

My Project

Generated by Doxygen 1.13.2

1 Topic Index	1
1.1 Topics	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Topic Documentation	7
4.1 DS3231 Real-Time Clock Driver	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 RTC_ClearFlagAlarm1()	8
4.1.2.2 RTC_ClearFlagAlarm2()	8
4.1.2.3 RTC_ConvertToBCD()	8
4.1.2.4 RTC_ConvertToDEC()	8
4.1.2.5 RTC_CtrlAlarm1()	9
4.1.2.6 RTC_CtrlAlarm2()	9
4.1.2.7 RTC_Deinit()	9
4.1.2.8 RTC_GetDate()	9
4.1.2.9 RTC_GetState()	10
4.1.2.10 RTC_GetTime()	10
4.1.2.11 RTC_Init()	10
4.1.2.12 RTC_Read()	10
4.1.2.13 RTC_SetDate()	11
4.1.2.14 RTC_SetDateDefault()	11
4.1.2.15 RTC_SetInterruptMode()	11
4.1.2.16 RTC_SetOscState()	11
4.1.2.17 RTC_SetTime()	12
4.1.2.18 RTC_SetTimeDefault()	12
4.1.2.19 RTC_Write()	12
5 Class Documentation	13
5.1 REC_config_t Struct Reference	13
5.1.1 Detailed Description	13
5.1.2 Member Data Documentation	13
5.1.2.1 baudrate	13
5.1.2.2 data_bits	14
5.1.2.3 free_space_limit_mb	14
5.1.2.4 max_bytes	14
5.1.2.5 parity	14
5.1.2.6 size	14
5.1.2.7 stop_bits	14

5.1.2.8 version	14
5.2 RTC_date_t Struct Reference	15
5.2.1 Detailed Description	15
5.2.2 Member Data Documentation	15
5.2.2.1 date	15
5.2.2.2 day	15
5.2.2.3 month	15
5.2.2.4 year	15
5.3 RTC_time_t Struct Reference	16
5.3.1 Detailed Description	16
5.3.2 Member Data Documentation	16
5.3.2.1 format	16
5.3.2.2 hrs	16
5.3.2.3 min	16
5.3.2.4 sec	16
6 File Documentation	17
6.1 drivers/rtc_ds3231.c File Reference	17
6.1.1 Macro Definition Documentation	18
6.1.1.1 BYTE_1	18
6.1.1.2 BYTE_2	18
6.1.1.3 CLK_STATE	19
6.1.1.4 I2C_DATA_LENGTH	19
6.1.2 Function Documentation	19
6.1.2.1 AT_NONCACHEABLE_SECTION() [1/2]	19
6.1.2.2 AT_NONCACHEABLE_SECTION() [2/2]	19
6.1.2.3 lpi2c_callback()	19
6.1.2.4 RTC_ConvertToBCD()	19
6.1.2.5 RTC_ConvertToDEC()	20
6.1.3 Variable Documentation	20
6.1.3.1 gCompletionFlag	20
6.1.3.2 gEdmaRxHandle	20
6.1.3.3 gEdmaTxHandle	20
6.1.3.4 xfer	20
6.2 drivers/rtc_ds3231.h File Reference	20
6.2.1 Macro Definition Documentation	23
6.2.1.1 ALARM_DISABLED	23
6.2.1.2 APP_SUCCESS	23
6.2.1.3 DS3231_A1F	23
6.2.1.4 DS3231_A1IE	23
6.2.1.5 DS3231_A2F	23
6.2.1.6 DS3231_A2IE	23

6.2.1.7 DS3231_ADDR_CENT	23
6.2.1.8 DS3231_ADDR_DATE	23
6.2.1.9 DS3231_ADDR_DAY	24
6.2.1.10 DS3231_ADDR_HRS	24
6.2.1.11 DS3231_ADDR_I2C	24
6.2.1.12 DS3231_ADDR_MIN	24
6.2.1.13 DS3231_ADDR_MONTH	24
6.2.1.14 DS3231_ADDR_SEC	24
6.2.1.15 DS3231_ADDR_YEAR	24
6.2.1.16 DS3231_INTCN	25
6.2.1.17 DS3231_REG_CTRL	25
6.2.1.18 DS3231_REG_STATUS	25
6.2.1.19 E_FAULT	25
6.2.1.20 FRIDAY	25
6.2.1.21 I2C_BAUDRATE	25
6.2.1.22 I2C_MASTER	25
6.2.1.23 LPI2C_RX_DMA_CHANNEL	26
6.2.1.24 LPI2C_RX_EDMA_CHANNEL	26
6.2.1.25 LPI2C_TX_CHANNEL	26
6.2.1.26 LPI2C_TX_DMA_CHANNEL	26
6.2.1.27 MONDAY	26
6.2.1.28 OSC_STOPPED	26
6.2.1.29 SATURDAY	26
6.2.1.30 SUNDAY	27
6.2.1.31 THURSDAY	27
6.2.1.32 TIM_CYCLE_12H	27
6.2.1.33 TIM_CYCLE_12H_AM	27
6.2.1.34 TIM_CYCLE_12H_PM	27
6.2.1.35 TIM_CYCLE_24H	27
6.2.1.36 TUESDAY	27
6.2.1.37 WEDNESDAY	27
6.2.2 Enumeration Type Documentation	27
6.2.2.1 RTC_interrupt_mode_t	27
6.2.2.2 RTC_osc_state_t	28
6.3 rtc_ds3231.h	28
6.4 include/app_init.h File Reference	30
6.4.1 Function Documentation	30
6.4.1.1 APP_InitBoard()	30
6.5 app_init.h	31
6.6 include/app_tasks.h File Reference	31
6.6.1 Macro Definition Documentation	32
6.6.1.1 TASK_Prio	32

6.6.2 Function Documentation	32
6.6.2.1 msc_task()	32
6.6.2.2 record_task()	32
6.6.2.3 vApplicationGetIdleTaskMemory()	33
6.6.2.4 vApplicationGetTimerTaskMemory()	33
6.6.3 Variable Documentation	33
6.6.3.1 g_xMscTaskHandle	33
6.6.3.2 g_xRecordTaskHandle	34
6.6.3.3 g_xSemMassStorage	34
6.6.3.4 g_xSemRecord	34
6.7 app_tasks.h	34
6.8 include/defs.h File Reference	35
6.8.1 Macro Definition Documentation	36
6.8.1.1 CONFIG_FILE	36
6.8.1.2 CONTROL_LED_ENABLED	36
6.8.1.3 DEBUG_ENABLED	36
6.8.1.4 DEFAULT_BAUDRATE	36
6.8.1.5 DEFAULT_DATA_BITS	37
6.8.1.6 DEFAULT_FREE_SPACE	37
6.8.1.7 DEFAULT_MAX_FILESIZE	37
6.8.1.8 DEFAULT_PARITY	37
6.8.1.9 DEFAULT_STOP_BITS	37
6.8.1.10 INFO_ENABLED	37
6.8.1.11 MSC_ENABLED	37
6.8.1.12 MSC_STACK_SIZE	38
6.8.1.13 NOT_IMPLEMENTED	38
6.8.1.14 PWRLOSS_DET_ACTIVE_IN_TIME	38
6.8.1.15 PWRLOSS_DET_PRIO	38
6.8.1.16 PWRLOSS_DETECTION_ENABLED	38
6.8.1.17 PWRLOSS_TEST_GPIOS	38
6.8.1.18 PWRLOSS_TIMER_PRIO	38
6.8.1.19 RECORD_STACK_SIZE	38
6.8.1.20 TAU5	39
6.8.1.21 TEMPERATURE_MEAS_ENABLED	39
6.8.1.22 UART_FIFO_ENABLED	39
6.8.1.23 UART_FIFO_LENGTH	39
6.8.1.24 UART_PRINT_ENABLED	39
6.8.1.25 UART_RECEIVE_PRIO	39
6.9 defs.h	40
6.10 include/error.h File Reference	41
6.10.1 Macro Definition Documentation	41
6.10.1.1 ERROR_ADMA	41

6.10.1.2 ERROR_CLOSE	42
6.10.1.3 ERROR_CONFIG	42
6.10.1.4 ERROR_FILESYSTEM	42
6.10.1.5 ERROR_IRTC	42
6.10.1.6 ERROR_NONE	42
6.10.1.7 ERROR_OPEN	42
6.10.1.8 ERROR_OUT_OF_CYCLE	42
6.10.1.9 ERROR_READ	42
6.10.1.10 ERROR_RECORD	43
6.10.1.11 ERROR_UNKNOWN	43
6.10.2 Typedef Documentation	43
6.10.2.1 error_t	43
6.10.3 Function Documentation	43
6.10.3.1 ERR_HandleError()	43
6.10.3.2 ERR_Init()	43
6.10.3.3 ERR_SetState()	43
6.11 error.h	44
6.12 include/led.h File Reference	44
6.12.1 Macro Definition Documentation	46
6.12.1.1 BACKUP_POWER_LED_PIN	46
6.12.1.2 BACKUP_POWER_LED_PORT	46
6.12.1.3 ERROR_LED_PIN_RECORD	46
6.12.1.4 ERROR_LED_PORT	46
6.12.1.5 FLUSH_LED_PORT	46
6.12.1.6 MEMORY_LOW_LED_PIN	46
6.12.1.7 MEMORY_LOW_LED_PORT	47
6.12.1.8 RECORD_LED_PIN	47
6.12.1.9 RECORD_LED_PIN_FLUSH	47
6.12.1.10 RECORD_LED_PORT	47
6.12.2 Function Documentation	47
6.12.2.1 LED_ClearSignalFlush()	47
6.12.2.2 LED_SetHigh()	47
6.12.2.3 LED_SetLow()	48
6.12.2.4 LED_SignalBackUpPowerAvailable()	48
6.12.2.5 LED_SignalError()	48
6.12.2.6 LED_SignalFlush()	48
6.12.2.7 LED_SignalLowMemory()	48
6.12.2.8 LED_SignalReady()	49
6.12.2.9 LED_SignalRecording()	49
6.12.2.10 LED_SignalRecordingStop()	49
6.13 led.h	49
6.14 include/mass_storage.h File Reference	50

6.14.1 Function Documentation	50
6.14.1.1 MSC_DeviceMscApp()	50
6.14.1.2 MSC_DeviceMscAppTask()	50
6.15 mass_storage.h	51
6.16 include/parser.h File Reference	51
6.16.1 Enumeration Type Documentation	52
6.16.1.1 REC_version_t	52
6.16.2 Function Documentation	53
6.16.2.1 PARSER_ClearConfig()	53
6.16.2.2 PARSER_GetBaudrate()	53
6.16.2.3 PARSER_GetConfig()	53
6.16.2.4 PARSER_GetDataBits()	53
6.16.2.5 PARSER_GetFileSize()	54
6.16.2.6 PARSER_GetFreeSpaceLimitMB()	54
6.16.2.7 PARSER_GetMaxBytes()	54
6.16.2.8 PARSER_GetParity()	54
6.16.2.9 PARSER_GetStopBits()	55
6.16.2.10 PARSER_GetVersion()	55
6.16.2.11 PARSER_ParseBaudrate()	55
6.16.2.12 PARSER_ParseDataBits()	55
6.16.2.13 PARSER_ParseFileSize()	55
6.16.2.14 PARSER_ParseFreeSpace()	56
6.16.2.15 PARSER_ParseParity()	56
6.16.2.16 PARSER_ParseStopBits()	56
6.17 parser.h	57
6.18 include/pwrloss_det.h File Reference	58
6.18.1 Macro Definition Documentation	59
6.18.1.1 CTIMER	59
6.18.1.2 CTIMER_CLK_FREQ	59
6.18.1.3 CTIMER_EMT0_OUT	59
6.18.1.4 CTIMER_MAT0_OUT	59
6.18.1.5 LPCMP_BASE	59
6.18.1.6 LPCMP_DAC_CHANNEL	60
6.18.1.7 LPCMP_IRQ_ID	60
6.18.1.8 LPCMP_USER_CHANNEL	60
6.18.1.9 SPC_BASE	60
6.18.2 Function Documentation	60
6.18.2.1 PWRLOSS_DetectionInit()	60
6.19 pwrloss_det.h	61
6.20 include/record.h File Reference	61
6.20.1 Macro Definition Documentation	63
6.20.1.1 FLUSH_TIMEOUT_TICKS	63

6.20.2 Function Documentation	63
6.20.2.1 CONSOLELOG_CheckFileSystem()	63
6.20.2.2 CONSOLELOG_ClearTransferredBytes()	63
6.20.2.3 CONSOLELOG_CreateDirectory()	63
6.20.2.4 CONSOLELOG_CreateFile()	63
6.20.2.5 CONSOLELOG_Deinit()	64
6.20.2.6 CONSOLELOG_Flush()	64
6.20.2.7 CONSOLELOG_GetFlushCompleted()	64
6.20.2.8 CONSOLELOG_GetFreeSpaceMB()	64
6.20.2.9 CONSOLELOG_GetMaxBytes()	65
6.20.2.10 CONSOLELOG_GetTransferredBytes()	65
6.20.2.11 CONSOLELOG_Init()	65
6.20.2.12 CONSOLELOG_PowerLossFlush()	65
6.20.2.13 CONSOLELOG_ProccessConfigFile()	65
6.20.2.14 CONSOLELOG_ReadConfig()	66
6.20.2.15 CONSOLELOG_Recording()	66
6.21 record.h	66
6.22 include/task_switching.h File Reference	67
6.22.1 Function Documentation	68
6.22.1.1 USB_State()	68
6.22.2 Variable Documentation	68
6.22.2.1 g_msc	68
6.23 task_switching.h	69
6.24 include/temperature.h File Reference	69
6.24.1 Function Documentation	69
6.24.1.1 Read()	69
6.24.1.2 TMP_GetTemperature()	70
6.24.1.3 TMP_Init()	70
6.24.1.4 Write()	70
6.25 temperature.h	71
6.26 include/time.h File Reference	71
6.26.1 Macro Definition Documentation	72
6.26.1.1 LPI2C_DMA_BASEADDR	72
6.26.2 Function Documentation	72
6.26.2.1 TIME_InitIRTC()	72
6.27 time.h	72
6.28 include/uart.h File Reference	73
6.28.1 Macro Definition Documentation	73
6.28.1.1 LPUART3_CLK_FREQ	73
6.28.2 Function Documentation	73
6.28.2.1 UART_Deinit()	73
6.28.2.2 UART_Disable()	74

6.28.2.3 UART_Enable()	74
6.28.2.4 UART_Init()	74
6.28.2.5 UART_Print()	74
6.29 uart.h	74
Index	77

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

DS3231 Real-Time Clock Driver	7
---	-------------------

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

REC_config_t	Configuration structure for the recording system	13
RTC_date_t	Structure for Keeping Date	15
RTC_time_t	Structure for Keeping Time	16

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

drivers/ rtc_ds3231.c	17
drivers/ rtc_ds3231.h	20
include/ app_init.h	30
include/ app_tasks.h	31
include/ defs.h	35
include/ error.h	41
include/ led.h	44
include/ mass_storage.h	50
include/ parser.h	51
include/ pwrloss_det.h	58
include/ record.h	61
include/ task_switching.h	67
include/ temperature.h	69
include/ time.h	71
include/ uart.h	73

Chapter 4

Topic Documentation

4.1 DS3231 Real-Time Clock Driver

Real-Time Circuit Related Functions.

Functions

- uint8_t [RTC_Init](#) (void)
Initialize The RTC DS3231.
- void [RTC_Deinit](#) (void)
De-Initialize The RTC DS3231.
- static uint8_t [RTC_ConvertToBCD](#) (uint8_t dec)
Converts Numbers From Decimal Base To BCD Base.
- static uint8_t [RTC_ConvertToDEC](#) (uint8_t bcd)
Converts Numbers From BCD Base To Decimal Base.
- uint8_t [RTC_GetState](#) (void)
This Function Checks If The Oscillator Is Still Running.
- void [RTC_SetOscState](#) (RTC_osc_state_t state)
Sets The Oscillator Stop Flag (OSF).
- uint8_t [RTC_Write](#) (uint8_t regAddress, uint8_t val)
Writes Value Into RTC Registers.
- uint8_t [RTC_Read](#) (uint8_t regAddress)
Reads Value From RTC Register.
- void [RTC_SetTime](#) (RTC_time_t *pTime)
Sets Time.
- void [RTC_GetTime](#) (RTC_time_t *pTime)
Gets Time.
- void [RTC_SetDate](#) (RTC_date_t *pDate)
Sets Date.
- void [RTC_GetDate](#) (RTC_date_t *pDate)
Gets Date.
- void [RTC_SetInterruptMode](#) (RTC_interrupt_mode_t mode)
Sets The INTCN Bit in Control Register.
- void [RTC_CtrlAlarm1](#) (uint8_t enable)
Enables/Disables The Alarm 1.

- void [RTC_ClearFlagAlarm1](#) (void)
Clears The A1F Flag in Control Register.
- void [RTC_CtrlAlarm2](#) (uint8_t enable)
Enables/Disables The Alarm 2.
- void [RTC_ClearFlagAlarm2](#) (void)
Clears The A1F Flag in Control Register.
- void [RTC_SetTimeDefault](#) (RTC_time_t *pTime)
Sets Time To Default.
- void [RTC_SetDateDefault](#) (RTC_date_t *pDate)
Sets Date To Default.

4.1.1 Detailed Description

Real-Time Circuit Related Functions.

4.1.2 Function Documentation

4.1.2.1 [RTC_ClearFlagAlarm1\(\)](#)

```
void RTC_ClearFlagAlarm1 (
    void )
```

Clears The A1F Flag in Control Register.

4.1.2.2 [RTC_ClearFlagAlarm2\(\)](#)

```
void RTC_ClearFlagAlarm2 (
    void )
```

Clears The A1F Flag in Control Register.

4.1.2.3 [RTC_ConvertToBCD\(\)](#)

```
static uint8_t RTC_ConvertToBCD (
    uint8_t dec) [static]
```

Converts Numbers From Decimal Base To BCD Base.

Parameters

<i>dec</i>	Decimal Number.
------------	-----------------

Returns

Number in Binary Coded Decimal.

4.1.2.4 [RTC_ConvertToDEC\(\)](#)

```
static uint8_t RTC_ConvertToDEC (
    uint8_t bcd) [static]
```

Converts Numbers From BCD Base To Decimal Base.

Parameters

<i>bcd</i>	Binary-Coded Decimal Number.
------------	------------------------------

Returns

Number in Decimal Base.

4.1.2.5 RTC_CtrlAlarm1()

```
void RTC_CtrlAlarm1 (  
    uint8_t enable)
```

Enables/Disables The Alarm 1.

Parameters

<i>enable</i>	Specifies If The Alarm Will Be Enabled or Not.
---------------	--

4.1.2.6 RTC_CtrlAlarm2()

```
void RTC_CtrlAlarm2 (  
    uint8_t enable)
```

Enables/Disables The Alarm 2.

Parameters

<i>enable</i>	Specifies If The Alarm Will Be Enabled or Not.
---------------	--

4.1.2.7 RTC_Deinit()

```
void RTC_Deinit (  
    void )
```

De-Initialize The RTC DS3231.

Pins Should Also Be De-Initialised Lately.

Parameters

<i>void</i>	
-------------	--

4.1.2.8 RTC_GetDate()

```
void RTC_GetDate (  
    RTC_date_t * pDate)
```

Gets Date.

Parameters

<i>pDate</i>	Pointer To Date Structure.
--------------	----------------------------

4.1.2.9 RTC_GetState()

```
uint8_t RTC_GetState (
    void )
```

This Function Checks If The Oscillator Is Still Running.

Returns

Function Returns 1 If The Oscillator Is Running. If The Oscillator Has Stopped Returns 0.

4.1.2.10 RTC_GetTime()

```
void RTC_GetTime (
    RTC_time_t * pTime)
```

Gets Time.

Parameters

<i>pTime</i>	Pointer To Time Structure.
--------------	----------------------------

4.1.2.11 RTC_Init()

```
uint8_t RTC_Init (
    void )
```

Initialize The RTC DS3231.

Parameters

<i>void</i>	
-------------	--

Before Calling of RTC_Init Function Is Important To Prepare I2C (To Keep The Driver As Universal As Possible, e.g To Use I2C With DMA, Interrupt/Polling Mode,...). enableRoundRobinArbitration = false; enableHaltOnError = true; enableContinuousLinkMode = false; enableDebugMode = false;

4.1.2.12 RTC_Read()

```
uint8_t RTC_Read (
    uint8_t regAddress)
```

Reads Value From RTC Register.

Parameters

<i>regAddress</i>	Address of Register From Which Will Be Read.
-------------------	--

Returns

Read Value.

4.1.2.13 RTC_SetDate()

```
void RTC_SetDate (  
    RTC_date_t * pDate)
```

Sets Date.

Parameters

<i>pDate</i>	Pointer To Date Structure.
--------------	----------------------------

4.1.2.14 RTC_SetDateDefault()

```
void RTC_SetDateDefault (  
    RTC_date_t * pDate)
```

Sets Date To Default.

Parameters

<i>pDate</i>	Pointer to Date Structure That Will Be Configured To Default.
--------------	---

4.1.2.15 RTC_SetInterruptMode()

```
void RTC_SetInterruptMode (  
    RTC_interrupt_mode_t mode)
```

Sets The INTCN Bit in Control Register.

Parameters

<i>mode</i>	Interrupt Mode.
-------------	-----------------

4.1.2.16 RTC_SetOscState()

```
void RTC_SetOscState (  
    RTC_osc_state_t state)
```

Sets The Oscillator Stop Flag (OSF).

Parameters

<i>state</i>	State of The Flag.
--------------	--------------------

4.1.2.17 RTC_SetTime()

```
void RTC_SetTime (
    RTC_time_t * pTime)
```

Sets Time.

Parameters

<i>pTime</i>	Pointer To Time Structure.
--------------	----------------------------

4.1.2.18 RTC_SetTimeDefault()

```
void RTC_SetTimeDefault (
    RTC_time_t * pTime)
```

Sets Time To Default.

Parameters

<i>pTime</i>	Pointer to Time Structure That Will Be Configured To Default.
--------------	---

4.1.2.19 RTC_Write()

```
uint8_t RTC_Write (
    uint8_t regAddress,
    uint8_t val)
```

Writes Value Into RTC Registers.

Parameters

<i>regAddress</i>	Address of Register To Which Will Be Value Written.
<i>val</i>	Value That Will Be Writen Into Register.

Returns

void

Chapter 5

Class Documentation

5.1 REC_config_t Struct Reference

Configuration structure for the recording system.

```
#include <parser.h>
```

Public Attributes

- [REC_version_t](#) version
- uint32_t [baudrate](#)
- lpuart_stop_bit_count_t [stop_bits](#)
- lpuart_data_bits_t [data_bits](#)
- lpuart_parity_mode_t [parity](#)
- uint32_t [size](#)
- uint32_t [max_bytes](#)
- uint32_t [free_space_limit_mb](#)

5.1.1 Detailed Description

Configuration structure for the recording system.

Structure Holds The Configuration Parameters Required For Initializing The Recording System, Including The Board Version and Baud Rate...

5.1.2 Member Data Documentation

5.1.2.1 baudrate

```
uint32_t REC_config_t::baudrate
```

Desired Baudrate

5.1.2.2 data_bits

```
lpuart_data_bits_t REC_config_t::data_bits
```

Number of Data Bit

5.1.2.3 free_space_limit_mb

```
uint32_t REC_config_t::free_space_limit_mb
```

Defines The Threshold Level of Free Memory on The SD card, Below Which The Lack of Memory is Indicated.

5.1.2.4 max_bytes

```
uint32_t REC_config_t::max_bytes
```

Number of Bytes Between LED Signal

5.1.2.5 parity

```
lpuart_parity_mode_t REC_config_t::parity
```

Parity Bit

5.1.2.6 size

```
uint32_t REC_config_t::size
```

Maximum File Size Maximal Log. Time In Per File

5.1.2.7 stop_bits

```
lpuart_stop_bit_count_t REC_config_t::stop_bits
```

Number of Stop Bit

5.1.2.8 version

```
REC_version_t REC_config_t::version
```

NXP Board That Will Be Recorded

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

- [include/parser.h](#)

5.2 RTC_date_t Struct Reference

Structure for Keeping Date.

```
#include <rtc_ds3231.h>
```

Public Attributes

- [uint8_t date](#)
- [uint8_t day](#)
- [uint8_t month](#)
- [uint8_t year](#)

5.2.1 Detailed Description

Structure for Keeping Date.

5.2.2 Member Data Documentation

5.2.2.1 date

```
uint8_t RTC_date_t::date
```

Date (1 [SUNDAY] .. 31 [SATURDAY])

5.2.2.2 day

```
uint8_t RTC_date_t::day
```

Day (1..7)

5.2.2.3 month

```
uint8_t RTC_date_t::month
```

Month (1..12)

5.2.2.4 year

```
uint8_t RTC_date_t::year
```

Year (From Base Year 2000)

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

- [drivers/rtc_ds3231.h](#)

5.3 RTC_time_t Struct Reference

Structure for Keeping Time.

```
#include <rtc_ds3231.h>
```

Public Attributes

- `uint8_t` [format](#)
- `uint8_t` [sec](#)
- `uint8_t` [min](#)
- `uint8_t` [hrs](#)

5.3.1 Detailed Description

Structure for Keeping Time.

5.3.2 Member Data Documentation

5.3.2.1 format

```
uint8_t RTC_time_t::format
```

Time Format (e.g. AM/PM or 24H Cycle)

5.3.2.2 hrs

```
uint8_t RTC_time_t::hrs
```

Hours (0..12/0..24 Based on Parameter Time Format)

5.3.2.3 min

```
uint8_t RTC_time_t::min
```

Minutes (0..60)

5.3.2.4 sec

```
uint8_t RTC_time_t::sec
```

Seconds (0..60)

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

- [drivers/rtc_ds3231.h](#)

Chapter 6

File Documentation

6.1 drivers/rtc_ds3231.c File Reference

```
#include "rtc_ds3231.h"
```

Macros

- #define [I2C_DATA_LENGTH](#) (2) /* MAX is 256 */
Length of I2C Rx Buffer.
- #define [BYTE_1](#) (0U)
- #define [BYTE_2](#) (1U)
- #define [CLK_STATE](#) (x)

Functions

- [AT_NONCACHEABLE_SECTION](#) (uint8_t g_aRxBuff[I2C_DATA_LENGTH])
Reception Buffer.
- [AT_NONCACHEABLE_SECTION](#) (lpi2c_master_edma_handle_t gEdmaHandle)
eDMA Driver Handle Used For Non-Blocking DMA Transfer.
- static void [lpi2c_callback](#) (LPI2C_Type *base, lpi2c_master_edma_handle_t *handle, status_t status, void *userData)
- uint8_t [RTC_Init](#) (void)
Initialize The RTC DS3231.
- void [RTC_Deinit](#) (void)
De-Initialize The RTC DS3231.
- static uint8_t [RTC_ConvertToBCD](#) (uint8_t dec)
- static uint8_t [RTC_ConvertToDEC](#) (uint8_t bcd)
- [RTC_osc_state_t](#) [RTC_GetState](#) (void)
This Function Checks If The Oscillator Is Still Running.
- void [RTC_SetOscState](#) ([RTC_osc_state_t](#) state)
Sets The Oscillator Stop Flag (OSF).
- uint8_t [RTC_Write](#) (uint8_t regAddress, uint8_t val)
Writes Value Into RTC Registers.
- uint8_t [RTC_Read](#) (uint8_t regAddress)

- *Reads Value From RTC Register.*
- void [RTC_SetTime](#) ([RTC_time_t](#) *pTime)
- *Sets Time.*
- void [RTC_GetTime](#) ([RTC_time_t](#) *pTime)
- *Gets Time.*
- void [RTC_SetDate](#) ([RTC_date_t](#) *pDate)
- *Sets Date.*
- void [RTC_GetDate](#) ([RTC_date_t](#) *pDate)
- *Gets Date.*
- void [RTC_SetInterruptMode](#) ([RTC_interrupt_mode_t](#) mode)
- *Sets The INTCN Bit in Control Register.*
- void [RTC_CtrlAlarm1](#) ([uint8_t](#) enable)
- *Enables/Disables The Alarm 1.*
- void [RTC_ClearFlagAlarm1](#) (void)
- *Clears The A1F Flag in Control Register.*
- void [RTC_CtrlAlarm2](#) ([uint8_t](#) enable)
- *Enables/Disables The Alarm 2.*
- void [RTC_ClearFlagAlarm2](#) (void)
- *Clears The A1F Flag in Control Register.*
- void [RTC_SetTimeDefault](#) ([RTC_time_t](#) *pTime)
- *Sets Time To Default.*
- void [RTC_SetDateDefault](#) ([RTC_date_t](#) *pDate)
- *Sets Date To Default.*

Variables

- [edma_handle_t](#) [gEdmaTxHandle](#)
- *Tx eDMA Handle.*
- [edma_handle_t](#) [gEdmaRxHandle](#)
- *Rx eDMA Handle.*
- volatile bool [gCompletionFlag](#) = false
- *Flag Indicating Whether The Transfer Has Finished.*
- [lpi2c_master_transfer_t](#) [xfer](#) = {0}
- *Transfer Descriptor for I2C Communication.*

6.1.1 Macro Definition Documentation

6.1.1.1 BYTE_1

```
#define BYTE_1 (0U)
```

6.1.1.2 BYTE_2

```
#define BYTE_2 (1U)
```

6.1.1.3 CLK_STATE

```
#define CLK_STATE(  
    x)
```

Value:

```
(uint8_t)((x >> 7) & 0x1)
```

6.1.1.4 I2C_DATA_LENGTH

```
#define I2C_DATA_LENGTH (2) /* MAX is 256 */
```

Length of I2C Rx Buffer.

6.1.2 Function Documentation

6.1.2.1 AT_NONCACHEABLE_SECTION() [1/2]

```
AT_NONCACHEABLE_SECTION (  
    lpi2c_master_edma_handle_t gEdmaHandle)
```

eDMA Driver Handle Used For Non-Blocking DMA Transfer.

6.1.2.2 AT_NONCACHEABLE_SECTION() [2/2]

```
AT_NONCACHEABLE_SECTION (  
    uint8_t g_aRxBuff[I2C_DATA_LENGTH])
```

Reception Buffer.

Transmission Buffer.

Must Be In Non-Cacheable Memory Due To Usage of DMA.

6.1.2.3 lpi2c_callback()

```
static void lpi2c_callback (  
    LPI2C_Type * base,  
    lpi2c_master_edma_handle_t * handle,  
    status_t status,  
    void * userData) [static]
```

6.1.2.4 RTC_ConvertToBCD()

```
static uint8_t RTC_ConvertToBCD (  
    uint8_t dec) [static]
```

6.1.2.5 RTC_ConvertToDEC()

```
static uint8_t RTC_ConvertToDEC (  
    uint8_t bcd) [static]
```

6.1.3 Variable Documentation

6.1.3.1 gCompletionFlag

```
volatile bool gCompletionFlag = false
```

Flag Indicating Whether The Transfer Has Finished.

6.1.3.2 gEdmaRxHandle

```
edma_handle_t gEdmaRxHandle
```

Rx eDMA Handle.

6.1.3.3 gEdmaTxHandle

```
edma_handle_t gEdmaTxHandle
```

Tx eDMA Handle.

6.1.3.4 xfer

```
lpi2c_master_transfer_t xfer = {0}
```

Transfer Descriptor for I2C Communication.

Used to Configure and Execute Data Transfer Operations With The DS3231 via LPI2C using EDMA.

6.2 drivers/rtc_ds3231.h File Reference

```
#include <stdint.h>  
#include "defs.h"  
#include "fsl_lpi2c.h"  
#include "fsl_lpi2c_edma.h"  
#include "fsl_edma.h"
```

Classes

- struct [RTC_time_t](#)
Structure for Keeping Time.
- struct [RTC_date_t](#)
Structure for Keeping Date.

Macros

- #define `E_FAULT` 1
General Error Return Code.
- #define `APP_SUCCESS` 0
Return Value If Operation Succeeded.
- #define `LPI2C_TX_DMA_CHANNEL` 0U
I2C DMA Channel For Transmission.
- #define `LPI2C_RX_DMA_CHANNEL` 1U
I2C DMA Channel For Reception.
- #define `LPI2C_TX_CHANNEL` kDma0RequestMuxLpFlexcomm2Tx
Connection Between DMA Channel 0 and LP_FLEXCOMM2 Tx.
- #define `LPI2C_RX_EDMA_CHANNEL` kDma0RequestMuxLpFlexcomm2Rx
Connection Between DMA Channel 0 and LP_FLEXCOMM2 Rx.
- #define `I2C_MASTER` ((LPI2C_Type *)LPI2C2_BASE)
Points To I2C Peripheral Unit (Specifically LPI2C2 Instance).
- #define `DS3231_A1IE` (0x00u)
Alarm Interrupt Enable Bits.
- #define `DS3231_A2IE` (0x1u)
- #define `DS3231_INTCN` (0x2u)
Interrupt Control Bit.
- #define `DS3231_A1F` (0x0u)
Alarm 1 & Alarm 2 Flags.
- #define `DS3231_A2F` (0x1u)
- #define `ALARM_DISABLED` (0x0u)
- #define `OSC_STOPPED` (0x00u)
Indicates That Oscillator Has Stopped & Time Has To Updated.
- #define `DS3231_ADDR_SEC` (0x00U)
Registers Addresses of DS3231.
- #define `DS3231_ADDR_MIN` (0x01U)
- #define `DS3231_ADDR_HRS` (0x02U)
- #define `DS3231_ADDR_DAY` (0x03U)
- #define `DS3231_ADDR_DATE` (0x04U)
- #define `DS3231_ADDR_MONTH` (0x05U)
- #define `DS3231_ADDR_YEAR` (0x06U)
- #define `DS3231_ADDR_CENT` (0x07U)
- #define `DS3231_REG_STATUS` (0x0Fu)
Address of Status Register.
- #define `DS3231_REG_CTRL` (0x0Eu)
Address of Control Register.
- #define `DS3231_ADDR_I2C` (0x68U)
Address of I2C Slave.
- #define `SUNDAY` (0x1)
Definition of Dates.
- #define `MONDAY` (0x2)
- #define `TUESDAY` (0x3)
- #define `WEDNESDAY` (0x4)
- #define `THURSDAY` (0x5)
- #define `FRIDAY` (0x6)
- #define `SATURDAY` (0x7)
- #define `TIM_CYCLE_12H` (0x0)
Definitions For Time Format Handling.

- #define TIM_CYCLE_12H_AM (0x0)
- #define TIM_CYCLE_12H_PM (0x1)
- #define TIM_CYCLE_24H (0x2)
- #define I2C_BAUDRATE 100000U

Application Configurable Items.

Enumerations

- enum RTC_osc_state_t { OSC_OK = 0 , OSC_INTERRUPTED }
- enum RTC_interrupt_mode_t { SQUARE_WAVE_INTERRUPT = 0 , ALARM_INTERRUPT }

An Enum to Capture The State of The Oscillator,.

Functions

- uint8_t RTC_Init (void)
Initialize The RTC DS3231.
- void RTC_Deinit (void)
De-Initialize The RTC DS3231.
- static uint8_t RTC_ConvertToBCD (uint8_t dec)
Converts Numbers From Decimal Base To BCD Base.
- static uint8_t RTC_ConvertToDEC (uint8_t bcd)
Converts Numbers From BCD Base To Decimal Base.
- uint8_t RTC_GetState (void)
This Function Checks If The Oscillator Is Still Running.
- void RTC_SetOscState (RTC_osc_state_t state)
Sets The Oscillator Stop Flag (OSF).
- uint8_t RTC_Write (uint8_t regAddress, uint8_t val)
Writes Value Into RTC Registers.
- uint8_t RTC_Read (uint8_t regAddress)
Reads Value From RTC Register.
- void RTC_SetTime (RTC_time_t *pTime)
Sets Time.
- void RTC_GetTime (RTC_time_t *pTime)
Gets Time.
- void RTC_SetDate (RTC_date_t *pDate)
Sets Date.
- void RTC_GetDate (RTC_date_t *pDate)
Gets Date.
- void RTC_SetInterruptMode (RTC_interrupt_mode_t mode)
Sets The INTCN Bit in Control Register.
- void RTC_CtrlAlarm1 (uint8_t enable)
Enables/Disables The Alarm 1.
- void RTC_ClearFlagAlarm1 (void)
Clears The A1F Flag in Control Register.
- void RTC_CtrlAlarm2 (uint8_t enable)
Enables/Disables The Alarm 2.
- void RTC_ClearFlagAlarm2 (void)
Clears The A1F Flag in Control Register.
- void RTC_SetTimeDefault (RTC_time_t *pTime)
Sets Time To Default.
- void RTC_SetDateDefault (RTC_date_t *pDate)
Sets Date To Default.

6.2.1 Macro Definition Documentation

6.2.1.1 ALARM_DISABLED

```
#define ALARM_DISABLED (0x0u)
```

6.2.1.2 APP_SUCCESS

```
#define APP_SUCCESS 0
```

Return Value If Operation Succeeded.

6.2.1.3 DS3231_A1F

```
#define DS3231_A1F (0x0u)
```

Alarm 1 & Alarm 2 Flags.

A Logic 1 in The Alarm 'x' Flag Bit Indicates That The Time Matched The Alarm 'x' Registers.

6.2.1.4 DS3231_A1IE

```
#define DS3231_A1IE (0x00u)
```

Alarm Interrupt Enable Bits.

6.2.1.5 DS3231_A2F

```
#define DS3231_A2F (0x1u)
```

6.2.1.6 DS3231_A2IE

```
#define DS3231_A2IE (0x1u)
```

6.2.1.7 DS3231_ADDR_CENT

```
#define DS3231_ADDR_CENT (0x07U)
```

Century Flag Bit

6.2.1.8 DS3231_ADDR_DATE

```
#define DS3231_ADDR_DATE (0x04U)
```

Day of Month Register

6.2.1.9 DS3231_ADDR_DAY

```
#define DS3231_ADDR_DAY (0x03U)
```

Day of Week rRegister

6.2.1.10 DS3231_ADDR_HRS

```
#define DS3231_ADDR_HRS (0x02U)
```

Hours Register

6.2.1.11 DS3231_ADDR_I2C

```
#define DS3231_ADDR_I2C (0x68U)
```

Address of I2C Slave.

6.2.1.12 DS3231_ADDR_MIN

```
#define DS3231_ADDR_MIN (0x01U)
```

Minutes Register

6.2.1.13 DS3231_ADDR_MONTH

```
#define DS3231_ADDR_MONTH (0x05U)
```

Month Register

6.2.1.14 DS3231_ADDR_SEC

```
#define DS3231_ADDR_SEC (0x00U)
```

Registers Addresses of DS3231.

Seconds Register

6.2.1.15 DS3231_ADDR_YEAR

```
#define DS3231_ADDR_YEAR (0x06U)
```

Year Register

6.2.1.16 DS3231_INTCN

```
#define DS3231_INTCN (0x2u)
```

Interrupt Control Bit.

6.2.1.17 DS3231_REG_CTRL

```
#define DS3231_REG_CTRL (0x0Eu)
```

Address of Control Register.

6.2.1.18 DS3231_REG_STATUS

```
#define DS3231_REG_STATUS (0x0Fu)
```

Address of Status Register.

6.2.1.19 E_FAULT

```
#define E_FAULT 1
```

General Error Return Code.

6.2.1.20 FRIDAY

```
#define FRIDAY (0x6)
```

Friday

6.2.1.21 I2C_BAUDRATE

```
#define I2C_BAUDRATE 100000U
```

Application Configurable Items.

Desired Baud Rate For I2C Bus.

Frequency - 100kHz (Up To 400kHz -> Defined By DS3231).

6.2.1.22 I2C_MASTER

```
#define I2C_MASTER ((LPI2C_Type *)LPI2C2_BASE)
```

Points To I2C Peripheral Unit (Specifically LPI2C2 Instance).

6.2.1.23 LPI2C_RX_DMA_CHANNEL

```
#define LPI2C_RX_DMA_CHANNEL 1U
```

I2C DMA Channel For Reception.

6.2.1.24 LPI2C_RX_EDMA_CHANNEL

```
#define LPI2C_RX_EDMA_CHANNEL kDma0RequestMuxLpFlexcomm2Rx
```

Connection Between DMA Channel 0 and LP_FLEXCOMM2 Rx.

6.2.1.25 LPI2C_TX_CHANNEL

```
#define LPI2C_TX_CHANNEL kDma0RequestMuxLpFlexcomm2Tx
```

Connection Between DMA Channel 0 and LP_FLEXCOMM2 Tx.

6.2.1.26 LPI2C_TX_DMA_CHANNEL

```
#define LPI2C_TX_DMA_CHANNEL 0U
```

I2C DMA Channel For Transmission.

6.2.1.27 MONDAY

```
#define MONDAY (0x2)
```

Monday

6.2.1.28 OSC_STOPPED

```
#define OSC_STOPPED (0x00u)
```

Indicates That Oscillator Has Stopped & Time Has To Updated.

6.2.1.29 SATURDAY

```
#define SATURDAY (0x7)
```

Saturday

6.2.1.30 SUNDAY

```
#define SUNDAY (0x1)
```

Definition of Dates.

From Sunday (0x1) To Saturday (0x7). Sunday

6.2.1.31 THURSDAY

```
#define THURSDAY (0x5)
```

Thursday

6.2.1.32 TIM_CYCLE_12H

```
#define TIM_CYCLE_12H (0x0)
```

Definitions For Time Format Handling.

12-Hour Format

6.2.1.33 TIM_CYCLE_12H_AM

```
#define TIM_CYCLE_12H_AM (0x0)
```

AM in 12-Hour Mode

6.2.1.34 TIM_CYCLE_12H_PM

```
#define TIM_CYCLE_12H_PM (0x1)
```

PM in 12-Hour Mode

6.2.1.35 TIM_CYCLE_24H

```
#define TIM_CYCLE_24H (0x2)
```

24-Hour Format

6.2.1.36 TUESDAY

```
#define TUESDAY (0x3)
```

Tuesday

6.2.1.37 WEDNESDAY

```
#define WEDNESDAY (0x4)
```

Wednesday

6.2.2 Enumeration Type Documentation

6.2.2.1 RTC_interrupt_mode_t

```
enum RTC_interrupt_mode_t
```

Enumerator

SQUARE_WAVE_INTERRUPT	
ALARM_INTERRUPT	

6.2.2.2 RTC_osc_state_t

```
enum RTC_osc_state_t
```

An Enum to Capture The State of The Oscillator,.

Enumerator

OSC_OK	RTC Oscillator Is OK
OSC_INTERRUPTED	RTC Oscillator Was Interrupted -> Need to Update the Time

6.3 rtc_ds3231.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  *      Author:      Tomas Dolak
00003  *      File Name:    rtc_DS3231.h
00004  *      Description:   Header File to RTC DS3231 Driver.
00005  *      Created on:    Aug 7, 2024
00006  *
00007  *      @author      Tomas Dolak
00008  *      @brief        Header File to RTC DS3231 Driver.
00009  *      @filename     rtc_DS3231.h
00010  */
00011
00012 #ifndef RTC_DS3231_H_
00013 #define RTC_DS3231_H_
00014
00015 /*****
00016  * Includes
00017  *****/
00018 #include <stdint.h>
00019
00020 #include "defs.h"
00021
00022 #include "fsl_lpi2c.h"
00023 #include "fsl_lpi2c_edma.h"
00024 #include "fsl_edma.h"
00025 /*****
00026  * Definitions
00027  *****/
00028
00032 #define E_FAULT                1
00033
00037 #define APP_SUCCESS            0
00038
00039
00043 #define LPI2C_TX_DMA_CHANNEL    0U
00044
00048 #define LPI2C_RX_DMA_CHANNEL    1U
00049
00053 #define LPI2C_TX_CHANNEL        kDma0RequestMuxLpFlexcomm2Tx
00054
00058 #define LPI2C_RX_EDMA_CHANNEL    kDma0RequestMuxLpFlexcomm2Rx
00059
00063 #define I2C_MASTER              ((LPI2C_Type *)LPI2C2_BASE)
00064
00068 #define DS3231_A1IE            (0x00u)
00069 #define DS3231_A2IE            (0x1u)
00073 #define DS3231_INTCN            (0x2u)
00078 #define DS3231_A1F            (0x0u)
00079 #define DS3231_A2F            (0x1u)

```

```

00080
00081 #define ALARM_DISABLED (0x0u)
00085 #define OSC_STOPPED (0x00u)
00086
00090 #define DS3231_ADDR_SEC (0x00U)
00091 #define DS3231_ADDR_MIN (0x01U)
00092 #define DS3231_ADDR_HRS (0x02U)
00093 #define DS3231_ADDR_DAY (0x03U)
00094
00095 #define DS3231_ADDR_DATE (0x04U)
00096 #define DS3231_ADDR_MONTH (0x05U)
00097 #define DS3231_ADDR_YEAR (0x06U)
00098 #define DS3231_ADDR_CENT (0x07U)
00099
00103 #define DS3231_REG_STATUS (0x0Fu)
00107
00108 #define DS3231_REG_CTRL (0x0Eu)
00112 #define DS3231_ADDR_I2C (0x68U) //<! The Slave Address Byte Contains In 7-bit: 1101000
00113
00118 #define SUNDAY (0x1)
00119 #define MONDAY (0x2)
00120 #define TUESDAY (0x3)
00121 #define WEDNESDAY (0x4)
00122 #define THURSDAY (0x5)
00123 #define FRIDAY (0x6)
00124 #define SATURDAY (0x7)
00125
00129 #define TIM_CYCLE_12H (0x0)
00130 #define TIM_CYCLE_12H_AM (0x0)
00131 #define TIM_CYCLE_12H_PM (0x1)
00132 #define TIM_CYCLE_24H (0x2)
00133
00134
00138
00139
00144 #define I2C_BAUDRATE 100000U
00145
00146 /*****
00147  * Structures
00148  *****/
00152 typedef struct
00153 {
00154     uint8_t format;
00155     uint8_t sec;
00156     uint8_t min;
00157     uint8_t hrs;
00158 } RTC_time_t;
00159
00160
00164 typedef struct
00165 {
00166     uint8_t date;
00167     uint8_t day;
00168     uint8_t month;
00169     uint8_t year;
00170 } RTC_date_t;
00171
00172
00176 typedef enum
00177 {
00178     OSC_OK = 0,
00179     OSC_INTERRUPTED
00180 } RTC_osc_state_t;
00181
00182
00183 /*
00184  * @brief An Enum For Interrupt Mode.
00185  */
00186 typedef enum
00187 {
00188     SQUARE_WAVE_INTERRUPT = 0,
00189     ALARM_INTERRUPT
00190 } RTC_interrupt_mode_t;
00191
00192 /*****
00193  * Prototypes
00194  *****/
00195
00201
00209 uint8_t RTC_Init(void);
00210
00216 void RTC_Deinit(void);
00217
00223 static uint8_t RTC_ConvertToBCD(uint8_t dec);
00224
00230 static uint8_t RTC_ConvertToDEC(uint8_t bcd);
00231

```

```

00237 uint8_t RTC_GetState(void);
00238
00243 void RTC_SetOscState(RTC_osc_state_t state);
00244
00251 uint8_t RTC_Write(uint8_t regAddress, uint8_t val);
00252
00258 uint8_t RTC_Read(uint8_t regAddress);
00259
00260
00265 void RTC_SetTime(RTC_time_t *pTime);
00266
00271 void RTC_GetTime(RTC_time_t *pTime);
00272
00277 void RTC_SetDate(RTC_date_t *pDate);
00278
00283 void RTC_GetDate(RTC_date_t *pDate);
00284
00289 void RTC_SetInterruptMode(RTC_interrupt_mode_t mode);
00290
00295 void RTC_CtrlAlarm1(uint8_t enable);
00296
00300 void RTC_ClearFlagAlarm1(void);
00301
00306 void RTC_CtrlAlarm2(uint8_t enable);
00307
00311 void RTC_ClearFlagAlarm2(void);
00312
00317 void RTC_SetTimeDefault(RTC_time_t *pTime);
00318
00323 void RTC_SetDateDefault(RTC_date_t *pDate);
00324 // end of RTC group
00326
00327 #endif /* RTC_DS3231_H */

```

6.4 include/app_init.h File Reference

```

#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "fsl_clock.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "led.h"
#include "uart.h"
#include "time.h"
#include "error.h"
#include "temperature.h"
#include "pwrloss_det.h"

```

Functions

- void [APP_InitBoard](#) (void)
Initializes Board Peripherals And Modules For Proper Functionality of The Logger.

6.4.1 Function Documentation

6.4.1.1 APP_InitBoard()

```

void APP_InitBoard (
    void )

```

Initializes Board Peripherals And Modules For Proper Functionality of The Logger.

6.5 app_init.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    init.c
00004  * Author:       Tomas Dolak
00005  * Date:         31.03.2025
00006  * Description:  Implements Datalogger Application.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          init.c
00013  * @author        Tomas Dolak
00014  * @date          31.03.2025
00015  * @brief         Implements Datalogger Application.
00016  * *****/
00017
00018 #ifndef APP_INIT_H_
00019 #define APP_INIT_H_
00020 /*****
00021  * Includes
00022  *****/
00023 /* NXP Board Specific */
00024 #include "fsl_device_registers.h"
00025 #include "fsl_debug_console.h"
00026 #include "fsl_clock.h"
00027
00028 #include "pin_mux.h"
00029 #include "clock_config.h"
00030 #include "board.h"
00031
00032 /* Application Includes */
00033 #include "led.h"           // Control LEDs Module
00034 #include "uart.h"         // UART Module
00035 #include "time.h"         // Time Module
00036 #include "error.h"        // Error Handling
00037 #include "temperature.h"  // Temperature Measurement Module
00038 #include "pwrloss_det.h"  // Power Loss Detection Module
00039
00040 /*****
00041  * Global Definitions
00042  *****/
00043
00044 /*****
00045  * Global Structures
00046  *****/
00047
00048 /*****
00049  * Function Prototypes
00050  *****/
00051
00052 void APP_InitBoard(void);
00053
00054
00055 #endif /* APP_INIT_H_ */

```

6.6 include/app_tasks.h File Reference

```

#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "timers.h"
#include <disk.h>

```

Macros

- #define [TASK_PRIO](#) (configMAX_PRIORITIES - 1)

Functions

- void [msc_task](#) (void *handle)
Task Responsible For Mass Storage Functionality in Device Mode.
- void [record_task](#) (void *handle)
Task Recording Serial Data.
- void [vApplicationGetIdleTaskMemory](#) (StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t **ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize)
Hook Function to Provide Memory For The Idle Task in FreeRTOS.
- void [vApplicationGetTimerTaskMemory](#) (StaticTask_t **ppxTimerTaskTCBBuffer, StackType_t **ppxTimerTaskStackBuffer, uint32_t *pulTimerTaskStackSize)
Hook Function to Provide Memory For The Timer Task in FreeRTOS.

Variables

- TaskHandle_t [g_xMscTaskHandle](#)
- TaskHandle_t [g_xRecordTaskHandle](#)
- SemaphoreHandle_t [g_xSemRecord](#)
- SemaphoreHandle_t [g_xSemMassStorage](#)

6.6.1 Macro Definition Documentation

6.6.1.1 TASK_Prio

```
#define TASK_Prio (configMAX_PRIORITIES - 1)
```

6.6.2 Function Documentation

6.6.2.1 msc_task()

```
void msc_task (
    void * handle)
```

Task Responsible For Mass Storage Functionality in Device Mode.

This Task Implements USB Mass Storage Class (MSC) Operations, Allowing The System to Act As a Mass Storage Device. It Handles Communication With The Host and Manages Read/Write Operations.

Parameters

<i>handle</i>	Pointer to The Device Handle Used For The USB Operations.
---------------	---

6.6.2.2 record_task()

```
void record_task (
    void * handle)
```

Task Recording Serial Data.

The Task Provides Data Reception, Data Processing (For Example, Adding Time Stamps) And Also Data Storage. In Case The Monitored Device Is Disconnected From The Data Logger, All The Buffered Data In RAM Is Stored On The Memory Card.

Parameters

<i>handle</i>	Pointer to The Device Handle Used For The USB Operations.
---------------	---

6.6.2.3 vApplicationGetIdleTaskMemory()

```
void vApplicationGetIdleTaskMemory (
    StaticTask_t ** ppxIdleTaskTCBBuffer,
    StackType_t ** ppxIdleTaskStackBuffer,
    uint32_t * pulIdleTaskStackSize)
```

Hook Function to Provide Memory For The Idle Task in FreeRTOS.

This Hook Function Provides The Memory Needed For The Idle Task, Which Is Statically Allocated When configSUPPORT_STATIC_ALLOCATION Is Set to 1. The FreeRTOS Scheduler Calls This Function to Get The Task Control Block (TCB) and Stack For The Idle Task.

Parameters

out	<i>ppxIdleTaskTCBBuffer</i>	Pointer to The TCB Buffer For The Idle Task.
out	<i>ppxIdleTaskStackBuffer</i>	Pointer to The Stack Buffer For The Idle Task.
out	<i>pulIdleTaskStackSize</i>	Pointer to a Variable Holding The Stack Size.

6.6.2.4 vApplicationGetTimerTaskMemory()

```
void vApplicationGetTimerTaskMemory (
    StaticTask_t ** ppxTimerTaskTCBBuffer,
    StackType_t ** ppxTimerTaskStackBuffer,
    uint32_t * pulTimerTaskStackSize)
```

Hook Function to Provide Memory For The Timer Task in FreeRTOS.

This Hook Function Provides The Memory Needed For The Timer Task, Which Is Statically Allocated When configSUPPORT_STATIC_ALLOCATION Is Set To 1 And configUSE_TIMERS Is Enabled. The FreeRTOS Scheduler Calls This Function to Get The Task Control Block (TCB) And Stack For The Timer Task.

Parameters

out	<i>ppxTimerTaskTCBBuffer</i>	Pointer to The TCB Buffer For The Timer Task.
out	<i>ppxTimerTaskStackBuffer</i>	Pointer to The Stack Buffer For The Timer Task.
out	<i>pulTimerTaskStackSize</i>	Pointer to a Variable Holding The Stack Size.

6.6.3 Variable Documentation**6.6.3.1 g_xMscTaskHandle**

```
TaskHandle_t g_xMscTaskHandle [extern]
```

6.6.3.2 g_xRecordTaskHandle

TaskHandle_t g_xRecordTaskHandle [extern]

6.6.3.3 g_xSemMassStorage

SemaphoreHandle_t g_xSemMassStorage [extern]

6.6.3.4 g_xSemRecord

SemaphoreHandle_t g_xSemRecord [extern]

6.7 app_tasks.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    app_tasks.c
00004  * Author:      Tomas Dolak
00005  * Date:        14.09.2024
00006  * Description:  Header File For Implementation of FreeRTOS Task.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         app_tasks.c
00013  * @author       Tomas Dolak
00014  * @date        14.09.2024
00015  * @brief       Header File For Implementation of FreeRTOS Task.
00016  * *****/
00017
00018 #ifndef APP_TASKS_H_
00019 #define APP_TASKS_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024
00025 /* FreeRTOS Include */
00026 #include "FreeRTOS.h"
00027 #include "task.h"
00028 #include "queue.h"
00029 #include "semphr.h"
00030 #include "timers.h"
00031
00032 /* Mass Storage Includes */
00033 #include <disk.h>
00034 /*****
00035  * Definitions
00036  *****/
00037 #define TASK_PRIO      (configMAX_PRIORITIES - 1) /*< Task Priorities.
00038
00039 /*****
00040  * Global Variables
00041  *****/
00042
00043 /* Task Handles */
00044 extern TaskHandle_t g_xMscTaskHandle;
00045
00046 extern TaskHandle_t g_xRecordTaskHandle;
00047
00048 /* Semaphores */
00049 extern SemaphoreHandle_t g_xSemRecord;
00050
00051 extern SemaphoreHandle_t g_xSemMassStorage;
00052
00053 /*****
00054  * Prototypes
00055  *****/

```

```

00056
00057
00067 void msc_task(void *handle);
00068
00078 void record_task(void *handle);
00079
00092 void vApplicationGetIdleTaskMemory(StaticTask_t **ppxIdleTaskTCBBuffer,
00093                                     StackType_t **ppxIdleTaskStackBuffer,
00094                                     uint32_t *pulIdleTaskStackSize);
00095
00108 void vApplicationGetTimerTaskMemory(StaticTask_t **ppxTimerTaskTCBBuffer,
00109                                     StackType_t **ppxTimerTaskStackBuffer,
00110                                     uint32_t *pulTimerTaskStackSize);
00111
00112 #endif /* APP_TASKS_H_ */

```

6.8 include/defs.h File Reference

```
#include <stdbool.h>
```

Macros

- #define `NOT_IMPLEMENTED` #error "This Feature Is Not Implemented!"
- #define `MSC_STACK_SIZE` ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
Defines The Stack Size For Mass Storage Task.
- #define `RECORD_STACK_SIZE` ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
Defines The Stack Size For Recording Task.
- #define `MSC_ENABLED` (true)
Enables/Disables Mass Storage Functionality.
- #define `INFO_ENABLED` (false)
Enables/Disables Debug Mode.
- #define `DEBUG_ENABLED` (false)
Enables/Disables Debug Mode.
- #define `UART_FIFO_ENABLED` (true)
Enables/Disables HW FIFO Queue on Application LPUART.
- #define `UART_FIFO_LENHT` (4u)
Defines The Size of HW FIFO Queue.
- #define `UART_PRINT_ENABLED` (false)
Enables/Disables Print of Received Bytes From Application LPUART To Console.
- #define `PWRLOSS_DETECTION_ENABLED` (true)
Enables/Disables Power Loss Detection.
- #define `PWRLOSS_TEST_GPIOS` (false)
Enables/Disables GPIO For Testing Power Loss Detection.
- #define `TAU5` 16UL
Constant 5 Tau, When Back-Up Power Capacitor is Charged To 99%.
- #define `PWRLOSS_DET_ACTIVE_IN_TIME TAU5`
Time Interval When Power Loss Detection Became Active.
- #define `UART_RECEIVE_PRIO` (6U)
Priority of LP_FLEXCOMM Interrupt (UART) For Rx Of Recorded Data.
- #define `PWRLOSS_DET_PRIO` (5U)
Priority of Comparator Interrupt For Power Loss Detection.
- #define `PWRLOSS_TIMER_PRIO` (7U)
Priority of Comparator Interrupt For Power Loss Detection.

- `#define TEMPERATURE_MEAS_ENABLED (false)`
Enables/Disables Temperature Recording.
- `#define CONTROL_LED_ENABLED (true)`
Enables/Disables Signaling By LEDs.
- `#define DEFAULT_MAX_FILESIZE 8192`
Default Maximal File Size If The Configuration File Could Not Be Read Properly.
- `#define CONFIG_FILE "config"`
Configuration File.
- `#define DEFAULT_BAUDRATE 230400UL`
Default Baud Rate If The Configuration File Could Not Be Read Properly.
- `#define DEFAULT_DATA_BITS kLPUART_EightDataBits`
Default Number of Data Bits If The Configuration File Could Not Be Read Properly.
- `#define DEFAULT_STOP_BITS kLPUART_OneStopBit`
Default Number of Stop Bits If The Configuration File Could Not Be Read Properly.
- `#define DEFAULT_PARITY kLPUART_ParityDisabled`
Default UART Parity If The Configuration File Could Not Be Read Properly.
- `#define DEFAULT_FREE_SPACE 50UL`
Defines The Threshold Level of Free Memory on The SD card, Below Which The Lack of Memory is Indicated.

6.8.1 Macro Definition Documentation

6.8.1.1 CONFIG_FILE

```
#define CONFIG_FILE "config"
```

Configuration File.

6.8.1.2 CONTROL_LED_ENABLED

```
#define CONTROL_LED_ENABLED (true)
```

Enables/Disables Signaling By LEDs.

6.8.1.3 DEBUG_ENABLED

```
#define DEBUG_ENABLED (false)
```

Enables/Disables Debug Mode.

If Debug Mode Is Enabled Then Extended Logs Are Printed Into Debug Console.

6.8.1.4 DEFAULT_BAUDRATE

```
#define DEFAULT_BAUDRATE 230400UL
```

Default Baud Rate If The Configuration File Could Not Be Read Properly.

6.8.1.5 DEFAULT_DATA_BITS

```
#define DEFAULT_DATA_BITS kLPUART_EightDataBits
```

Default Number of Data Bits If The Configuration File Could Not Be Read Properly.

6.8.1.6 DEFAULT_FREE_SPACE

```
#define DEFAULT_FREE_SPACE 50UL
```

Defines The Threshold Level of Free Memory on The SD card, Below Which The Lack of Memory is Indicated.

In Megabytes.

6.8.1.7 DEFAULT_MAX_FILESIZE

```
#define DEFAULT_MAX_FILESIZE 8192
```

Default Maximal File Size If The Configuration File Could Not Be Read Properly.

6.8.1.8 DEFAULT_PARITY

```
#define DEFAULT_PARITY kLPUART_ParityDisabled
```

Default UART Parity If The Configuration File Could Not Be Read Properly.

6.8.1.9 DEFAULT_STOP_BITS

```
#define DEFAULT_STOP_BITS kLPUART_OneStopBit
```

Default Number of Stop Bits If The Configuration File Could Not Be Read Properly.

6.8.1.10 INFO_ENABLED

```
#define INFO_ENABLED (false)
```

Enables/Disables Debug Mode.

If Debug Mode Is Enabled Then Extended Logs Are Printed Into Debug Console.

6.8.1.11 MSC_ENABLED

```
#define MSC_ENABLED (true)
```

Enables/Disables Mass Storage Functionality.

6.8.1.12 MSC_STACK_SIZE

```
#define MSC_STACK_SIZE ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
```

Defines The Stack Size For Mass Storage Task.

6.8.1.13 NOT_IMPLEMENTED

```
#define NOT_IMPLEMENTED #error "This Feature Is Not Implemented!"
```

6.8.1.14 PWRLOSS_DET_ACTIVE_IN_TIME

```
#define PWRLOSS_DET_ACTIVE_IN_TIME TAU5
```

Time Interval When Power Loss Detection Became Active.

In Seconds.

6.8.1.15 PWRLOSS_DET_Prio

```
#define PWRLOSS_DET_Prio (5U)
```

Priority of Comparator Interrupt For Power Loss Detection.

6.8.1.16 PWRLOSS_DETECTION_ENABLED

```
#define PWRLOSS_DETECTION_ENABLED (true)
```

Enables/Disables Power Loss Detection.

6.8.1.17 PWRLOSS_TEST_GPIOS

```
#define PWRLOSS_TEST_GPIOS (false)
```

Enables/Disables GPIO For Testing Power Loss Detection.

6.8.1.18 PWRLOSS_TIMER_Prio

```
#define PWRLOSS_TIMER_Prio (7U)
```

Priority of Comparator Interrupt For Power Loss Detection.

6.8.1.19 RECORD_STACK_SIZE

```
#define RECORD_STACK_SIZE ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
```

Defines The Stack Size For Recording Task.

6.8.1.20 TAU5

```
#define TAU5 16UL
```

Constant 5 Tau, When Back-Up Power Capacitor is Charged To 99%.

In Seconds (It's Actually 16s).

6.8.1.21 TEMPERATURE_MEAS_ENABLED

```
#define TEMPERATURE_MEAS_ENABLED (false)
```

Enables/Disables Temperature Recording.

6.8.1.22 UART_FIFO_ENABLED

```
#define UART_FIFO_ENABLED (true)
```

Enables/Disables HW FIFO Queue on Application LPUART.

6.8.1.23 UART_FIFO_LENHT

```
#define UART_FIFO_LENHT (4u)
```

Defines The Size of HW FIFO Queue.

6.8.1.24 UART_PRINT_ENABLED

```
#define UART_PRINT_ENABLED (false)
```

Enables/Disables Print of Received Bytes From Application LPUART To Console.

6.8.1.25 UART_RECEIVE_PRIO

```
#define UART_RECEIVE_PRIO (6U)
```

Priority of LP_FLEXCOMM Interrupt (UART) For Rx Of Recorded Data.

6.9 defs.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    defs.h
00004  * Author:       Tomas Dolak
00005  * Date:         22.09.2024
00006  * Description:  Header File Providing Definitions To Set Up The Project.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         defs.h
00013  * @author       Tomas Dolak
00014  * @date         22.09.2024
00015  * @brief        Header File Providing Definitions To Set Up The Project.
00016  * *****/
00017
00018 #ifndef DEFS_H_
00019 #define DEFS_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include <stdbool.h>
00025
00026 /*****
00027  * Definitions
00028  *****/
00029 #define NOT_IMPLEMENTED          #error "This Feature Is Not Implemented!"
00030
00031 #define MSC_STACK_SIZE           ((uint32_t) (5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
00032
00033 #define RECORD_STACK_SIZE       ((uint32_t) (5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
00034
00035 #define MSC_ENABLED              (true)
00036
00037 #define INFO_ENABLED             (false)
00038
00039 #define DEBUG_ENABLED            (false)
00040
00041 #define UART_FIFO_ENABLED       (true)
00042
00043 #define UART_FIFO_LENHT         (4u)
00044
00045 #define UART_PRINT_ENABLED       (false)
00046
00047 #define PWRLOSS_DETECTION_ENABLED (true)
00048
00049 #define PWRLOSS_TEST_GPIOS      (false)
00050
00051 #define TAU5                     16UL
00052
00053 #define PWRLOSS_DET_ACTIVE_IN_TIME TAU5
00054
00055 #define UART_RECEIVE_PRIO        (6U)
00056
00057 #define PWRLOSS_DET_PRIO         (5U)
00058
00059 #define PWRLOSS_TIMER_PRIO       (7U)
00060
00061 #define TEMPERATURE_MEAS_ENABLED (false)
00062
00063 #define CONTROL_LED_ENABLED      (true)
00064
00065 #define DEFAULT_MAX_FILESIZE     8192
00066
00067 #define CONFIG_FILE              "config"
00068
00069 #define DEFAULT_BAUDRATE         230400UL
00070
00071 #define DEFAULT_DATA_BITS        kLPUART_EightDataBits
00072
00073 #define DEFAULT_STOP_BITS        kLPUART_OneStopBit
00074
00075 #define DEFAULT_PARITY           kLPUART_ParityDisabled
00076
00077 #define DEFAULT_FREE_SPACE       50UL
00078
00079 #endif /* DEFS_H_ */

```

6.10 include/error.h File Reference

```
#include <stdint.h>
```

Macros

- #define [ERROR_NONE](#) 0x0000u
Return Code When a Successful Operation is Performed.
- #define [ERROR_IRTC](#) 0x0001u
Error Related To Internal RTC Integrated In MCXN947 MCU.
- #define [ERROR_RECORD](#) 0x0002u
Error Related To Data Logging.
- #define [ERROR_OPEN](#) 0x0005u
Error Related To Opening File.
- #define [ERROR_READ](#) 0x0006u
Error Related To Reading File.
- #define [ERROR_CLOSE](#) 0x0004u
Error Related To Closing File.
- #define [ERROR_FILESYSTEM](#) 0x0007u
Error Related To File System (e.g. Mount of File System, Setting Up File Meta-Data,...).
- #define [ERROR_CONFIG](#) 0x0008u
Error Related To Reading and Parsing Configuration File Stored On SDHC Card (config.txt).
- #define [ERROR_ADMA](#) 0x0003u
Error Related To Storing Data on SDHC Card Using ADMA (Advance DMA).
- #define [ERROR_OUT_OF_CYCLE](#) 0xFFFEu
Out Of The Main Cycle Error.
- #define [ERROR_UNKNOWN](#) 0xFFFFu
Occurrence Of an Unclassified Error.

Typedefs

- typedef uint16_t [error_t](#)

Functions

- void [ERR_HandleError](#) (void)
Base Function For Error Handling.
- void [ERR_Init](#) ()
- void [ERR_SetState](#) ([error_t](#) err)

6.10.1 Macro Definition Documentation

6.10.1.1 ERROR_ADMA

```
#define ERROR_ADMA 0x0003u
```

Error Related To Storing Data on SDHC Card Using ADMA (Advance DMA).

6.10.1.2 ERROR_CLOSE

```
#define ERROR_CLOSE 0x0004u
```

Error Related To Closing File.

6.10.1.3 ERROR_CONFIG

```
#define ERROR_CONFIG 0x0008u
```

Error Related To Reading and Parsing Configuration File Stored On SDHC Card (config.txt).

6.10.1.4 ERROR_FILESYSTEM

```
#define ERROR_FILESYSTEM 0x0007u
```

Error Related To File System (e.g. Mount of File System, Setting Up File Meta-Data,...).

6.10.1.5 ERROR_IRTC

```
#define ERROR_IRTC 0x0001u
```

Error Related To Internal RTC Integrated In MCXN947 MCU.

6.10.1.6 ERROR_NONE

```
#define ERROR_NONE 0x0000u
```

Return Code When a Successful Operation is Performed.

6.10.1.7 ERROR_OPEN

```
#define ERROR_OPEN 0x0005u
```

Error Related To Opening File.

6.10.1.8 ERROR_OUT_OF_CYCLE

```
#define ERROR_OUT_OF_CYCLE 0xFFFEu
```

Out Of The Main Cycle Error.

6.10.1.9 ERROR_READ

```
#define ERROR_READ 0x0006u
```

Error Related To Reading File.

6.10.1.10 ERROR_RECORD

```
#define ERROR_RECORD 0x0002u
```

Error Related To Data Logging.

6.10.1.11 ERROR_UNKNOWN

```
#define ERROR_UNKNOWN 0xFFFFu
```

Occurrence Of an Unclassified Error.

6.10.2 Typedef Documentation

6.10.2.1 error_t

```
typedef uint16_t error_t
```

6.10.3 Function Documentation

6.10.3.1 ERR_HandleError()

```
void ERR_HandleError (  
    void )
```

Base Function For Error Handling.

6.10.3.2 ERR_Init()

```
void ERR_Init ()
```

6.10.3.3 ERR_SetState()

```
void ERR_SetState (  
    error_t err)
```

6.11 error.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    error.h
00004  * Author:       Tomas Dolak
00005  * Date:         22.01.2025
00006  * Description:  Implements Error Handling Logic.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          error.h
00013  * @author        Tomas Dolak
00014  * @date          22.01.2025
00015  * @brief         Implements Error Handling Logic.
00016  * *****/
00017
00018 #ifndef ERROR_H_
00019 #define ERROR_H_
00020
00021 /*****
00022  * Includes
00023  * *****/
00024 #include <stdint.h>
00025 /*****
00026  * Definitions
00027  * *****/
00031 #define ERROR_NONE            0x0000u
00032
00036 #define ERROR_IRTC            0x0001u
00037
00041 #define ERROR_RECORD          0x0002u
00042
00046 #define ERROR_OPEN            0x0005u
00047
00051 #define ERROR_READ            0x0006u
00055 #define ERROR_CLOSE           0x0004u
00056
00060 #define ERROR_FILESYSTEM      0x0007u
00061
00065 #define ERROR_CONFIG          0x0008u
00066
00070 #define ERROR_ADMA            0x0003u
00071
00075 #define ERROR_OUT_OF_CYCLE    0xFFFEu
00076
00080 #define ERROR_UNKNOWN         0xFFFFu
00081 /*****
00082  * Structures
00083  * *****/
00084 typedef uint16_t error_t;
00085
00086 /*****
00087  * Prototypes
00088  * *****/
00092 void ERR_HandleError(void);
00093
00094 void ERR_Init();
00095
00096 void ERR_SetState(error_t err);
00097
00098
00099 #endif /* ERROR_H_ */

```

6.12 include/led.h File Reference

```

#include "fsl_gpio.h"
#include "board.h"

```

Macros

- #define [ERROR_LED_PORT](#) GPIO0
Error LED's Port Number.
- #define [ERROR_LED_PIN_RECORD](#) 0x09
Error LED's Pin 2.
- #define [MEMORY_LOW_LED_PORT](#) GPIO0
Low Memory LED's Port Number.
- #define [MEMORY_LOW_LED_PIN](#) 0x07
Error LED's Pin 1.
- #define [FLUSH_LED_PORT](#) GPIO0
Flush LED's Port Number.
- #define [RECORD_LED_PIN_FLUSH](#) 0x0D
Error LED's Pin 3.
- #define [RECORD_LED_PORT](#) GPIO2
Recording LED's Port Number.
- #define [RECORD_LED_PIN](#) 0x0B
Record LED's Pin.
- #define [BACKUP_POWER_LED_PORT](#) GPIO4
Recording LED's Port Number.
- #define [BACKUP_POWER_LED_PIN](#) 0x11
Record LED's Pin.

Functions

- void [LED_SetHigh](#) (GPIO_Type *port_base, uint32_t pin)
Sets Logic 1 on GPIO Pin.
- void [LED_SetLow](#) (GPIO_Type *port_base, uint32_t pin)
Sets Logic 0 on GPIO Pin.
- void [LED_SignalReady](#) (void)
Indicates Ready State of The Recording Device.
- void [LED_SignalRecording](#) (void)
Signals That Device Is Currently Receiving Bytes (Recording).
- void [LED_SignalRecordingStop](#) (void)
Signals That Device Is Stopped Receiving Bytes (Recording).
- void [LED_SignalBackUpPowerAvailable](#) (void)
Signals Configuration File Missing or Contains Unexpected Data.
- void [LED_SignalLowMemory](#) (void)
Signals That There Will Be Empty Space on SD Card.
- void [LED_SignalError](#) (void)
Signals Error During Recording.
- void [LED_SignalFlush](#) (void)
Signals That Flush Has Been Activated.
- void [LED_ClearSignalFlush](#) (void)
Clears Flush LED After UART Re-Initialization.

6.12.1 Macro Definition Documentation

6.12.1.1 BACKUP_POWER_LED_PIN

```
#define BACKUP_POWER_LED_PIN 0x11
```

Record LED's Pin.

P4_17

6.12.1.2 BACKUP_POWER_LED_PORT

```
#define BACKUP_POWER_LED_PORT GPIO4
```

Recording LED's Port Number.

Signals That Device Records.

6.12.1.3 ERROR_LED_PIN_RECORD

```
#define ERROR_LED_PIN_RECORD 0x09
```

Error LED's Pin 2.

P0_9

6.12.1.4 ERROR_LED_PORT

```
#define ERROR_LED_PORT GPIO0
```

Error LED's Port Number.

6.12.1.5 FLUSH_LED_PORT

```
#define FLUSH_LED_PORT GPIO0
```

Flush LED's Port Number.

6.12.1.6 MEMORY_LOW_LED_PIN

```
#define MEMORY_LOW_LED_PIN 0x07
```

Error LED's Pin 1.

P0_7

6.12.1.7 MEMORY_LOW_LED_PORT

```
#define MEMORY_LOW_LED_PORT GPIO0
```

Low Memory LED's Port Number.

6.12.1.8 RECORD_LED_PIN

```
#define RECORD_LED_PIN 0x0B
```

Record LED's Pin.

P2_11

6.12.1.9 RECORD_LED_PIN_FLUSH

```
#define RECORD_LED_PIN_FLUSH 0x0D
```

Error LED's Pin 3.

P0_13

6.12.1.10 RECORD_LED_PORT

```
#define RECORD_LED_PORT GPIO2
```

Recording LED's Port Number.

Signals That Device Records.

6.12.2 Function Documentation

6.12.2.1 LED_ClearSignalFlush()

```
void LED_ClearSignalFlush (  
    void )
```

Clears Flush LED After UART Re-Initialization.

6.12.2.2 LED_SetHigh()

```
void LED_SetHigh (  
    GPIO_Type * port_base,  
    uint32_t pin)
```

Sets Logic 1 on GPIO Pin.

Parameters

<i>port_base</i>	Pointer to GPIO Instance.
<i>pin</i>	Pin.

6.12.2.3 LED_SetLow()

```
void LED_SetLow (
    GPIO_Type * port_base,
    uint32_t pin)
```

Sets Logic 0 on GPIO Pin.

Parameters

<i>port_base</i>	Pointer to GPIO Instance.
<i>pin</i>	Pin.

6.12.2.4 LED_SignalBackUpPowerAvailable()

```
void LED_SignalBackUpPowerAvailable (
    void )
```

Signals Configuration File Missing or Contains Unexpected Data.

6.12.2.5 LED_SignalError()

```
void LED_SignalError (
    void )
```

Signals Error During Recording.

6.12.2.6 LED_SignalFlush()

```
void LED_SignalFlush (
    void )
```

Signals That Flush Has Been Activated.

6.12.2.7 LED_SignalLowMemory()

```
void LED_SignalLowMemory (
    void )
```

Signals That There Will Be Empty Space on SD Card.

6.12.2.8 LED_SignalReady()

```
void LED_SignalReady (
    void )
```

Indicates Ready State of The Recording Device.

6.12.2.9 LED_SignalRecording()

```
void LED_SignalRecording (
    void )
```

Signals That Device Is Currently Receiving Bytes (Recording).

6.12.2.10 LED_SignalRecordingStop()

```
void LED_SignalRecordingStop (
    void )
```

Signals That Device Is Stopped Receiving Bytes (Recording).

6.13 led.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    leds.c
00004  * Author:       Tomas Dolak
00005  * Date:         06.02.2025
00006  * Description:  Implements The Logic For LEDs Control.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          leds.c
00013  * @author        Tomas Dolak
00014  * @date          06.02.2025
00015  * @brief         Implements The Logic For LEDs Control.
00016  * *****/
00017
00018 #ifndef LED_H_
00019 #define LED_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include "fsl_gpio.h"
00025 #include "board.h"
00026 /*****
00027  * Functions Macros
00028  *****/
00029
00032 #define ERROR_LED_PORT      GPIO0
00033
00038 #define ERROR_LED_PIN_RECORD 0x09
00039
00040
00044 #define MEMORY_LOW_LED_PORT  GPIO0
00045
00050 #define MEMORY_LOW_LED_PIN   0x07
00051
00052
00056 #define FLUSH_LED_PORT      GPIO0
00057
00062 #define RECORD_LED_PIN_FLUSH 0x0D
```

```

00063
00068 #define RECORD_LED_PORT          GPIO2
00069
00074 #define RECORD_LED_PIN            0x0B
00075
00080 #define BACKUP_POWER_LED_PORT     GPIO4
00081
00086 #define BACKUP_POWER_LED_PIN      0x11
00087 /*****
00088  * Functions Definitions
00089  *****/
00090
00096 void LED_SetHigh(GPIO_Type *port_base, uint32_t pin);
00097
00103 void LED_SetLow(GPIO_Type *port_base, uint32_t pin);
00104
00108 void LED_SignalReady(void);
00109
00113 void LED_SignalRecording(void);
00114
00118 void LED_SignalRecordingStop(void);
00119
00123 void LED_SignalBackUpPowerAvailable(void);
00124
00128 void LED_SignalLowMemory(void);
00129
00133 void LED_SignalError(void);
00134
00138 void LED_SignalFlush(void);
00139
00143 void LED_ClearSignalFlush(void);
00144
00145 #endif /* LED_H_ */

```

6.14 include/mass_storage.h File Reference

```
#include "disk.h"
```

Functions

- void [MSC_DeviceMscApp](#) (void)
Process Extension to Mass Storage.
- void [MSC_DeviceMscAppTask](#) (void)
Handles Mass Storage Application.

6.14.1 Function Documentation

6.14.1.1 MSC_DeviceMscApp()

```
void MSC_DeviceMscApp (
    void )
```

Process Extension to Mass Storage.

6.14.1.2 MSC_DeviceMscAppTask()

```
void MSC_DeviceMscAppTask (
    void )
```

Handles Mass Storage Application.

Communication and Data Transport Is Handled By USB1_HS ISR.

6.15 mass_storage.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    mass_storage.h
00004  * Author:       Tomas Dolak
00005  * Date:         06.02.2025
00006  * Description:  Header File For USB Mass Storage.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          mass_storage.h
00013  * @author        Tomas Dolak
00014  * @date          06.02.2025
00015  * @brief         Header File For USB Mass Storage.
00016  * *****/
00017
00018 #ifndef MASS_STORAGE_H_
00019 #define MASS_STORAGE_H_
00020
00021
00022 /*****
00023  * Includes
00024  * *****/
00025 #include "disk.h"
00026
00027 /*****
00028  * Global Variables
00029  * *****/
00030
00031 /*****
00032  * Function Declarations
00033  * *****/
00034
00038 void MSC_DeviceMscApp(void);
00039
00044 void MSC_DeviceMscAppTask(void);
00045
00046 #endif /* MASS_STORAGE_H_ */

```

6.16 include/parser.h File Reference

```

#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include "fsl_lpuart.h"
#include "fsl_debug_console.h"
#include "error.h"
#include "defs.h"

```

Classes

- struct [REC_config_t](#)
Configuration structure for the recording system.

Enumerations

- enum [REC_version_t](#) { [WCT_UNKOWN](#) = 0 , [WCT_AUTOS1](#) , [WCT_AUTOS2](#) }
Enumeration of recording board versions.

Functions

- [REC_config_t](#) [PARSER_GetConfig](#) (void)
Returns Active Configuration.
- [REC_version_t](#) [PARSER_GetVersion](#) (void)
Returns the Version of The Device Being Recorded.
- [uint32_t](#) [PARSER_GetBaudrate](#) (void)
Returns the Baudrate of The Device Being Recorded.
- [uint32_t](#) [PARSER_GetFileSize](#) (void)
Returns the Maximal File Size.
- [lpuart_data_bits_t](#) [PARSER_GetDataBits](#) (void)
Returns The Number of Data Bits.
- [lpuart_parity_mode_t](#) [PARSER_GetParity](#) (void)
Returns The Parity.
- [lpuart_stop_bit_count_t](#) [PARSER_GetStopBits](#) (void)
Returns The Number of Stop Bits.
- [uint32_t](#) [PARSER_GetFreeSpaceLimitMB](#) (void)
Returns The Free Space Limit on SD Card For LED Signaling.
- [uint32_t](#) [PARSER_GetMaxBytes](#) (void)
Returns The Number of Maximal Bytes Between LED Blinking.
- [void](#) [PARSER_ClearConfig](#) (void)
Clears The Configuration To Default.
- [error_t](#) [PARSER_ParseBaudrate](#) (const char *chContent)
Parse Baud Rate From Configuration File.
- [error_t](#) [PARSER_ParseFileSize](#) (const char *chContent)
Parse Record File Size From Configuration File.
- [error_t](#) [PARSER_ParseParity](#) (const char *chContent)
Parse Parity From Configuration File.
- [error_t](#) [PARSER_ParseStopBits](#) (const char *chContent)
Parse The Number of Stop Bits From Configuration File.
- [error_t](#) [PARSER_ParseDataBits](#) (const char *chContent)
Parse The Number of Data Bits From Configuration File.
- [error_t](#) [PARSER_ParseFreeSpace](#) (const char *chContent)
Parse The Size of Free Space When Data Logger Will Signal To The User That Data Logger Is Running Out of Space.

6.16.1 Enumeration Type Documentation

6.16.1.1 REC_version_t

```
enum REC\_version\_t
```

Enumeration of recording board versions.

This enum defines the possible versions of the board for which the recording system is configured.

Note

Difference Between AUTOS1 and AUTOS2 Is In Baudrate.

Enumerator

WCT_UNKOWN	Unknown board version.
WCT_AUTOS1	AUTOS1 Reference Board.
WCT_AUTOS2	AUTOS2 Reference Board.

6.16.2 Function Documentation**6.16.2.1 PARSER_ClearConfig()**

```
void PARSER_ClearConfig (
    void )
```

Clears The Configuration To Default.

6.16.2.2 PARSER_GetBaudrate()

```
uint32_t PARSER_GetBaudrate (
    void )
```

Returns the Baudrate of The Device Being Recorded.

Returns

uint32_t Baud Rate of Recorded Device.

6.16.2.3 PARSER_GetConfig()

```
REC_config_t PARSER_GetConfig (
    void )
```

Returns Active Configuration.

This Function Returns The Global Configuration Structure That Contains The Settings For The Recording, Such as Baudrate, Version, And Other Relevant Parameters.

Returns

[REC_config_t](#) The Current Recording Configuration.

6.16.2.4 PARSER_GetDataBits()

```
lpuart_data_bits_t PARSER_GetDataBits (
    void )
```

Returns The Number of Data Bits.

Returns

lpuart_data_bits_t Number of Data Bits.

6.16.2.5 **PARSER_GetFileSize()**

```
uint32_t PARSER_GetFileSize (
    void )
```

Returns the Maximal File Size.

Returns

uint32_t Maximal File Size.

6.16.2.6 **PARSER_GetFreeSpaceLimitMB()**

```
uint32_t PARSER_GetFreeSpaceLimitMB (
    void )
```

Returns The Free Space Limit on SD Card For LED Signaling.

Returns

uint32_t Free Space Limit.

6.16.2.7 **PARSER_GetMaxBytes()**

```
uint32_t PARSER_GetMaxBytes (
    void )
```

Returns The Number of Maximal Bytes Between LED Blinking.

Returns

uint32_t Number of Maximal Bytes Between LED Blinking.

6.16.2.8 **PARSER_GetParity()**

```
lpuart_parity_mode_t PARSER_GetParity (
    void )
```

Returns The Parity.

Returns

lpuart_parity_mode_t kLPUART_ParityDisabled, kLPUART_ParityEven or kLPUART_ParityOdd.

6.16.2.9 PARSER_GetStopBits()

```
lpuart_stop_bit_count_t PARSER_GetStopBits (  
    void )
```

Returns The Number of Stop Bits.

Returns

lpuart_stop_bit_count_t Number of Stop Bits.

6.16.2.10 PARSER_GetVersion()

```
REC_version_t PARSER_GetVersion (  
    void )
```

Returns the Version of The Device Being Recorded.

Returns

REC_version_t Current Version of Recorded Device (WCT_UNKOWN, WCT_AUTOS1 or WCT_AUTOS2).

6.16.2.11 PARSER_ParseBaudrate()

```
error_t PARSER_ParseBaudrate (  
    const char * chContent)
```

Parse Baud Rate From Configuration File.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.12 PARSER_ParseDataBits()

```
error_t PARSER_ParseDataBits (  
    const char * chContent)
```

Parse The Number of Data Bits From Configuration File.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.13 PARSER_ParseFileSize()

```
error_t PARSER_ParseFileSize (  
    const char * chContent)
```

Parse Record File Size From Configuration File.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.14 PARSER_ParseFreeSpace()

```
error_t PARSER_ParseFreeSpace (  
    const char * chContent)
```

Parse The Size of Free Space When Data Logger Will Signal To The User That Data Logger Is Running Out of Space.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.15 PARSER_ParseParity()

```
error_t PARSER_ParseParity (  
    const char * chContent)
```

Parse Parity From Configuration File.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.16 PARSER_ParseStopBits()

```
error_t PARSER_ParseStopBits (  
    const char * chContent)
```

Parse The Number of Stop Bits From Configuration File.

Parameters

in	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.17 parser.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    parser.h
00004  * Author:       Tomas Dolak
00005  * Date:         16.04.2025
00006  * Description:  Header File Providing Definitions For Configuration File Parser.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         parser.h
00013  * @author       Tomas Dolak
00014  * @date         16.04.2025
00015  * @brief        Header File Providing Definitions For Configuration File Parser.
00016  * *****/
00017 #ifndef PARSE_H_
00018 #define PARSE_H_
00019
00020 /*****
00021  * Includes
00022  *****/
00023 #include <stdint.h>
00024 #include <string.h>
00025 #include <errno.h>
00026 #include <stdlib.h>
00027
00028 #include "fsl_lpuart.h"
00029 #include "fsl_debug_console.h"
00030 #include "error.h"
00031 #include "defs.h"
00032 /*****
00033  * Definitions
00034  *****/
00035
00036 /*****
00037  * Structures
00038  *****/
00039 typedef enum
00040 {
00041     WCT_UNKOWN = 0,
00042     WCT_AUTOS1,
00043     WCT_AUTOS2
00044 } REC_version_t;
00045
00046 typedef struct
00047 {
00048     /* Recorded NXP Device */
00049     REC_version_t version;
00050
00051     /* UART Setup */
00052     uint32_t baudrate;
00053     lpuart_stop_bit_count_t stop_bits;
00054     lpuart_data_bits_t data_bits;
00055     lpuart_parity_mode_t parity;
00056
00057     uint32_t size;
00058     uint32_t max_bytes;
00059     uint32_t free_space_limit_mb;
00060 } REC_config_t;
00061
00062 /*****
00063  * Prototypes
00064  *****/

```

```

00083  *****/
00093 REC_config_t PARSER_GetConfig(void);
00094
00101 REC_version_t PARSER_GetVersion(void);
00102
00109 uint32_t PARSER_GetBaudrate(void);
00110
00117 uint32_t PARSER_GetFileSize(void);
00118
00125 lpuart_data_bits_t PARSER_GetDataBits(void);
00126
00133 lpuart_parity_mode_t PARSER_GetParity(void);
00134
00141 lpuart_stop_bit_count_t PARSER_GetStopBits(void);
00142
00149 uint32_t PARSER_GetFreeSpaceLimitMB(void);
00150
00157 uint32_t PARSER_GetMaxBytes(void);
00158
00162 void PARSER_ClearConfig(void);
00163
00170 error_t PARSER_ParseBaudrate(const char *chContent);
00171
00178 error_t PARSER_ParseFileSize(const char *chContent);
00179
00186 error_t PARSER_ParseParity(const char *chContent);
00187
00194 error_t PARSER_ParseStopBits(const char *chContent);
00195
00202 error_t PARSER_ParseDataBits(const char *chContent);
00203
00211 error_t PARSER_ParseFreeSpace(const char *chContent);
00212
00213 #endif /* PARSER_H_ */

```

6.18 include/pwrloss_det.h File Reference

```

#include "fsl_lpcmp.h"
#include "fsl_spc.h"
#include "led.h"
#include "defs.h"
#include "fsl_ctimer.h"

```

Macros

- #define [LPCMP_BASE](#) CMP1
Base Address of The Low-Power Comparator (LPCMP).
- #define [LPCMP_USER_CHANNEL](#) 0x02
Comparator Input Channel Used For 5V Monitoring.
- #define [LPCMP_DAC_CHANNEL](#) 0x07
Comparator Internal DAC Reference Channel.
- #define [LPCMP_IRQ_ID](#) HSCMP1_IRQn
IRQ Number for The Comparator Peripheral.
- #define [SPC_BASE](#) SPC0
Base Address of The Supply Power Controller (SPC).
- #define [CTIMER](#) CTIMER4
Timer Used To Control Delayed Activation of Power Loss Detection.
- #define [CTIMER_MAT0_OUT](#) kCTIMER_Match_0
Timer Match Output Channel 0.
- #define [CTIMER_EMT0_OUT](#) (1u << kCTIMER_Match_0)
Timer External Match Output Mask For Channel 0.
- #define [CTIMER_CLK_FREQ](#) CLOCK_GetCTimerClkFreq(4U)
Clock Frequency Retrieval Macro For The Timer Instance.

Functions

- void [PWRLOSS_DetectionInit](#) (void)
Initializes Power Loss Detection Components.

6.18.1 Macro Definition Documentation

6.18.1.1 CTIMER

```
#define CTIMER CTIMER4
```

Timer Used To Control Delayed Activation of Power Loss Detection.

The Comparator is Enabled After a Delay of 5tau, Which Corresponds To The Time Needed For The Backup Capacitor To Become Fully Charged. This Delay Ensures Valid Detection of Power Drop Events and Avoids False Triggering During Startup.

6.18.1.2 CTIMER_CLK_FREQ

```
#define CTIMER_CLK_FREQ CLOCK_GetCTimerClkFreq(4U)
```

Clock Frequency Retrieval Macro For The Timer Instance.

6.18.1.3 CTIMER_EMT0_OUT

```
#define CTIMER_EMT0_OUT (1u << kCTIMER_Match_0)
```

Timer External Match Output Mask For Channel 0.

6.18.1.4 CTIMER_MAT0_OUT

```
#define CTIMER_MAT0_OUT kCTIMER_Match_0
```

Timer Match Output Channel 0.

6.18.1.5 LPCMP_BASE

```
#define LPCMP_BASE CMP1
```

Base Address of The Low-Power Comparator (LPCMP).

Used To Detect Loss of Primary 5V Supply. If The Voltage Drops Below a Defined Threshold, It Triggers a Transition To Backup Power Mode for Safe Data Preservation.

6.18.1.6 LPCMP_DAC_CHANNEL

```
#define LPCMP_DAC_CHANNEL 0x07
```

Comparator Internal DAC Reference Channel.

6.18.1.7 LPCMP_IRQ_ID

```
#define LPCMP_IRQ_ID HSCMP1_IRQn
```

IRQ Number for The Comparator Peripheral.

6.18.1.8 LPCMP_USER_CHANNEL

```
#define LPCMP_USER_CHANNEL 0x02
```

Comparator Input Channel Used For 5V Monitoring.

6.18.1.9 SPC_BASE

```
#define SPC_BASE SPC0
```

Base Address of The Supply Power Controller (SPC).

For Activation Analog Modules.

6.18.2 Function Documentation

6.18.2.1 PWRLOSS_DetectionInit()

```
void PWRLOSS_DetectionInit (  
    void )
```

Initializes Power Loss Detection Components.

This Function Configures The Comparator (LPCMP) for Monitoring The 5V Power Rail and The Timer (CTIMER4) That Delays The Activation of The Comparator Until The Backup Capacitor Is Fully Charged ($\sim 5\tau$). After Initialization, The Timer is Started and The Comparator Will Be Enabled Upon Timer Match Event.

6.19 pwrloss_det.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    pwrloss_det.h
00004  * Author:       Tomas Dolak
00005  * Date:         01.02.2024
00006  * Description:  Implements The Logic Of Power Loss Detection.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          pwrloss_det.h
00013  * @author        Tomas Dolak
00014  * @date          01.02.2024
00015  * @brief         Implements The Logic Of Power Loss Detection.
00016  * *****/
00017
00018 #ifndef PWRLOSS_DET_H_
00019 #define PWRLOSS_DET_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024
00025 #include "fsl_lpcmp.h"
00026 #include "fsl_spc.h"
00027
00028 #include "led.h"
00029 #include "defs.h"
00030
00031
00032 #include "fsl_ctimer.h"
00033 /*****
00034  * Global Definitions
00035  *****/
00036
00042 #define LPCMP_BASE          CMP1
00043
00047 #define LPCMP_USER_CHANNEL  0x02
00048
00052 #define LPCMP_DAC_CHANNEL   0x07
00053
00057 #define LPCMP_IRQ_ID        HSCMP1_IRQn
00058
00063 #define SPC_BASE            SPC0
00064
00071 #define CTIMER              CTIMER4
00072
00076 #define CTIMER_MAT0_OUT     kCTIMER_Match_0
00077
00081 #define CTIMER_EMT0_OUT     (1u << kCTIMER_Match_0)
00082
00086 #define CTIMER_CLK_FREQ     CLOCK_GetCTimerClkFreq(4U)
00087
00088 /*****
00089  * Prototypes
00090  *****/
00098 void PWRLOSS_DetectionInit(void);
00099
00100
00101 #endif /* PWRLOSS_DET_H_ */

```

6.20 include/record.h File Reference

```

#include <led.h>
#include <string.h>
#include "fsl_sd.h"
#include "ff.h"
#include "ffconf.h"
#include <stdio.h>
#include "fsl_sd_disk.h"
#include "fsl_common.h"

```

```
#include "diskio.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "sdmmc_config.h"
#include "rtc_ds3231.h"
#include "fsl_common_arm.h"
#include "task.h"
#include "error.h"
#include "uart.h"
#include "parser.h"
```

Macros

- #define [FLUSH_TIMEOUT_TICKS](#) pdMS_TO_TICKS(3000)
Timeout Interval Before Flush If No New Data Were Received [In Mili-Seconds].

Functions

- uint32_t [CONSOLELOG_GetFreeSpaceMB](#) (void)
Gets Free Space on SD Card.
- error_t [CONSOLELOG_CreateDirectory](#) (void)
Creates Directory Based Actual Date.
- error_t [CONSOLELOG_CreateFile](#) (void)
Creates File Based Actual Date and Counter Value.
- uint32_t [CONSOLELOG_GetTransferredBytes](#) (void)
Returns Currently Received Bytes Between LED Blinking.
- bool [CONSOLELOG_GetFlushCompleted](#) (void)
Returns If Flush Was Completed or Not.
- void [CONSOLELOG_ClearTransferredBytes](#) (void)
Clears Currently Received Bytes After LED Blinking.
- uint32_t [CONSOLELOG_GetMaxBytes](#) (void)
Maximal Received Bytes Between LED Blinking.
- FRESULT [CONSOLELOG_CheckFileSystem](#) (void)
Checks If The File System Is Initialized.
- error_t [CONSOLELOG_Init](#) (void)
Initializes The Recording System and Mounts The File System.
- error_t [CONSOLELOG_Recording](#) (uint32_t file_size)
Starts The Recording Process by Initializing the File System, Creating a Directory, and Writing to a File.
- error_t [CONSOLELOG_Flush](#) (void)
Flushes Collected Data To The File If No Other Data Have Been Received By The Time Specified By TIMEOUT Macro.
- error_t [CONSOLELOG_PowerLossFlush](#) (void)
Flushes Collected Data To The File If Power Loss Was Detected.
- error_t [CONSOLELOG_Deinit](#) (void)
De-Initializes The Recording System and Un-Mounts The File System.
- error_t [CONSOLELOG_ReadConfig](#) (void)
Reads and Processes The Configuration File From The Root directory.
- error_t [CONSOLELOG_ProccessConfigFile](#) (const char *content)
Processes The Content of The Configuration File To Extract and Validate The Baudrate.

6.20.1 Macro Definition Documentation

6.20.1.1 FLUSH_TIMEOUT_TICKS

```
#define FLUSH_TIMEOUT_TICKS pdMS_TO_TICKS(3000)
```

Timeout Interval Before Flush If No New Data Were Received [In Mili-Seconds].

6.20.2 Function Documentation

6.20.2.1 CONSOLELOG_CheckFileSystem()

```
FRESULT CONSOLELOG_CheckFileSystem (  
    void )
```

Checks If The File System Is Initialized.

return F_OK If File System Initialized.

6.20.2.2 CONSOLELOG_ClearTransferredBytes()

```
void CONSOLELOG_ClearTransferredBytes (  
    void )
```

Clears Currently Received Bytes After LED Blinking.

6.20.2.3 CONSOLELOG_CreateDirectory()

```
error_t CONSOLELOG_CreateDirectory (  
    void )
```

Creates Directory Based Actual Date.

Returns

Returns Zero If Directory Creation Succeeded.

6.20.2.4 CONSOLELOG_CreateFile()

```
error_t CONSOLELOG_CreateFile (  
    void )
```

Creates File Based Actual Date and Counter Value.

Returns

Returns Zero If File Creation Succeeded.

6.20.2.5 CONSOLELOG_Deinit()

```
error_t CONSOLELOG_Deinit (  
    void )
```

De-Initializes The Recording System and Un-Mounts The File System.

Returns

`error_t` Returns 0 on Success, Otherwise E_FAULT.

6.20.2.6 CONSOLELOG_Flush()

```
error_t CONSOLELOG_Flush (  
    void )
```

Flushes Collected Data To The File If No Other Data Have Been Received By The Time Specified By TIMEOUT Macro.

If The Data Does Not Arrive By The Time Specified By The TIMEOUT Macro, Then This Function Flushes All The Data So Far Stored In The DMA Buffer, Saves It To a File on The Physical Media and Closes The File.

Returns

`error_t` Returns 0 on Success, Otherwise Returns a Non-Zero Value.

6.20.2.7 CONSOLELOG_GetFlushCompleted()

```
bool CONSOLELOG_GetFlushCompleted (  
    void )
```

Returns If Flush Was Completed or Not.

Returns

If Recording Is Ongoing That Return False.

6.20.2.8 CONSOLELOG_GetFreeSpaceMB()

```
uint32_t CONSOLELOG_GetFreeSpaceMB (  
    void )
```

Gets Free Space on SD Card.

Returns

Returns Free Space on SD Card.

6.20.2.9 CONSOLELOG_GetMaxBytes()

```
uint32_t CONSOLELOG_GetMaxBytes (
    void )
```

Maximal Received Bytes Between LED Blinking.

6.20.2.10 CONSOLELOG_GetTransferredBytes()

```
uint32_t CONSOLELOG_GetTransferredBytes (
    void )
```

Returns Currently Received Bytes Between LED Blinking.

Returns

uint32_t Received Bytes Between LED Blinking.

6.20.2.11 CONSOLELOG_Init()

```
error_t CONSOLELOG_Init (
    void )
```

Initializes The Recording System and Mounts The File System.

This Function Performs the Initialization of The Recording System, Which includes:

- Setting Default Configuration Parameters.
- Mounting the File System on The Logical Disk.
- Optionally Setting Up The Current Working Drive.
- Formatting The File System If It is Not Found (If Formatting is Enabled).

Returns

error_t Returns ERROR_NONE on Success, Otherwise ERROR_FILESYSTEM.

6.20.2.12 CONSOLELOG_PowerLossFlush()

```
error_t CONSOLELOG_PowerLossFlush (
    void )
```

Flushes Collected Data To The File If Power Loss Was Detected.

If Power Loss Was Detected, Then This Function Flushes All The Data So Far Stored In The DMA Buffer, Saves It To a File on The Physical Media and Closes The File.

Returns

error_t Returns 0 on Success, Otherwise Returns a Non-Zero Value.

6.20.2.13 CONSOLELOG_ProccessConfigFile()

```
error_t CONSOLELOG_ProccessConfigFile (
    const char * content)
```

Processes The Content of The Configuration File To Extract and Validate The Baudrate.

Parameters

<code>in</code>	<code>content</code>	Content The Content of The Configuration File as a Null-Terminated String.
-----------------	----------------------	--

Returns

`error_t` Returns 0 If Configuration File Is Correctly Processed, Otherwise Returns E_FAULT.

6.20.2.14 CONSOLELOG_ReadConfig()

```
error_t CONSOLELOG_ReadConfig (
    void )
```

Reads and Processes The Configuration File From The Root directory.

This Function Scans The Root Directory For a Configuration File, If The File is Found Reads its Contents Into g_config Buffer.

Returns

`error_t` Returns 0 If Configuration File Is Correctly Processed, Otherwise Returns E_FAULT.

6.20.2.15 CONSOLELOG_Recording()

```
error_t CONSOLELOG_Recording (
    uint32_t file_size)
```

Starts The Recording Process by Initializing the File System, Creating a Directory, and Writing to a File.

Function Uses CONSOLELOG_Init To Initialize The Recording System.

Returns

`error_t` Returns 0 on Success, Otherwise Returns a Non-Zero Value.

6.21 record.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    fatfs.h
00004  * Author:      Tomas Dolak
00005  * Date:        18.11.2024
00006  * Description:  File Includes Operation For Recoding Mode.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          fatfs.h
00013  * @author        Tomas Dolak
00014  * @date          18.11.2024
00015  * @brief        File Includes Operation For Recoding Mode.
00016  * *****/
00017
```

```

00018 #ifndef RECORD_H_
00019 #define RECORD_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include <led.h>
00025 #include <string.h>
00026 #include "fsl_sd.h"
00027 #include "ff.h" /*<! File System */
00028 #include "ffconf.h" /*<! File System Configuration */
00029 #include <stdio.h>
00030
00031 #include "fsl_sd_disk.h"
00032 #include "fsl_common.h"
00033 #include "diskio.h"
00034 #include "pin_mux.h"
00035 #include "clock_config.h"
00036 #include "board.h"
00037 #include "sdmmc_config.h"
00038 #include "rtc_ds3231.h"
00039 #include "fsl_common_arm.h"
00040
00041 #include "task.h"
00042
00043 #include "error.h"
00044 #include "uart.h"
00045 #include "parser.h"
00046 /*****
00047  * Definitions
00048  *****/
00052 #define FLUSH_TIMEOUT_TICKS pdMS_TO_TICKS(3000)
00053
00054 /*****
00055  * Structures
00056  *****/
00057
00058 /*****
00059  * Prototypes
00060  *****/
00061
00067 uint32_t  CONSOLELOG_GetFreeSpaceMB(void);
00068
00074 error_t  CONSOLELOG_CreateDirectory(void);
00075
00081 error_t  CONSOLELOG_CreateFile(void);
00082
00088 uint32_t  CONSOLELOG_GetTransferredBytes(void);
00089
00095 bool  CONSOLELOG_GetFlushCompleted(void);
00096
00100 void  CONSOLELOG_ClearTransferredBytes(void);
00101
00105 uint32_t  CONSOLELOG_GetMaxBytes(void);
00106
00112 FRESULT  CONSOLELOG_CheckFileSystem(void);
00113
00127 error_t  CONSOLELOG_Init(void);
00128
00138 error_t  CONSOLELOG_Recording(uint32_t file_size);
00139
00149 error_t  CONSOLELOG_Flush(void);
00150
00158 error_t  CONSOLELOG_PowerLossFlush(void);
00159
00165 error_t  CONSOLELOG_Deinit(void);
00166
00176 error_t  CONSOLELOG_ReadConfig(void);
00177
00188 error_t  CONSOLELOG_ProccessConfigFile(const char *content);
00189
00190
00191 #endif /* RECORD_H_ */

```

6.22 include/task_switching.h File Reference

```
#include "usb_device_dci.h"
```

Functions

- `usb_device_notification_t` [USB_State](#) (`usb_device_struct_t *pDeviceHandle`)
Checks If Application USB Is Attached To Digital Data Logger.

Variables

- `usb_msc_struct_t` [g_msc](#)
Mass Storage Descriptor.

6.22.1 Function Documentation

6.22.1.1 USB_State()

```
usb_device_notification_t USB_State (
    usb_device_struct_t * pDeviceHandle)
```

Checks If Application USB Is Attached To Digital Data Logger.

Checks Thru USB OTG (USB On-To-Go) SC Register.

Parameters

in	<i>pDeviceHandle</i>	Pointer to USB Device Handle (e.g. Mass Storage Handle).
----	----------------------	--

Returns

kUSB_DeviceNotifyAttach if USB Is Attached Otherwise Returns kUSB_DeviceNotifyDetach.

6.22.2 Variable Documentation

6.22.2.1 g_msc

```
usb_msc_struct_t g_msc [extern]
```

Mass Storage Descriptor.

6.23 task_switching.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    task_switching.c
00004  * Author:       Tomas Dolak
00005  * Date:         14.09.2024
00006  * Description:   Includes Implementation of Task Switching (Record / USB Mass Storage).
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          task_switching.c
00013  * @author        Tomas Dolak
00014  * @date          14.09.2024
00015  * @brief         Includes Implementation of Task Switching (Record / USB Mass Storage).
00016  * *****/
00017
00018 #ifndef TASK_SWITCHING_H_
00019 #define TASK_SWITCHING_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include "usb_device_dci.h"
00025
00026 /*****
00027  * Global Variables
00028  *****/
00029
00030 extern usb_msc_struct_t g_msc;
00031 /*****
00032  * Prototypes
00033  *****/
00034 usb_device_notification_t USB_State(usb_device_struct_t *pDeviceHandle);
00035
00036 #endif /* TASK_SWITCHING_H_ */

```

6.24 include/temperature.h File Reference

```

#include "error.h"
#include "fsl_lpi2c.h"
#include "fsl_lpi2c_edma.h"
#include "fsl_edma.h"

```

Functions

- uint8_t [Write](#) (uint8_t regAddress, uint8_t val[])
I2C Write Function.
- uint16_t [Read](#) (uint8_t regAddress)
I2C Read Function.
- float [TMP_GetTemperature](#) (void)
Gets Temperature From On-Board P3T1755 Temperature Sensor.
- uint8_t [TMP_Init](#) (void)
Initialize On-Board P3T1755 Temperature Sensor.

6.24.1 Function Documentation

6.24.1.1 Read()

```

uint16_t Read (
    uint8_t regAddress)

```

I2C Read Function.

Parameters

<i>in</i>	<i>regAddress</i>	Address of The Register From Which The Reading Will Take Place.
-----------	-------------------	---

Return values

<i>Returns</i>	The Read Value From The Registry.
----------------	-----------------------------------

6.24.1.2 TMP_GetTemperature()

```
float TMP_GetTemperature (
    void )
```

Gets Temperature From On-Board P3T1755 Temperature Sensor.

Return values

<i>Returns</i>	Temperature As Float Number.
----------------	------------------------------

6.24.1.3 TMP_Init()

```
uint8_t TMP_Init (
    void )
```

Initialize On-Board P3T1755 Temperature Sensor.

6.24.1.4 Write()

```
uint8_t Write (
    uint8_t regAddress,
    uint8_t val [])
```

I2C Write Function.

Parameters

<i>in</i>	<i>regAddress</i>	Address of The Register To Be Written To.
<i>in</i>	<i>val</i>	Array of Values To Be Written To The Register.

Return values

<i>If</i>	The Write Succeeds, Returns 0.
-----------	--------------------------------

6.25 temperature.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    temperature.h
00004  * Author:       Tomas Dolak
00005  * Date:         11.02.2025
00006  * Description:  Implements The Logic For Temperature Measurement.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         temperature.h
00013  * @author       Tomas Dolak
00014  * @date         11.02.2025
00015  * @brief        Implements The Logic For Temperature Measurement.
00016  * *****/
00017
00018 #ifndef TEMPERATURE_H_
00019 #define TEMPERATURE_H_
00020
00021 /*****
00022  * Includes
00023  * *****/
00024
00025 #include "error.h"
00026
00027 #include "fsl_lpi2c.h"
00028 #include "fsl_lpi2c_edma.h"
00029 #include "fsl_edma.h"
00030 /*****
00031  * Global Definitions
00032  * *****/
00033
00034
00035 /*****
00036  * Prototypes
00037  * *****/
00046 uint8_t Write(uint8_t regAddress, uint8_t val[]);
00047
00055 uint16_t Read(uint8_t regAddress);
00056
00062 float TMP_GetTemperature(void);
00063
00067 uint8_t TMP_Init(void);
00068
00069 #endif /* TEMPERATURE_H_ */

```

6.26 include/time.h File Reference

```

#include "fsl_irtc.h"
#include "rtc_ds3231.h"
#include "fsl_debug_console.h"
#include "error.h"

```

Macros

- #define [LPI2C_DMA_BASEADDR](#) (DMA0)
I2C Definitions.

Functions

- [error_t TIME_InitIRTC](#) (void)
Initialize Internal And External Real-Time Circuits And Passes Timestamp Information From The External To The Internal.

6.26.1 Macro Definition Documentation

6.26.1.1 LPI2C_DMA_BASEADDR

```
#define LPI2C_DMA_BASEADDR (DMA0)
```

I2C Definitions.

6.26.2 Function Documentation

6.26.2.1 TIME_InitIRTC()

```
error_t TIME_InitIRTC (
    void )
```

Initialize Internal And External Real-Time Circuits And Passes Timestamp Information From The External To The Internal.

6.27 time.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    main.c
00004  * Author:       Tomas Dolak
00005  * Date:         07.08.2024
00006  * Description:  Implements Datalogger Application.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          main.c
00013  * @author        Tomas Dolak
00014  * @date          07.08.2024
00015  * @brief         Implements Datalogger Application.
00016  * *****/
00017
00018
00019 #ifndef TIME_H_
00020 #define TIME_H_
00021
00022 /*****
00023  * Includes
00024  * *****/
00025 #include "fsl_irtc.h"
00026 #include "rtc_ds3231.h"
00027 #include "fsl_debug_console.h"
00028 #include "error.h"
00029
00030 /*****
00031  * Global Definitions
00032  * *****/
00033
00037 #define LPI2C_DMA_BASEADDR (DMA0)
00038
00039 /*****
00040  * Prototypes
00041  * *****/
00042
00047 error_t TIME_InitIRTC(void);
00048
00049 #endif /* TIME_H_ */
```

6.28 include/uart.h File Reference

```
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_lpuart.h"
#include "fsl_debug_console.h"
#include "fsl_clock.h"
```

Macros

- #define [LPUART3_CLK_FREQ](#) `CLOCK_GetLPFlexCommClkFreq(3u)`
Frequency of LPUART7.

Functions

- void [UART_Init](#) (uint32_t baudrate)
Initializes LPUART For Recording.
- void [UART_Print](#) (uint8_t ch)
Prints Character on The Terminal.
- void [UART_Enable](#) (void)
Enables Interrupt For Application LPUART.
- void [UART_Disable](#) (void)
Disables Interrupt For Application LPUART.
- void [UART_Deinit](#) (void)
De-Initialize LPUART.

6.28.1 Macro Definition Documentation

6.28.1.1 LPUART3_CLK_FREQ

```
#define LPUART3_CLK_FREQ CLOCK_GetLPFlexCommClkFreq(3u)
```

Frequency of LPUART7.

6.28.2 Function Documentation

6.28.2.1 UART_Deinit()

```
void UART_Deinit (  
    void )
```

De-Initialize LPUART.

The Pins Should Be De-Initialized After This Function.

6.28.2.2 UART_Disable()

```
void UART_Disable (
    void )
```

Disables Interrupt For Application LPUART.

6.28.2.3 UART_Enable()

```
void UART_Enable (
    void )
```

Enables Interrupt For Application LPUART.

6.28.2.4 UART_Init()

```
void UART_Init (
    uint32_t baudrate)
```

Initializes LPUART For Recording.

6.28.2.5 UART_Print()

```
void UART_Print (
    uint8_t ch)
```

Prints Character on The Terminal.

Parameters

in	ch	Character in uint8_t.
----	----	-----------------------

6.29 uart.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    uart.h
00004  * Author:      Tomas Dolak
00005  * Date:        25.11.2024
00006  * Description:  Header File For The UART Peripheral Support.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         uart.h
00013  * @author       Tomas Dolak
00014  * @date        25.11.2024
00015  * @brief       Header File For The UART Peripheral Support.
00016  * *****/
00017
00018 #ifndef UART_H_
00019 #define UART_H_
```

```
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include "pin_mux.h"
00025 #include "clock_config.h"
00026 #include "board.h"
00027 #include "fsl_lpuart.h"
00028
00029 #include "fsl_debug_console.h"
00030 #include "fsl_clock.h"
00031 /*****
00032  * Definitions
00033  *****/
00037 #define LPUART3_CLK_FREQ    CLOCK_GetLPFlexCommClkFreq(3u)
00038
00039
00040 /*****
00041  * Declarations
00042  *****/
00043
00047 void UART_Init(uint32_t baudrate);
00048
00054 void UART_Print(uint8_t ch);
00055
00059 void UART_Enable(void);
00060
00064 void UART_Disable(void);
00065
00070 void UART_Deinit(void);
00071
00072 #endif /* UART_H_ */
00073
```


Index

ALARM_DISABLED
 rtc_ds3231.h, [23](#)

ALARM_INTERRUPT
 rtc_ds3231.h, [28](#)

app_init.h
 APP_InitBoard, [30](#)

APP_InitBoard
 app_init.h, [30](#)

APP_SUCCESS
 rtc_ds3231.h, [23](#)

app_tasks.h
 g_xMscTaskHandle, [33](#)
 g_xRecordTaskHandle, [33](#)
 g_xSemMassStorage, [34](#)
 g_xSemRecord, [34](#)
 msc_task, [32](#)
 record_task, [32](#)
 TASK_PRIO, [32](#)
 vApplicationGetIdleTaskMemory, [33](#)
 vApplicationGetTimerTaskMemory, [33](#)

AT_NONCACHEABLE_SECTION
 rtc_ds3231.c, [19](#)

BACKUP_POWER_LED_PIN
 led.h, [46](#)

BACKUP_POWER_LED_PORT
 led.h, [46](#)

baudrate
 REC_config_t, [13](#)

BYTE_1
 rtc_ds3231.c, [18](#)

BYTE_2
 rtc_ds3231.c, [18](#)

CLK_STATE
 rtc_ds3231.c, [18](#)

CONFIG_FILE
 defs.h, [36](#)

CONSOLELOG_CheckFileSystem
 record.h, [63](#)

CONSOLELOG_ClearTransferredBytes
 record.h, [63](#)

CONSOLELOG_CreateDirectory
 record.h, [63](#)

CONSOLELOG_CreateFile
 record.h, [63](#)

CONSOLELOG_Deinit
 record.h, [63](#)

CONSOLELOG_Flush
 record.h, [64](#)

CONSOLELOG_GetFlushCompleted
 record.h, [64](#)

CONSOLELOG_GetFreeSpaceMB
 record.h, [64](#)

CONSOLELOG_GetMaxBytes
 record.h, [64](#)

CONSOLELOG_GetTransferredBytes
 record.h, [65](#)

CONSOLELOG_Init
 record.h, [65](#)

CONSOLELOG_PowerLossFlush
 record.h, [65](#)

CONSOLELOG_ProccessConfigFile
 record.h, [65](#)

CONSOLELOG_ReadConfig
 record.h, [66](#)

CONSOLELOG_Recording
 record.h, [66](#)

CONTROL_LED_ENABLED
 defs.h, [36](#)

CTIMER
 pwrloss_det.h, [59](#)

CTIMER_CLK_FREQ
 pwrloss_det.h, [59](#)

CTIMER_EMT0_OUT
 pwrloss_det.h, [59](#)

CTIMER_MAT0_OUT
 pwrloss_det.h, [59](#)

data_bits
 REC_config_t, [13](#)

date
 RTC_date_t, [15](#)

day
 RTC_date_t, [15](#)

DEBUG_ENABLED
 defs.h, [36](#)

DEFAULT_BAUDRATE
 defs.h, [36](#)

DEFAULT_DATA_BITS
 defs.h, [36](#)

DEFAULT_FREE_SPACE
 defs.h, [37](#)

DEFAULT_MAX_FILESIZE
 defs.h, [37](#)

DEFAULT_PARITY
 defs.h, [37](#)

DEFAULT_STOP_BITS
 defs.h, [37](#)

defs.h

- CONFIG_FILE, [36](#)
- CONTROL_LED_ENABLED, [36](#)
- DEBUG_ENABLED, [36](#)
- DEFAULT_BAUDRATE, [36](#)
- DEFAULT_DATA_BITS, [36](#)
- DEFAULT_FREE_SPACE, [37](#)
- DEFAULT_MAX_FILESIZE, [37](#)
- DEFAULT_PARITY, [37](#)
- DEFAULT_STOP_BITS, [37](#)
- INFO_ENABLED, [37](#)
- MSC_ENABLED, [37](#)
- MSC_STACK_SIZE, [37](#)
- NOT_IMPLEMENTED, [38](#)
- PWRLOSS_DET_ACTIVE_IN_TIME, [38](#)
- PWRLOSS_DET_PRIO, [38](#)
- PWRLOSS_DETECTION_ENABLED, [38](#)
- PWRLOSS_TEST_GPIOS, [38](#)
- PWRLOSS_TIMER_PRIO, [38](#)
- RECORD_STACK_SIZE, [38](#)
- TAU5, [38](#)
- TEMPERATURE_MEAS_ENABLED, [39](#)
- UART_FIFO_ENABLED, [39](#)
- UART_FIFO_LENHT, [39](#)
- UART_PRINT_ENABLED, [39](#)
- UART_RECEIVE_PRIO, [39](#)
- drivers/rtc_ds3231.c, [17](#)
- drivers/rtc_ds3231.h, [20, 28](#)
- DS3231 Real-Time Clock Driver, [7](#)
 - RTC_ClearFlagAlarm1, [8](#)
 - RTC_ClearFlagAlarm2, [8](#)
 - RTC_ConvertToBCD, [8](#)
 - RTC_ConvertToDEC, [8](#)
 - RTC_CtrlAlarm1, [9](#)
 - RTC_CtrlAlarm2, [9](#)
 - RTC_Deinit, [9](#)
 - RTC_GetDate, [9](#)
 - RTC_GetState, [10](#)
 - RTC_GetTime, [10](#)
 - RTC_Init, [10](#)
 - RTC_Read, [10](#)
 - RTC_SetDate, [11](#)
 - RTC_SetDateDefault, [11](#)
 - RTC_SetInterruptMode, [11](#)
 - RTC_SetOscState, [11](#)
 - RTC_SetTime, [12](#)
 - RTC_SetTimeDefault, [12](#)
 - RTC_Write, [12](#)
- DS3231_A1F
 - rtc_ds3231.h, [23](#)
- DS3231_A1IE
 - rtc_ds3231.h, [23](#)
- DS3231_A2F
 - rtc_ds3231.h, [23](#)
- DS3231_A2IE
 - rtc_ds3231.h, [23](#)
- DS3231_ADDR_CENT
 - rtc_ds3231.h, [23](#)
- DS3231_ADDR_DATE
 - rtc_ds3231.h, [23](#)
- DS3231_ADDR_DAY
 - rtc_ds3231.h, [23](#)
- DS3231_ADDR_HRS
 - rtc_ds3231.h, [24](#)
- DS3231_ADDR_I2C
 - rtc_ds3231.h, [24](#)
- DS3231_ADDR_MIN
 - rtc_ds3231.h, [24](#)
- DS3231_ADDR_MONTH
 - rtc_ds3231.h, [24](#)
- DS3231_ADDR_SEC
 - rtc_ds3231.h, [24](#)
- DS3231_ADDR_YEAR
 - rtc_ds3231.h, [24](#)
- DS3231_INTCN
 - rtc_ds3231.h, [24](#)
- DS3231_REG_CTRL
 - rtc_ds3231.h, [25](#)
- DS3231_REG_STATUS
 - rtc_ds3231.h, [25](#)
- E_FAULT
 - rtc_ds3231.h, [25](#)
- ERR_HandleError
 - error.h, [43](#)
- ERR_Init
 - error.h, [43](#)
- ERR_SetState
 - error.h, [43](#)
- error.h
 - ERR_HandleError, [43](#)
 - ERR_Init, [43](#)
 - ERR_SetState, [43](#)
 - ERROR_ADMA, [41](#)
 - ERROR_CLOSE, [41](#)
 - ERROR_CONFIG, [42](#)
 - ERROR_FILESYSTEM, [42](#)
 - ERROR_IRTC, [42](#)
 - ERROR_NONE, [42](#)
 - ERROR_OPEN, [42](#)
 - ERROR_OUT_OF_CYCLE, [42](#)
 - ERROR_READ, [42](#)
 - ERROR_RECORD, [42](#)
 - error_t, [43](#)
 - ERROR_UNKNOWN, [43](#)
- ERROR_ADMA
 - error.h, [41](#)
- ERROR_CLOSE
 - error.h, [41](#)
- ERROR_CONFIG
 - error.h, [42](#)
- ERROR_FILESYSTEM
 - error.h, [42](#)
- ERROR_IRTC
 - error.h, [42](#)
- ERROR_LED_PIN_RECORD
 - led.h, [46](#)
- ERROR_LED_PORT

- led.h, 46
- ERROR_NONE
 - error.h, 42
- ERROR_OPEN
 - error.h, 42
- ERROR_OUT_OF_CYCLE
 - error.h, 42
- ERROR_READ
 - error.h, 42
- ERROR_RECORD
 - error.h, 42
- error_t
 - error.h, 43
- ERROR_UNKNOWN
 - error.h, 43
- FLUSH_LED_PORT
 - led.h, 46
- FLUSH_TIMEOUT_TICKS
 - record.h, 63
- format
 - RTC_time_t, 16
- free_space_limit_mb
 - REC_config_t, 14
- FRIDAY
 - rtc_ds3231.h, 25
- g_msc
 - task_switching.h, 68
- g_xMscTaskHandle
 - app_tasks.h, 33
- g_xRecordTaskHandle
 - app_tasks.h, 33
- g_xSemMassStorage
 - app_tasks.h, 34
- g_xSemRecord
 - app_tasks.h, 34
- gCompletionFlag
 - rtc_ds3231.c, 20
- gEdmaRxHandle
 - rtc_ds3231.c, 20
- gEdmaTxHandle
 - rtc_ds3231.c, 20
- hrs
 - RTC_time_t, 16
- I2C_BAUDRATE
 - rtc_ds3231.h, 25
- I2C_DATA_LENGTH
 - rtc_ds3231.c, 19
- I2C_MASTER
 - rtc_ds3231.h, 25
- include/app_init.h, 30, 31
- include/app_tasks.h, 31, 34
- include/defs.h, 35, 40
- include/error.h, 41, 44
- include/led.h, 44, 49
- include/mass_storage.h, 50, 51

- include/parser.h, 51, 57
- include/pwrloss_det.h, 58, 61
- include/record.h, 61, 66
- include/task_switching.h, 67, 69
- include/temperature.h, 69, 71
- include/time.h, 71, 72
- include/uart.h, 73, 74
- INFO_ENABLED
 - defs.h, 37
- led.h
 - BACKUP_POWER_LED_PIN, 46
 - BACKUP_POWER_LED_PORT, 46
 - ERROR_LED_PIN_RECORD, 46
 - ERROR_LED_PORT, 46
 - FLUSH_LED_PORT, 46
 - LED_ClearSignalFlush, 47
 - LED_SetHigh, 47
 - LED_SetLow, 48
 - LED_SignalBackUpPowerAvailable, 48
 - LED_SignalError, 48
 - LED_SignalFlush, 48
 - LED_SignalLowMemory, 48
 - LED_SignalReady, 48
 - LED_SignalRecording, 49
 - LED_SignalRecordingStop, 49
 - MEMORY_LOW_LED_PIN, 46
 - MEMORY_LOW_LED_PORT, 46
 - RECORD_LED_PIN, 47
 - RECORD_LED_PIN_FLUSH, 47
 - RECORD_LED_PORT, 47
- LED_ClearSignalFlush
 - led.h, 47
- LED_SetHigh
 - led.h, 47
- LED_SetLow
 - led.h, 48
- LED_SignalBackUpPowerAvailable
 - led.h, 48
- LED_SignalError
 - led.h, 48
- LED_SignalFlush
 - led.h, 48
- LED_SignalLowMemory
 - led.h, 48
- LED_SignalReady
 - led.h, 48
- LED_SignalRecording
 - led.h, 49
- LED_SignalRecordingStop
 - led.h, 49
- LPCMP_BASE
 - pwrloss_det.h, 59
- LPCMP_DAC_CHANNEL
 - pwrloss_det.h, 59
- LPCMP_IRQ_ID
 - pwrloss_det.h, 60
- LPCMP_USER_CHANNEL
 - pwrloss_det.h, 60

- lpi2c_callback
 - rtc_ds3231.c, [19](#)
- LPI2C_DMA_BASEADDR
 - time.h, [72](#)
- LPI2C_RX_DMA_CHANNEL
 - rtc_ds3231.h, [25](#)
- LPI2C_RX_EDMA_CHANNEL
 - rtc_ds3231.h, [26](#)
- LPI2C_TX_CHANNEL
 - rtc_ds3231.h, [26](#)
- LPI2C_TX_DMA_CHANNEL
 - rtc_ds3231.h, [26](#)
- LPUART3_CLK_FREQ
 - uart.h, [73](#)
- mass_storage.h
 - MSC_DeviceMscApp, [50](#)
 - MSC_DeviceMscAppTask, [50](#)
- max_bytes
 - REC_config_t, [14](#)
- MEMORY_LOW_LED_PIN
 - led.h, [46](#)
- MEMORY_LOW_LED_PORT
 - led.h, [46](#)
- min
 - RTC_time_t, [16](#)
- MONDAY
 - rtc_ds3231.h, [26](#)
- month
 - RTC_date_t, [15](#)
- MSC_DeviceMscApp
 - mass_storage.h, [50](#)
- MSC_DeviceMscAppTask
 - mass_storage.h, [50](#)
- MSC_ENABLED
 - defs.h, [37](#)
- MSC_STACK_SIZE
 - defs.h, [37](#)
- msc_task
 - app_tasks.h, [32](#)
- NOT_IMPLEMENTED
 - defs.h, [38](#)
- OSC_INTERRUPTED
 - rtc_ds3231.h, [28](#)
- OSC_OK
 - rtc_ds3231.h, [28](#)
- OSC_STOPPED
 - rtc_ds3231.h, [26](#)
- parity
 - REC_config_t, [14](#)
- parser.h
 - PARSER_ClearConfig, [53](#)
 - PARSER_GetBaudrate, [53](#)
 - PARSER_GetConfig, [53](#)
 - PARSER_GetDataBits, [53](#)
 - PARSER_GetFileSize, [53](#)
 - PARSER_GetFreeSpaceLimitMB, [54](#)
 - PARSER_GetMaxBytes, [54](#)
 - PARSER_GetParity, [54](#)
 - PARSER_GetStopBits, [54](#)
 - PARSER_GetVersion, [55](#)
 - PARSER_ParseBaudrate, [55](#)
 - PARSER_ParseDataBits, [55](#)
 - PARSER_ParseFileSize, [55](#)
 - PARSER_ParseFreeSpace, [56](#)
 - PARSER_ParseParity, [56](#)
 - PARSER_ParseStopBits, [56](#)
 - REC_version_t, [52](#)
 - WCT_AUTOS1, [53](#)
 - WCT_AUTOS2, [53](#)
 - WCT_UNKOWN, [53](#)
- PARSER_ClearConfig
 - parser.h, [53](#)
- PARSER_GetBaudrate
 - parser.h, [53](#)
- PARSER_GetConfig
 - parser.h, [53](#)
- PARSER_GetDataBits
 - parser.h, [53](#)
- PARSER_GetFileSize
 - parser.h, [53](#)
- PARSER_GetFreeSpaceLimitMB
 - parser.h, [54](#)
- PARSER_GetMaxBytes
 - parser.h, [54](#)
- PARSER_GetParity
 - parser.h, [54](#)
- PARSER_GetStopBits
 - parser.h, [54](#)
- PARSER_GetVersion
 - parser.h, [55](#)
- PARSER_ParseBaudrate
 - parser.h, [55](#)
- PARSER_ParseDataBits
 - parser.h, [55](#)
- PARSER_ParseFileSize
 - parser.h, [55](#)
- PARSER_ParseFreeSpace
 - parser.h, [56](#)
- PARSER_ParseParity
 - parser.h, [56](#)
- PARSER_ParseStopBits
 - parser.h, [56](#)
- pwrloss_det.h
 - CTIMER, [59](#)
 - CTIMER_CLK_FREQ, [59](#)
 - CTIMER_EMT0_OUT, [59](#)
 - CTIMER_MAT0_OUT, [59](#)
 - LPCMP_BASE, [59](#)
 - LPCMP_DAC_CHANNEL, [59](#)
 - LPCMP_IRQ_ID, [60](#)
 - LPCMP_USER_CHANNEL, [60](#)
 - PWRLOSS_DetectionInit, [60](#)
 - SPC_BASE, [60](#)

PWRLOSS_DET_ACTIVE_IN_TIME
 defs.h, [38](#)
 PWRLOSS_DET_PRIO
 defs.h, [38](#)
 PWRLOSS_DETECTION_ENABLED
 defs.h, [38](#)
 PWRLOSS_DetectionInit
 pwrloss_det.h, [60](#)
 PWRLOSS_TEST_GPIO
 defs.h, [38](#)
 PWRLOSS_TIMER_PRIO
 defs.h, [38](#)

 Read
 temperature.h, [69](#)
 REC_config_t, [13](#)
 baudrate, [13](#)
 data_bits, [13](#)
 free_space_limit_mb, [14](#)
 max_bytes, [14](#)
 parity, [14](#)
 size, [14](#)
 stop_bits, [14](#)
 version, [14](#)
 REC_version_t
 parser.h, [52](#)
 record.h
 CONSOLELOG_CheckFileSystem, [63](#)
 CONSOLELOG_ClearTransferredBytes, [63](#)
 CONSOLELOG_CreateDirectory, [63](#)
 CONSOLELOG_CreateFile, [63](#)
 CONSOLELOG_Deinit, [63](#)
 CONSOLELOG_Flush, [64](#)
 CONSOLELOG_GetFlushCompleted, [64](#)
 CONSOLELOG_GetFreeSpaceMB, [64](#)
 CONSOLELOG_GetMaxBytes, [64](#)
 CONSOLELOG_GetTransferredBytes, [65](#)
 CONSOLELOG_Init, [65](#)
 CONSOLELOG_PowerLossFlush, [65](#)
 CONSOLELOG_ProccessConfigFile, [65](#)
 CONSOLELOG_ReadConfig, [66](#)
 CONSOLELOG_Recording, [66](#)
 FLUSH_TIMEOUT_TICKS, [63](#)
 RECORD_LED_PIN
 led.h, [47](#)
 RECORD_LED_PIN_FLUSH
 led.h, [47](#)
 RECORD_LED_PORT
 led.h, [47](#)
 RECORD_STACK_SIZE
 defs.h, [38](#)
 record_task
 app_tasks.h, [32](#)
 RTC_ClearFlagAlarm1
 DS3231 Real-Time Clock Driver, [8](#)
 RTC_ClearFlagAlarm2
 DS3231 Real-Time Clock Driver, [8](#)
 RTC_ConvertToBCD
 DS3231 Real-Time Clock Driver, [8](#)
 rtc_ds3231.c, [19](#)
 RTC_ConvertToDEC
 DS3231 Real-Time Clock Driver, [8](#)
 rtc_ds3231.c, [19](#)
 RTC_CtrlAlarm1
 DS3231 Real-Time Clock Driver, [9](#)
 RTC_CtrlAlarm2
 DS3231 Real-Time Clock Driver, [9](#)
 RTC_date_t, [15](#)
 date, [15](#)
 day, [15](#)
 month, [15](#)
 year, [15](#)
 RTC_Deinit
 DS3231 Real-Time Clock Driver, [9](#)
 rtc_ds3231.c
 AT_NONCACHEABLE_SECTION, [19](#)
 BYTE_1, [18](#)
 BYTE_2, [18](#)
 CLK_STATE, [18](#)
 gCompletionFlag, [20](#)
 gEdmaRxHandle, [20](#)
 gEdmaTxHandle, [20](#)
 I2C_DATA_LENGTH, [19](#)
 lpi2c_callback, [19](#)
 RTC_ConvertToBCD, [19](#)
 RTC_ConvertToDEC, [19](#)
 xfer, [20](#)
 rtc_ds3231.h
 ALARM_DISABLED, [23](#)
 ALARM_INTERRUPT, [28](#)
 APP_SUCCESS, [23](#)
 DS3231_A1F, [23](#)
 DS3231_A1IE, [23](#)
 DS3231_A2F, [23](#)
 DS3231_A2IE, [23](#)
 DS3231_ADDR_CENT, [23](#)
 DS3231_ADDR_DATE, [23](#)
 DS3231_ADDR_DAY, [23](#)
 DS3231_ADDR_HRS, [24](#)
 DS3231_ADDR_I2C, [24](#)
 DS3231_ADDR_MIN, [24](#)
 DS3231_ADDR_MONTH, [24](#)
 DS3231_ADDR_SEC, [24](#)
 DS3231_ADDR_YEAR, [24](#)
 DS3231_INTCN, [24](#)
 DS3231_REG_CTRL, [25](#)
 DS3231_REG_STATUS, [25](#)
 E_FAULT, [25](#)
 FRIDAY, [25](#)
 I2C_BAUDRATE, [25](#)
 I2C_MASTER, [25](#)
 LPI2C_RX_DMA_CHANNEL, [25](#)
 LPI2C_RX_EDMA_CHANNEL, [26](#)
 LPI2C_TX_CHANNEL, [26](#)
 LPI2C_TX_DMA_CHANNEL, [26](#)
 MONDAY, [26](#)
 OSC_INTERRUPTED, [28](#)

- OSC_OK, [28](#)
- OSC_STOPPED, [26](#)
- RTC_interrupt_mode_t, [27](#)
- RTC_osc_state_t, [28](#)
- SATURDAY, [26](#)
- SQUARE_WAVE_INTERRUPT, [28](#)
- SUNDAY, [26](#)
- THURSDAY, [27](#)
- TIM_CYCLE_12H, [27](#)
- TIM_CYCLE_12H_AM, [27](#)
- TIM_CYCLE_12H_PM, [27](#)
- TIM_CYCLE_24H, [27](#)
- TUESDAY, [27](#)
- WEDNESDAY, [27](#)
- RTC_GetDate
 - DS3231 Real-Time Clock Driver, [9](#)
- RTC_GetState
 - DS3231 Real-Time Clock Driver, [10](#)
- RTC_GetTime
 - DS3231 Real-Time Clock Driver, [10](#)
- RTC_Init
 - DS3231 Real-Time Clock Driver, [10](#)
- RTC_interrupt_mode_t
 - rtc_ds3231.h, [27](#)
- RTC_osc_state_t
 - rtc_ds3231.h, [28](#)
- RTC_Read
 - DS3231 Real-Time Clock Driver, [10](#)
- RTC_SetDate
 - DS3231 Real-Time Clock Driver, [11](#)
- RTC_SetDateDefault
 - DS3231 Real-Time Clock Driver, [11](#)
- RTC_SetInterruptMode
 - DS3231 Real-Time Clock Driver, [11](#)
- RTC_SetOscState
 - DS3231 Real-Time Clock Driver, [11](#)
- RTC_SetTime
 - DS3231 Real-Time Clock Driver, [12](#)
- RTC_SetTimeDefault
 - DS3231 Real-Time Clock Driver, [12](#)
- RTC_time_t, [16](#)
 - format, [16](#)
 - hrs, [16](#)
 - min, [16](#)
 - sec, [16](#)
- RTC_Write
 - DS3231 Real-Time Clock Driver, [12](#)
- SATURDAY
 - rtc_ds3231.h, [26](#)
- sec
 - RTC_time_t, [16](#)
- size
 - REC_config_t, [14](#)
- SPC_BASE
 - pwrloss_det.h, [60](#)
- SQUARE_WAVE_INTERRUPT
 - rtc_ds3231.h, [28](#)
- stop_bits
 - REC_config_t, [14](#)
- SUNDAY
 - rtc_ds3231.h, [26](#)
- TASK_PRIO
 - app_tasks.h, [32](#)
- task_switching.h
 - g_msc, [68](#)
 - USB_State, [68](#)
- TAU5
 - defs.h, [38](#)
- temperature.h
 - Read, [69](#)
 - TMP_GetTemperature, [70](#)
 - TMP_Init, [70](#)
 - Write, [70](#)
- TEMPERATURE_MEAS_ENABLED
 - defs.h, [39](#)
- THURSDAY
 - rtc_ds3231.h, [27](#)
- TIM_CYCLE_12H
 - rtc_ds3231.h, [27](#)
- TIM_CYCLE_12H_AM
 - rtc_ds3231.h, [27](#)
- TIM_CYCLE_12H_PM
 - rtc_ds3231.h, [27](#)
- TIM_CYCLE_24H
 - rtc_ds3231.h, [27](#)
- time.h
 - LPI2C_DMA_BASEADDR, [72](#)
 - TIME_InitIRTC, [72](#)
 - TIME_InitIRTC
 - time.h, [72](#)
 - TMP_GetTemperature
 - temperature.h, [70](#)
 - TMP_Init
 - temperature.h, [70](#)
- TUESDAY
 - rtc_ds3231.h, [27](#)
- uart.h
 - LPUART3_CLK_FREQ, [73](#)
 - UART_Deinit, [73](#)
 - UART_Disable, [73](#)
 - UART_Enable, [74](#)
 - UART_Init, [74](#)
 - UART_Print, [74](#)
 - UART_Deinit
 - uart.h, [73](#)
 - UART_Disable
 - uart.h, [73](#)
 - UART_Enable
 - uart.h, [74](#)
 - UART_FIFO_ENABLED
 - defs.h, [39](#)
 - UART_FIFO_LENGTH
 - defs.h, [39](#)
 - UART_Init
 - uart.h, [74](#)

- UART_Print
 - uart.h, [74](#)
- UART_PRINT_ENABLED
 - defs.h, [39](#)
- UART_RECEIVE_PRIO
 - defs.h, [39](#)
- USB_State
 - task_switching.h, [68](#)

- vApplicationGetIdleTaskMemory
 - app_tasks.h, [33](#)
- vApplicationGetTimerTaskMemory
 - app_tasks.h, [33](#)
- version
 - REC_config_t, [14](#)

- WCT_AUTOS1
 - parser.h, [53](#)
- WCT_AUTOS2
 - parser.h, [53](#)
- WCT_UNKOWN
 - parser.h, [53](#)
- WEDNESDAY
 - rtc_ds3231.h, [27](#)
- Write
 - temperature.h, [70](#)

- xfer
 - rtc_ds3231.c, [20](#)

- year
 - RTC_date_t, [15](#)