\_stop()**

```
void SYSTICK_stop (
    void )
```

Stops the timer.

## 4.31 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Src/Uart.c File Reference

This is the implementation for the UART driver.

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

## Data Structures

- struct **uart\_t**
- struct **dataBuffer\_t**

## Macros

- #define **UART\_NUMBER\_OF\_MODULES** 5
- #define **UART\_INT\_NUMBER** 37
- #define **UART\_BUFFER\_IDLE** 0
- #define **UART\_BUFFER\_BUSY** 1
- #define **UART\_TXE\_CLR** 0xFFFFF7F
- #define **UART\_TC\_CLR** 0xFFFFFBF
- #define **UART\_RXNE\_CLR** 0xFFFFFDF
- #define **UART\_PE\_CLR** 0xFFFFFE
- #define **UART\_DR\_CLR** 0xFFFFE00
- #define **UART\_STOP\_CLR** 0xFFFFCFFF
- #define **UART\_TXEIE\_CLR** 0xFFFFF7F
- #define **UART\_PS\_CLR** 0xFFFFDFF
- #define **UART\_M\_CLR** 0xFFFFEFF
- #define **UART\_TXE\_GET** 0x0000080
- #define **UART\_TC\_GET** 0x0000040
- #define **UART\_RXNE\_GET** 0x0000020
- #define **UART\_PE\_GET** 0x0000001
- #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\_RTSE\_CLR** 0xFFFFFEFF
- #define **UART\_NO\_PRESCALER** 0x1

## Functions

- void **USART1\_IRQHandler** (void)  
*The UART 1 Handler.*
- void **USART2\_IRQHandler** (void)  
*The UART 2 Handler.*
- void **USART3\_IRQHandler** (void)  
*The UART 3 Handler.*
- void **UART4\_IRQHandler** (void)  
*The UART 4 Handler.*
- void **UART5\_IRQHandler** (void)  
*The UART 5 Handler.*
- Std\_ReturnType **Uart\_Init** (uint32\_t baudRate, uint32\_t stopBits, uint32\_t parity, uint32\_t flowControl, uint32\_t sysClk, uint8\_t uartModule)  
*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\_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.*

## Variables

- const uint32\_t **Uart\_Address** [UART\_NUMBER\_OF\_MODULES]

### 4.31.1 Detailed Description

This is the implementation for the UART driver.

#### Author

Mark Attia ( markjosephattia@gmail.com)

#### Version

0.1

#### Date

2020-03-26

#### Copyright

Copyright (c) 2020

### 4.31.2 Function Documentation

#### 4.31.2.1 UART4\_IRQHandler()

```
void UART4_IRQHandler (  
    void )
```

The UART 4 Handler.

#### 4.31.2.2 UART5\_IRQHandler()

```
void UART5_IRQHandler (
    void )
```

The UART 5 Handler.

#### 4.31.2.3 Uart\_Init()

```
Std_ReturnType Uart_Init (
    uint32_t baudRate,
    uint32_t stopBits,
    uint32_t parity,
    uint32_t flowControl,
    uint32_t sysClk,
    uint8_t uartModule )
```

Initializes the UART.

##### Parameters

<i>baudRate</i>	the baud rate of the UART (uint32_t)
<i>stopBits</i>	The number of the stop bits UART_ONE_STOP_BIT UART_TWO_STOP_BITS
<i>parity</i>	The parity of the transmission UART_ODD_PARITY UART_EVEN_PARITY UART_NO_PARITY
<i>flowControl</i>	the flow control UART_FLOW_CONTROL_EN UART_FLOW_CONTROL_DIS
<i>sysClk</i>	the clock of the system
<i>uartModule</i>	the module number of the UART UART1 UART2 UART3 UART4 UART5

##### Returns

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

#### 4.31.2.4 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 UART1 UART2 UART3 UART4 UART5



**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

**4.31.2.5 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 UART1 UART2 UART3 UART4 UART5

**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

**4.31.2.6 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 UART1 UART2 UART3 UART4 UART5

**Returns**

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

#### 4.31.2.7 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 UART1 UART2 UART3 UART4 UART5

##### Returns

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

#### 4.31.2.8 USART1\_IRQHandler()

```
void USART1_IRQHandler (
    void )
```

The UART 1 Handler.

#### 4.31.2.9 USART2\_IRQHandler()

```
void USART2_IRQHandler (
    void )
```

The UART 2 Handler.

#### 4.31.2.10 USART3\_IRQHandler()

```
void USART3_IRQHandler (
    void )
```

The UART 3 Handler.

### 4.31.3 Variable Documentation

#### 4.31.3.1 Uart\_Address

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

Initial value:

```
= {  
    0x40013800,  
    0x40004400,  
    0x40004800,  
    0x40004C00,  
    0x40005000  
}
```

## 4.32 C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Std\_Types.h File Reference

Those are the standard types used in the drivers.

### Macros

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

### Typedefs

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

### 4.32.1 Detailed Description

Those are the standard types used in the drivers.

**Author**

Mark Attia

**Version**

0.1

**Date**

2020-03-29

**Copyright**

Copyright (c) 2020

# Index

App.c	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
APP_init, 55	50
APP_receiveFcn, 55	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
APP_sendTask, 55	54
App.h	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
APP_init, 12	56
APP_receiveFcn, 12	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
APP_sendTask, 12	60
APP_init	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
App.c, 55	61
App.h, 12	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
APP_receiveFcn	64
App.c, 55	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
App.h, 12	68
APP_sendTask	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
App.c, 55	71
App.h, 12	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
	72
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/App.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
11	73
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/CLcd.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
12	77
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Gpio.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
17	78
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HBcc.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
20	80
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUART.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
22	81
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/HUART_Cfg.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
26	83
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led.h,	C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/
27	89
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Led_Cfg.h,	CLcd.c
30	CLcd_ClearDisplay, 57
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/NVIC.h,	CLcd_ConfigCursor, 58
30	CLcd_ConfigDisplay, 58
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/RCC.h,	CLcd_GotoXY, 59
37	CLcd_Init, 59
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED1.h,	CLcd_SetDoneNotification, 59
43	CLcd_Task, 60
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SCHED_CONF.h,	CLcd_WriteString, 60
44	CLcd.h
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch.h,	CLcd_ClearDisplay, 14
45	CLcd_ConfigCursor, 14
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/Switch_Cfg.h,	CLcd_ConfigDisplay, 14
47	CLcd_GotoXY, 15
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK.h,	CLcd_Init, 15
47	CLcd_SetDoneNotification, 15
C:/Users/Mark/Desktop/TwoCountersProject-master/TwoCountersProject/Include/SYSTICK_CONF.h,	CLcd_Task, 16
49	

---

- CLcd\_WriteString, 16
- CLcd\_Cfg.c
  - CLcd\_clcd, 61
- CLcd\_clcd
  - CLcd\_Cfg.c, 61
- CLcd\_ClearDisplay
  - Clcd.c, 57
  - CLcd.h, 14
- CLcd\_ConfigCursor
  - Clcd.c, 58
  - CLcd.h, 14
- CLcd\_ConfigDisplay
  - Clcd.c, 58
  - CLcd.h, 14
- CLcd\_GotoXY
  - Clcd.c, 59
  - CLcd.h, 15
- CLcd\_Init
  - Clcd.c, 59
  - CLcd.h, 15
- CLcd\_SetDoneNotification
  - Clcd.c, 59
  - CLcd.h, 15
- clcd\_t, 5
- CLcd\_Task
  - Clcd.c, 60
  - CLcd.h, 16
- CLcd\_WriteString
  - Clcd.c, 60
  - CLcd.h, 16
- dataBuffer\_t, 5
- frame\_t, 6
- Gpio.c
  - Gpio\_InitPins, 62
  - Gpio\_ReadPin, 63
  - Gpio\_WritePin, 63
- Gpio.h
  - Gpio\_InitPins, 18
  - Gpio\_ReadPin, 19
  - Gpio\_WritePin, 19
- Gpio\_InitPins
  - Gpio.c, 62
  - Gpio.h, 18
- Gpio\_ReadPin
  - Gpio.c, 63
  - Gpio.h, 19
- gpio\_t, 6
- Gpio\_WritePin
  - Gpio.c, 63
  - Gpio.h, 19
- HRcc.h
  - HRcc\_EnPortClock, 21
  - HRcc\_SystemClockInit, 21
- HRcc\_EnPortClock
  - HRcc.h, 21
- HRcc\_SystemClockInit
  - HRcc.h, 21
- HUart.c
  - HUart\_Config, 65
  - HUart\_Init, 66
  - HUart\_Receive, 66
  - HUart\_Send, 66
  - HUart\_SetModule, 67
  - HUart\_SetRxCb, 67
  - HUart\_SetTxCb, 68
- HUart.h
  - HUart\_Config, 23
  - HUart\_Init, 23
  - HUart\_Receive, 24
  - HUart\_Send, 24
  - HUart\_SetModule, 25
  - HUart\_SetRxCb, 25
  - HUart\_SetTxCb, 25
- HUart\_Config
  - HUart.c, 65
  - HUart.h, 23
- HUart\_Init
  - HUart.c, 66
  - HUart.h, 23
- HUart\_Receive
  - HUart.c, 66
  - HUart.h, 24
- HUart\_Send
  - HUart.c, 66
  - HUart.h, 24
- HUart\_SetModule
  - HUart.c, 67
  - HUart.h, 25
- HUart\_SetRxCb
  - HUart.c, 67
  - HUart.h, 25
- HUart\_SetTxCb
  - HUart.c, 68
  - HUart.h, 25
- hUartConfig\_t, 6
- Led.c
  - Led\_Init, 69
  - Led\_SetLedOff, 69
  - Led\_SetLedOn, 70
  - Led\_SetLedStatus, 70
- Led.h
  - Led\_Init, 27
  - Led\_SetLedOff, 28
  - Led\_SetLedOn, 28
  - Led\_SetLedStatus, 29
- Led\_Cfg.c
  - Led\_leds, 71
- Led\_Init
  - Led.c, 69
  - Led.h, 27
- Led\_leds
  - Led\_Cfg.c, 71
- Led\_SetLedOff

- Led.c, 69
- Led.h, 28
- Led\_SetLedOn
  - Led.c, 70
  - Led.h, 28
- Led\_SetLedStatus
  - Led.c, 70
  - Led.h, 29
- led\_t, 7
- NVIC.c
  - NVIC\_configurePriority, 74
  - NVIC\_controlAllPeripheral, 74
  - NVIC\_controlFault, 75
  - NVIC\_controlInterrupt, 75
  - NVIC\_controlPendingFlag, 75
  - NVIC\_filterInterrupts, 76
  - NVIC\_getActiveFlagStatus, 76
  - NVIC\_getPriority, 76
- NVIC.h
  - NVIC\_configurePriority, 33
  - NVIC\_controlAllPeripheral, 33
  - NVIC\_controlFault, 35
  - NVIC\_controlInterrupt, 35
  - NVIC\_controlPendingFlag, 35
  - NVIC\_DISABLE, 33
  - NVIC\_filterInterrupts, 36
  - NVIC\_getActiveFlagStatus, 36
  - NVIC\_getPriority, 36
  - NVIC\_IRQNUM\_WWDG, 33
  - NVIC\_RESET, 33
- NVIC\_configurePriority
  - NVIC.c, 74
  - NVIC.h, 33
- NVIC\_controlAllPeripheral
  - NVIC.c, 74
  - NVIC.h, 33
- NVIC\_controlFault
  - NVIC.c, 75
  - NVIC.h, 35
- NVIC\_controlInterrupt
  - NVIC.c, 75
  - NVIC.h, 35
- NVIC\_controlPendingFlag
  - NVIC.c, 75
  - NVIC.h, 35
- NVIC\_DISABLE
  - NVIC.h, 33
- NVIC\_filterInterrupts
  - NVIC.c, 76
  - NVIC.h, 36
- NVIC\_getActiveFlagStatus
  - NVIC.c, 76
  - NVIC.h, 36
- NVIC\_getPriority
  - NVIC.c, 76
  - NVIC.h, 36
- NVIC\_IRQNUM\_WWDG
  - NVIC.h, 33
- NVIC\_regMap, 7
- NVIC\_RESET
  - NVIC.h, 33
- RCC.h
  - RCC\_configureMCO, 39
  - RCC\_configurePLL, 40
  - RCC\_configurePrescalers, 40
  - RCC\_controlAHBPeripheral, 41
  - RCC\_controlAPB1Peripheral, 41
  - RCC\_controlAPB2Peripheral, 42
  - RCC\_selectSystemClock, 42
  - RCC\_setClockState, 42
- RCC\_configureMCO
  - RCC.h, 39
- RCC\_configurePLL
  - RCC.h, 40
- RCC\_configurePrescalers
  - RCC.h, 40
- RCC\_controlAHBPeripheral
  - RCC.h, 41
- RCC\_controlAPB1Peripheral
  - RCC.h, 41
- RCC\_controlAPB2Peripheral
  - RCC.h, 42
- RCC\_regMap, 7
- RCC\_selectSystemClock
  - RCC.h, 42
- RCC\_setClockState
  - RCC.h, 42
- SCHED.c
  - SCHED\_createTask, 77
  - SCHED\_init, 78
  - SCHED\_start, 78
- SCHED1.h
  - SCHED\_createTask, 43
  - SCHED\_init, 44
  - SCHED\_start, 44
- SCHED\_createTask
  - SCHED.c, 77
  - SCHED1.h, 43
- SCHED\_init
  - SCHED.c, 78
  - SCHED1.h, 44
- SCHED\_start
  - SCHED.c, 78
  - SCHED1.h, 44
- Switch.c
  - Switch\_GetSwitchStatus, 79
  - Switch\_Init, 79
  - Switch\_Task, 80
- Switch.h
  - Switch\_GetSwitchStatus, 45
  - Switch\_Init, 46
  - Switch\_Task, 46
- Switch\_Cfg.c
  - Switch\_switches, 80
- Switch\_GetSwitchStatus

- Switch.c, 79
- Switch.h, 45
- Switch\_Init
  - Switch.c, 79
  - Switch.h, 46
- Switch\_switches
  - Switch\_Cfg.c, 80
- switch\_t, 8
- Switch\_Task
  - Switch.c, 80
  - Switch.h, 46
- SysTask, 8
- SYSTICK.c
  - SysTick\_Handler, 82
  - SYSTICK\_init, 82
  - SYSTICK\_setCallbackFcn, 82
  - SYSTICK\_setTime, 83
  - SYSTICK\_start, 83
  - SYSTICK\_stop, 83
- SYSTICK.h
  - SYSTICK\_init, 48
  - SYSTICK\_setCallbackFcn, 48
  - SYSTICK\_setTime, 49
  - SYSTICK\_start, 49
  - SYSTICK\_stop, 49
- SysTick\_Handler
  - SYSTICK.c, 82
- SYSTICK\_init
  - SYSTICK.c, 82
  - SYSTICK.h, 48
- SYSTICK\_regMap, 8
- SYSTICK\_setCallbackFcn
  - SYSTICK.c, 82
  - SYSTICK.h, 48
- SYSTICK\_setTime
  - SYSTICK.c, 83
  - SYSTICK.h, 49
- SYSTICK\_start
  - SYSTICK.c, 83
  - SYSTICK.h, 49
- SYSTICK\_stop
  - SYSTICK.c, 83
  - SYSTICK.h, 49
- Task, 8
- Uart.c
  - UART4\_IRQHandler, 85
  - UART5\_IRQHandler, 85
  - Uart\_Address, 88
  - Uart\_Init, 86
  - Uart\_Receive, 86
  - Uart\_Send, 87
  - Uart\_SetRxCb, 87
  - Uart\_SetTxCb, 87
  - USART1\_IRQHandler, 88
  - USART2\_IRQHandler, 88
  - USART3\_IRQHandler, 88
- Uart.h
  - Uart\_Init, 51
  - Uart\_Receive, 52
  - Uart\_Send, 52
  - Uart\_SetRxCb, 53
  - Uart\_SetTxCb, 53
  - UART4\_IRQHandler
    - Uart.c, 85
  - UART5\_IRQHandler
    - Uart.c, 85
  - Uart\_Address
    - Uart.c, 88
  - Uart\_Init
    - Uart.c, 86
    - Uart.h, 51
  - Uart\_Receive
    - Uart.c, 86
    - Uart.h, 52
  - Uart\_Send
    - Uart.c, 87
    - Uart.h, 52
  - Uart\_SetRxCb
    - Uart.c, 87
    - Uart.h, 53
  - Uart\_SetTxCb
    - Uart.c, 87
    - Uart.h, 53
  - uart\_t, 9
  - USART1\_IRQHandler
    - Uart.c, 88
  - USART2\_IRQHandler
    - Uart.c, 88
  - USART3\_IRQHandler
    - Uart.c, 88