My Project

Generated by Doxygen 1.13.2

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Topic Documentation

4.1 Task Management

Group Contains Task Management Variables and Metadata.

Variables

- static StaticTask_t xIdleTaskTCB

  TCB (Task Control Block) - Meta-Data of IDLE Task.
- static StackType_t uxIdleTaskStack [configMINIMAL_STACK_SIZE]

  Stack for Static Idle Task.
- static StaticTask_t xTimerTaskTCB

  TCB (Task Control Block) - Meta-Data of Timer Task.
- static StackType_t uxTimerTaskStack [configTIMER_TASK_STACK_DEPTH]

  Stack for Static Timer Task.

4.1.1 Detailed Description

Group Contains Task Management Variables and Metadata.

MISRA Deviation: Rule 21.10 Suppress: Use Of Banned Standard Header 'Time.h'. Justification: 'time.h' Is Used For Generating Timestamps.

4.1.2 Variable Documentation

4.1.2.1 uxIdleTaskStack

```
StackType_t uxIdleTaskStack[configMINIMAL_STACK_SIZE]  [static]
```

Stack for Static Idle Task.

4.1.2.2 uxTimerTaskStack

```
StackType_t uxTimerTaskStack[configTIMER_TASK_STACK_DEPTH]  [static]
```

Stack for Static Timer Task.

MISRA Deviation Note: Rule: MISRA 2012 Rule 10.4 [Required] Suppress: Standard FreeRTOS Macro Cast To uint32_t Justification: The macro 'configTIMER_TASK_STACK_DEPTH' is defined by the FreeRTOS kernel configuration as an integer constant.

4.1.2.3 xIdleTaskTCB

```
StaticTask_t xIdleTaskTCB  [static]
```

TCB (Task Control Block) - Meta-Data of IDLE Task.

4.1.2.4 xTimerTaskTCB

```
StaticTask_t xTimerTaskTCB  [static]
```

TCB (Task Control Block) - Meta-Data of Timer Task.

4.2 Buffers and Recording Management

Group Contains Variables For Management of Recording.

Functions

- SDK_ALIGN (static uint8_t g_au8DmaBuffer1[BLOCK_SIZE], BOARD_SDMMC_DATA_BUFFER_ALIGN_↩ SIZE)

  Buffer For Multi-Buffering - In Particular Dual-Buffering, One Is Always Filled, The Other Is Processed.

Variables

- static FATFS g_fileSystem

  File System Object.
- static FIL g_fileObject

  File Object.
- static char g_u8CurrentDirectory [32]

  Name Of The Folder Where The Files (Logs) From The Current Session Are Stored.
- static uint8_t * g_pu8BackDmaBuffer = g_au8DmaBuffer1

  Back Buffer Which Serves For Data Collection From Circular Buffer And Is Used For Data-Processing (Time Stamps Are Inserted To This Buffer).
- static uint8_t * g_pu8FrontDmaBuffer = NULL

  Front Buffer Which Serves For Storing Data Into SD Card.
- static uint16_t g_u16BackDmaBufferIdx = 0

  Pointer on Current Back DMA Buffer Into Which The Time Stamps Are Inserted.

- static bool g_bBackDmaBufferReady = false

  Indicates That Collection Buffer (Back Buffer) Is Full and Ready To Swap.
- static TickType_t g_lastDataTick = 0

  Value of Ticks When Last Character Was Received Thru LPUART.
- static uint32_t g_u32CurrentFileSize = 0

  Tracks Current File Size.
- static uint16_t g_u32FileCounter = 1

  Counter For Unique File Names.
- static bool g_bFlushCompleted = false

  Flush Completed Flag.
- static uint32_t g_u32BytesTransfered = 0U

  Transferred Bytes Between Blinking LEDs.

4.2.1 Detailed Description

Group Contains Variables For Management of Recording.

Includes DMA Buffers For Recording, Indexes, Pointers, ...

4.2.2 Function Documentation

4.2.2.1 SDK_ALIGN()

```
SDK_ALIGN (
            static uint8_t g_au8DmaBuffer1[BLOCK_SIZE],
            BOARD_SDMMC_DATA_BUFFER_ALIGN_SIZE )
```

Buffer For Multi-Buffering - In Particular Dual-Buffering, One Is Always Filled, The Other Is Processed.

Must Be Aligned on Multiple of 512B, Since SDHC/SDXC Card Uses 512-Byte Fixed Block Length. The Address of The R/W Buffer Should Be Also Align To The Specific DMA Data Buffer Address Align Value. At The Same Time Buffer Address/Size Should Be Aligned To The Cache Line Size.

4.2.3 Variable Documentation

4.2.3.1 g_bBackDmaBufferReady

```
bool g_bBackDmaBufferReady = false  [static]
```

Indicates That Collection Buffer (Back Buffer) Is Full and Ready To Swap.

4.2.3.2 g_bFlushCompleted

```
bool g_bFlushCompleted = false  [static]
```

Flush Completed Flag.

If No Data of The LPUART Periphery Are Received Within The `FLUSH_TIMEOUT_TICKS` Interval, The Data Collected So Far In The Buffer Are Flushed To The File.

4.2.3.3 g_fileObject

```
FIL g_fileObject  [static]
```

File Object.

Pointer to Current Opened File.

4.2.3.4 g_fileSystem

```
FATFS g_fileSystem  [static]
```

File System Object.

4.2.3.5 g_lastDataTick

```
TickType_t g_lastDataTick = 0  [static]
```

Value of Ticks When Last Character Was Received Thru LPUART.

4.2.3.6 g_pu8BackDmaBuffer

```
uint8_t* g_pu8BackDmaBuffer = g_au8DmaBuffer1  [static]
```

Back Buffer Which Serves For Data Collection From Circular Buffer And Is Used For Data-Processing (Time Stamps Are Inserted To This Buffer).

4.2.3.7 g_pu8FrontDmaBuffer

```
uint8_t* g_pu8FrontDmaBuffer = NULL  [static]
```

Front Buffer Which Serves For Storing Data Into SD Card.

4.2.3.8 g_u16BackDmaBufferIdx

```
uint16_t g_u16BackDmaBufferIdx = 0  [static]
```

Pointer on Current Back DMA Buffer Into Which The Time Stamps Are Inserted.

4.2.3.9 g_u32BytesTransfered

```
uint32_t g_u32BytesTransfered = 0U  [static]
```

Transferred Bytes Between Blinking LEDs.

4.2.3.10 g_u32CurrentFileSize

```
uint32_t g_u32CurrentFileSize = 0  [static]
```

Tracks Current File Size.

4.2.3.11 g_u32FileCounter

```
uint16_t g_u32FileCounter = 1  [static]
```

Counter For Unique File Names.

4.2.3.12 g_u8CurrentDirectory

```
char g_u8CurrentDirectory[32]  [static]
```

Name Of The Folder Where The Files (Logs) From The Current Session Are Stored.

4.3 Management

Group Contains Variables For Recording From UART.

Functions

- SDK_ALIGN (static volatile uint8_t g_au8CircBuffer[CIRCULAR_BUFFER_SIZE], BOARD_SDMMC_DATA↩ _BUFFER_ALIGN_SIZE)

  Circular Buffer For Reception of Data From UART Interrupt Service Routine.

Variables

- static volatile uint32_t g_u32WriteIndex = 0

  Index For Writing Into FIFO.
- static volatile uint32_t g_u32ReadIndex = 0

  Index For Reading From FIFO.

4.3.1 Detailed Description

Group Contains Variables For Recording From UART.

Data From UART Are Stored Into FIFO (Circular Buffer).

4.3.2 Function Documentation

4.3.2.1 SDK_ALIGN()

```
SDK_ALIGN (
            static volatile uint8_t g_au8CircBuffer[CIRCULAR_BUFFER_SIZE],
            BOARD_SDMMC_DATA_BUFFER_ALIGN_SIZE )
```

Circular Buffer For Reception of Data From UART Interrupt Service Routine.

Filled in LP_FLEXCOMM3_IRQHandler Interrupt Service Routine.

4.3.3 Variable Documentation

4.3.3.1 g_u32ReadIndex

```
volatile uint32_t g_u32ReadIndex = 0  [static]
```

Index For Reading From FIFO.

4.3.3.2 g_u32WriteIndex

```
volatile uint32_t g_u32WriteIndex = 0  [static]
```

Index For Writing Into FIFO.

4.4 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.

4.4.1 Detailed Description

Real-Time Circuit Related Functions.

4.4.2 Function Documentation

4.4.2.1 RTC_ClearFlagAlarm1()

```
void RTC_ClearFlagAlarm1 (
            void )
```

Clears The A1F Flag in Control Register.

4.4.2.2 RTC_ClearFlagAlarm2()

```
void RTC_ClearFlagAlarm2 (
            void )
```

Clears The A1F Flag in Control Register.

4.4.2.3 RTC_ConvertToBCD()

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

Converts Numbers From Decimal Base To BCD Base.

Parameters

dec	Decimal Number.

Returns

Number in Binary Coded Decimal.

4.4.2.4 RTC_ConvertToDEC()

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

Converts Numbers From BCD Base To Decimal Base.

Parameters

bcd	Binary-Coded Decimal Number.

Returns

Number in Decimal Base.

4.4.2.5 RTC_CtrlAlarm1()

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

Enables/Disables The Alarm 1.

Parameters

enable	Specifies If The Alarm Will Be Enabled or Not.

4.4.2.6 RTC_CtrlAlarm2()

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

Enables/Disables The Alarm 2.

Parameters

enable	Specifies If The Alarm Will Be Enabled or Not.

4.4.2.7 RTC_Deinit()

```
void RTC_Deinit (
            void )
```

De-Initialize The RTC DS3231.

Pins Should Also Be De-Initialised Lately.

Parameters

void	

4.4.2.8 RTC_GetDate()

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

Gets Date.

Parameters

pDate	Pointer To Date Structure.

4.4.2.9 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.

4.4.2.10 RTC_GetTime()

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

Gets Time.

Parameters

pTime	Pointer To Time Structure.

4.4.2.11 RTC_Init()

```
uint8_t RTC_Init (
            void )
```

Initialize The RTC DS3231.

Parameters

void	

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;

4.4.2.12 RTC_Read()

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

Reads Value From RTC Register.

Parameters

regAddress	Address of Register From Which Will Be Read.

Returns

Read Value.

4.4.2.13 RTC_SetDate()

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

Sets Date.

Parameters

pDate	Pointer To Date Structure.

4.4.2.14 RTC_SetDateDefault()

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

Sets Date To Default.

Parameters

pDate	Pointer to Date Structure That Will Be Configured To Default.

4.4.2.15 RTC_SetInterruptMode()

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

Sets The INTCN Bit in Control Register.

Parameters

mode	Interrupt Mode.

4.4.2.16 RTC_SetOscState()

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

Sets The Oscillator Stop Flag (OSF).

Parameters

state	State of The Flag.

4.4.2.17 RTC_SetTime()

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

Sets Time.

Parameters

pTime	Pointer To Time Structure.

4.4.2.18 RTC_SetTimeDefault()

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

Sets Time To Default.

Parameters

pTime	Pointer to Time Structure That Will Be Configured To Default.

4.4.2.19 RTC_Write()

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

Writes Value Into RTC Registers.

Parameters

regAddress	Address of Register To Which Will Be Value Written.
val	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 }

  An Enum to Capture The State of The Oscillator,.
- enum RTC_interrupt_mode_t { SQUARE_WAVE_INTERRUPT = 0 , ALARM_INTERRUPT }

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
00159 } RTC_time_t;
00160
00164 typedef struct
00165 {
00166     uint8_t date;
00167     uint8_t day;
00168     uint8_t month;
00169     uint8_t year;
00170
00171 } RTC_date_t;
00172
00176 typedef enum
00177 {
00178     OSC_OK = 0,
00179     OSC_INTERRUPTED
00180
00181 } RTC_osc_state_t;
00182
00183 /*
00184  * @brief An Enum For Interrupt Mode.
00185  */
00186 typedef enum
00187 {
00188     SQUARE_WAVE_INTERRUPT = 0,
00189     ALARM_INTERRUPT
00190
00191 }RTC_interrupt_mode_t;
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
00056 void APP_InitBoard(void);
00057
00058
00059 #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

  USB Mass Storage Task Handle.
- TaskHandle_t g_xRecordTaskHandle

  Record Task Handle.
- SemaphoreHandle_t g_xSemRecord

  Semaphore For Record Task Management.
- SemaphoreHandle_t g_xSemMassStorage

  Semaphore For USB Mass Storage Task Management.

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

handle	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

handle	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 config↩ SUPPORT_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	*ppxIdleTaskTCBBuffer*	Pointer to The TCB Buffer For The Idle Task.
out	*ppxIdleTaskStackBuffer*	Pointer to The Stack Buffer For The Idle Task.
out	*pulIdleTaskStackSize*	Pointer to a Variable Holding The Stack Size.

MISRA Deviation: Rule 10.8 [Required] Suppress: Conversion From Signed Macro To Unsigned Type. Justification: 'ConfigMINIMAL_STACK_SIZE' Is Defined By FreeRTOS As A Signed Macro. The Conversion To 'Uint32_t' Is Intentional And Safe In This Context. Fixing The Definition is Not Possible.

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 config↩ SUPPORT_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	*ppxTimerTaskTCBBuffer*	Pointer to The TCB Buffer For The Timer Task.
out	*ppxTimerTaskStackBuffer*	Pointer to The Stack Buffer For The Timer Task.
out	*pulTimerTaskStackSize*	Pointer to a Variable Holding The Stack Size.

MISRA Deviation: Rule 10.8 [Required] Suppress: Conversion From Signed Macro To Unsigned Type. Justification: 'configTIMER_TASK_STACK_DEPTH' Is Defined By FreeRTOS As A Signed Macro. The Conversion To 'Uint32_t' Is Intentional And Safe In This Context. Fixing The Definition is Not Possible.

6.6.3 Variable Documentation

6.6.3.1 g_xMscTaskHandle

```
TaskHandle_t g_xMscTaskHandle  [extern]
```

USB Mass Storage Task Handle.

6.6.3.2 g_xRecordTaskHandle

```
TaskHandle_t g_xRecordTaskHandle  [extern]
```

Record Task Handle.

6.6.3.3 g_xSemMassStorage

```
SemaphoreHandle_t g_xSemMassStorage  [extern]
```

Semaphore For USB Mass Storage Task Management.

6.6.3.4 g_xSemRecord

```
SemaphoreHandle_t g_xSemRecord  [extern]
```

Semaphore For Record Task Management.

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_LENGHT (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 16.5

  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 More 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 Info Mode Is Enabled Then Informational 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 16.5
```

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

In Seconds (It's Actually 16.5s).

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_LENGHT

```
#define UART_FIFO_LENGHT (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 /*******************************************************************************
00028  * Definitions
00029  ******************************************************************************/
00030 #define NOT_IMPLEMENTED             #error "This Feature Is Not Implemented!"
00031
00035 #define MSC_STACK_SIZE        ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
00036
00040 #define RECORD_STACK_SIZE     ((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))
00041
00045 #define MSC_ENABLED                 (true)
00046
00051 #define INFO_ENABLED                (false)
00052
00057 #define DEBUG_ENABLED               (false)
00058
00062 #define UART_FIFO_ENABLED           (true)
00063
00067 #define UART_FIFO_LENGHT            (4u)
00068
00072 #define UART_PRINT_ENABLED          (false)
00073
00077 #define PWRLOSS_DETECTION_ENABLED   (true)
00078
00082 #define PWRLOSS_TEST_GPIOS          (false)
00083
00088 #define TAU5                        16.5
00089
00094 #define PWRLOSS_DET_ACTIVE_IN_TIME  TAU5
00095
00099 #define UART_RECEIVE_PRIO           (6U)
00100
```

```
00104 #define PWRLOSS_DET_PRIO            (5U)
00105
00109 #define PWRLOSS_TIMER_PRIO          (7U)
00110
00111
00115 #define TEMPERATURE_MEAS_ENABLED    (false)
00116
00120 #define CONTROL_LED_ENABLED         (true)
00121
00122
00127 #define DEFAULT_MAX_FILESIZE        8192
00128
00132 #define CONFIG_FILE                 "config"
00133
00138 #define DEFAULT_BAUDRATE            230400UL
00139
00144 #define DEFAULT_DATA_BITS           kLPUART_EightDataBits
00145
00150 #define DEFAULT_STOP_BITS           kLPUART_OneStopBit
00151
00156 #define DEFAULT_PARITY              kLPUART_ParityDisabled
00157
00163 #define DEFAULT_FREE_SPACE          50UL
00164
00165 #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 ()
```

MISRA Deviation Note: Rule: MISRA 2012 Rule 8.4 [Required] Justification: The Function 'ERR_Init', Usage Here is Compliant With The Intended Structure of The Project.

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-Inicialization.

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-Inicialization.

6.12.2.2 LED_SetHigh()

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

Sets Logic 1 on GPIO Pin.

Parameters

port_base	Pointer to GPIO Instance.
pin	Pin.

6.12.2.3 LED_SetLow()

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

Sets Logic 0 on GPIO Pin.

Parameters

port_base	Pointer to GPIO Instance.
pin	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  ******************************************************************************/
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. MISRA Deviation: Rule 2.2 Justification: Function 'MSC_DeviceMscApp' is Called Periodically and Allows To Expand USB Mass Storage.

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	*chContent*	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	*chContent*	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	*chContent*	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	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

MISRA Deviation: Rule 21.6 [Required] Suppress: Use of Standard Library Function 'strtoul'. Justification: 'strtoul' is Used With Trusted Input For Converting a String to an Unsigned Long Value. The Usage is Controlled and Verified, Ensuring That The Input Cannot Cause Unexpected Behavior.

6.16.2.15 PARSER_ParseParity()

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

Parse Parity From Configuration File.

Parameters

in	*chContent*	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	*chContent*	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 PARSER_H_
00018 #define PARSER_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  ******************************************************************************/
00048 typedef enum
00049 {
00050     WCT_UNKOWN = 0,
00051     WCT_AUTOS1,
00052     WCT_AUTOS2
00053
00054 } REC_version_t;
00055
00062 typedef struct
00063 {
00064     /* Recorded NXP Device  */
00065     REC_version_t  version;
00066
00067     /* UART Setup */
00068     uint32_t                baudrate;
00069     lpuart_stop_bit_count_t stop_bits;
00070     lpuart_data_bits_t      data_bits;
00071     lpuart_parity_mode_t    parity;
00072
00073
00074     uint32_t        size;
00076     uint32_t        max_bytes;
00077     uint32_t        free_space_limit_mb;
00079
00080 } REC_config_t;
00081 /*******************************************************************************
00082  * Prototypes
```

```
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$5tau). 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_GetTransferedBytes (void)

  Returns Currently Received Bytes Between LED Blinking.
- bool CONSOLELOG_GetFlushCompleted (void)

  Returns If Flush Was Completed or Not.
- void CONSOLELOG_ClearTransferedBytes (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_ClearTransferedBytes()

```
void CONSOLELOG_ClearTransferedBytes (
            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.

MISRA Deviation: Rule 21.6 [Advisory] Suppress: Use Of Standard Library Function 'snprintf()' Which Is Not Fully Bounded In All Environments. Justification: The Use Of 'snprintf()' Is Intentional And Acceptable In This Context, The Format String And All Input Values Are Controlled and Predictable.

MISRA Deviation: Rule 21.6 [Advisory] Suppress: Use Of Standard Library Function 'snprintf()' Which Is Not Fully Bounded In All Environments. Justification: The Use Of 'snprintf()' Is Intentional And Acceptable In This Context, The Format String And All Input Values Are Controlled and Predictable.

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.

MISRA Deviation: Rule 21.6 Suppress: Use Of Standard Library Function 'Snprintf'. Justification: 'Snprintf' Is Deprecated But There Is No Better Equivalent For Safe String Formatting.

MISRA Deviation: Rule 10.1 [Required] Suppress: Bitwise Operation on Composite Constant Expression. Justification: FA_WRITE and FA_CREATE_ALWAYS are Standard Bitmask Flags Defined By The FatFs Library. These Constants Are Designed To Be Combined Using Bitwise OR, and The Use is Safe and Intentional.

MISRA Deviation Note: The Following Expression is Intentionally Written in a Compact and Readable Form For Setting FAT File Time Stamps. All Shifts and Bitwise Operations Are Used Correctly, Even Though They Trigger MISRA Rule 10.1, 10.3, 10.4, 10.7, 12.2 Warnings. These are Suppressed Here For Clarity and Maintainability.

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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

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_GetTransferedBytes()

```
uint32_t CONSOLELOG_GetTransferedBytes (
            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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

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

in	*content*	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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

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_GetTransferedBytes(void);
00089
00095 bool CONSOLELOG_GetFlushCompleted(void);
00096
00100 void CONSOLELOG_ClearTransferedBytes(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	*pDeviceHandle*	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.

Mass Storage Descriptor.

MISRA Deviation: Rule 8.4 [Required] Suppress: External Object Has Definition Without Prior Declaration in This File. Justification: Declaration of 'g_msc' is intentionally placed in 'app_tasks.h', The Current File Only Provides The Definition.

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  ******************************************************************************/
00032 extern usb_msc_struct_t g_msc;
00033 /*******************************************************************************
00034  * Prototypes
00035  ******************************************************************************/
00043 usb_device_notification_t USB_State(usb_device_struct_t *pDeviceHandle);
00044
00045
00046 #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

in	*regAddress*	Address of The Register From Which The Reading Will Take Place.

Return values

Returns	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

Returns	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

in	*regAddress*	Address of The Register To Be Written To.
in	*val*	Array of Values To Be Written To The Register.

Return values

If	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:      mainc.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.

MISRA Deviation Note: Rule 10.3: Cannot Assign 'enum' to a Different Essential Type Such As 'unsigned32'. [Required] Rule 11.4: Conversion Between Object Pointer Type and Integer Type. [Required] Justification: This Code Follows The Usage Pattern Provided By The NXP SDK.

6.28.2.4 UART_Init()

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

Initializes LPUART For Recording.

MISRA Deviation: Rule 11.4 [Required] Suppress: Conversion Between Object Pointer Type 'LPUART_Type ∗' and Integer Type 'Unsigned Int'. Justification: LPUART3 Is a Hardware Peripheral Base Address Defined In The NXP SDK. This Code Follows The Usage Pattern Provided By The NXP SDK.

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
```

6.30 src/app_init.c File Reference

```
#include "app_tasks.h"
#include "app_init.h"
```

Functions

- void APP_InitBoard (void)

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

6.30.1 Function Documentation

6.30.1.1 APP_InitBoard()

```
void APP_InitBoard (
            void )
```

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

6.31 src/app_tasks.c File Reference

```
#include "app_tasks.h"
#include "rtc_ds3231.h"
#include "mass_storage.h"
#include "task_switching.h"
#include "record.h"
#include <time.h>
```

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 **ppxIdleTask↩ StackBuffer, uint32_t *pulIdleTaskStackSize)

  Hook Function to Provide Memory For The Idle Task in FreeRTOS.
- void vApplicationGetTimerTaskMemory (StaticTask_t **ppxTimerTaskTCBBuffer, StackType_t **ppxTimer↩ TaskStackBuffer, uint32_t *pulTimerTaskStackSize)

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

Variables

- static StaticTask_t xIdleTaskTCB

  TCB (Task Control Block) - Meta-Data of IDLE Task.
- static StackType_t uxIdleTaskStack [configMINIMAL_STACK_SIZE]

  Stack for Static Idle Task.
- static StaticTask_t xTimerTaskTCB

  TCB (Task Control Block) - Meta-Data of Timer Task.
- static StackType_t uxTimerTaskStack [configTIMER_TASK_STACK_DEPTH]

  Stack for Static Timer Task.

6.31.1 Function Documentation

6.31.1.1 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

handle	Pointer to The Device Handle Used For The USB Operations.

6.31.1.2 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

handle	Pointer to The Device Handle Used For The USB Operations.

6.31.1.3 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 config↩ SUPPORT_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	*ppxIdleTaskTCBBuffer*	Pointer to The TCB Buffer For The Idle Task.
out	*ppxIdleTaskStackBuffer*	Pointer to The Stack Buffer For The Idle Task.
out	*pulIdleTaskStackSize*	Pointer to a Variable Holding The Stack Size.

MISRA Deviation: Rule 10.8 [Required] Suppress: Conversion From Signed Macro To Unsigned Type. Justification: 'ConfigMINIMAL_STACK_SIZE' Is Defined By FreeRTOS As A Signed Macro. The Conversion To 'Uint32_t' Is Intentional And Safe In This Context. Fixing The Definition is Not Possible.

6.31.1.4 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 config↩ SUPPORT_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	*ppxTimerTaskTCBBuffer*	Pointer to The TCB Buffer For The Timer Task.
out	*ppxTimerTaskStackBuffer*	Pointer to The Stack Buffer For The Timer Task.
out	*pulTimerTaskStackSize*	Pointer to a Variable Holding The Stack Size.

MISRA Deviation: Rule 10.8 [Required] Suppress: Conversion From Signed Macro To Unsigned Type. Justification: 'configTIMER_TASK_STACK_DEPTH' Is Defined By FreeRTOS As A Signed Macro. The Conversion To 'Uint32_t' Is Intentional And Safe In This Context. Fixing The Definition is Not Possible.

6.32 src/error.c File Reference

```
#include "error.h"
#include "stdbool.h"
```

Functions

- void ERR_HandleError ()

  Base Function For Error Handling.
- void ERR_Init ()
- void ERR_SetState (error_t err)

Variables

- error_t error

6.32.1 Function Documentation

6.32.1.1 ERR_HandleError()

```
void ERR_HandleError (
            void )
```

Base Function For Error Handling.

6.32.1.2 ERR_Init()

```
void ERR_Init ()
```

MISRA Deviation Note: Rule: MISRA 2012 Rule 8.4 [Required] Justification: The Function 'ERR_Init', Usage Here is Compliant With The Intended Structure of The Project.

6.32.1.3 ERR_SetState()

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

6.32.2 Variable Documentation

6.32.2.1 error

```
error_t error
```

MISRA Deviation Note: Rule: MISRA 2012 Rule 8.4 [Required] Justification: Variable 'error' is Declared in 'error.h', Usage Here is Compliant With The Intended Structure of The Project.

6.33 src/led.c File Reference

```
#include <led.h>
```

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-Inicialization.

6.33.1 Function Documentation

6.33.1.1 LED_ClearSignalFlush()

```
void LED_ClearSignalFlush (
            void )
```

Clears Flush LED After UART Re-Inicialization.

6.33.1.2 LED_SetHigh()

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

Sets Logic 1 on GPIO Pin.

Parameters

port_base	Pointer to GPIO Instance.
pin	Pin.

6.33.1.3 LED_SetLow()

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

Sets Logic 0 on GPIO Pin.

Parameters

port_base	Pointer to GPIO Instance.
pin	Pin.

6.33.1.4 LED_SignalBackUpPowerAvailable()

```
void LED_SignalBackUpPowerAvailable (
            void )
```

Signals Configuration File Missing or Contains Unexpected Data.

6.33.1.5 LED_SignalError()

```
void LED_SignalError (
            void )
```

Signals Error During Recording.

6.33.1.6 LED_SignalFlush()

```
void LED_SignalFlush (
            void )
```

Signals That Flush Has Been Activated.

6.33.1.7 LED_SignalLowMemory()

```
void LED_SignalLowMemory (
            void )
```

Signals That There Will Be Empty Space on SD Card.

6.33.1.8 LED_SignalReady()

```
void LED_SignalReady (
            void )
```

Indicates Ready State of The Recording Device.

6.33.1.9 LED_SignalRecording()

```
void LED_SignalRecording (
            void )
```

Signals That Device Is Currently Receiving Bytes (Recording).

6.33.1.10 LED_SignalRecordingStop()

```
void LED_SignalRecordingStop (
            void )
```

Signals That Device Is Stopped Receiving Bytes (Recording).

6.34 src/main.c 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 "semphr.h"
#include "app_init.h"
#include "app_tasks.h"
```

Functions

- int main (void)

  Application Entry Point.

Variables

- static StackType_t g_xMscTaskStack [MSC_STACK_SIZE]

  Buffer for Static Stack of Mass Storage Task.
- static StaticTask_t g_xMscTaskTCB

  TCB (Task Control Block) - Meta Data of Mass Storage Task.
- static StackType_t g_xRecordTaskStack [RECORD_STACK_SIZE]

  Buffer For Static Stack Of Record Task.
- static StaticTask_t g_xRecordTaskTCB

  TCB (Task Control Block) - Meta Data of Record Task.
- usb_msc_struct_t g_msc

  Global USB MSC Structure.
- TaskHandle_t g_xMscTaskHandle = NULL

  USB Mass Storage Task Handle.
- SemaphoreHandle_t g_xSemMassStorage

  Semaphore For USB Mass Storage Task Management.
- TaskHandle_t g_xRecordTaskHandle = NULL

  Record Task Handle.
- SemaphoreHandle_t g_xSemRecord

  Semaphore For Record Task Management.

6.34.1 Function Documentation

6.34.1.1 main()

```
int main (
            void )
```

Application Entry Point.

6.34.2 Variable Documentation

6.34.2.1 g_msc

```
usb_msc_struct_t g_msc
```

Global USB MSC Structure.

Mass Storage Descriptor.

MISRA Deviation: Rule 8.4 [Required] Suppress: External Object Has Definition Without Prior Declaration in This File. Justification: Declaration of 'g_msc' is intentionally placed in 'app_tasks.h', The Current File Only Provides The Definition.

6.34.2.2 g_xMscTaskHandle

```
TaskHandle_t g_xMscTaskHandle = NULL
```

USB Mass Storage Task Handle.

6.34.2.3 g_xMscTaskStack

```
StackType_t g_xMscTaskStack[MSC_STACK_SIZE]  [static]
```

Buffer for Static Stack of Mass Storage Task.

6.34.2.4 g_xMscTaskTCB

```
StaticTask_t g_xMscTaskTCB  [static]
```

TCB (Task Control Block) - Meta Data of Mass Storage Task.

Includes All The Information Needed to Manage The Task Such As Job Status, Job Stack Pointer, Values of Variables During Context Switching.

6.34.2.5 g_xRecordTaskHandle

```
TaskHandle_t g_xRecordTaskHandle = NULL
```

Record Task Handle.

6.34.2.6 g_xRecordTaskStack

```
StackType_t g_xRecordTaskStack[RECORD_STACK_SIZE]  [static]
```

Buffer For Static Stack Of Record Task.

6.34.2.7 g_xRecordTaskTCB

```
StaticTask_t g_xRecordTaskTCB  [static]
```

TCB (Task Control Block) - Meta Data of Record Task.

Includes All The Information Needed to Manage The Task Such As Job Status, Job Stack Pointer, Values of Variables During Context Switching.

6.34.2.8 g_xSemMassStorage

```
SemaphoreHandle_t g_xSemMassStorage
```

Semaphore For USB Mass Storage Task Management.

6.34.2.9 g_xSemRecord

```
SemaphoreHandle_t g_xSemRecord
```

Semaphore For Record Task Management.

6.35 src/mass_storage.c File Reference

```
#include "mass_storage.h"
#include "task_switching.h"
```

Functions

- void USB1_HS_IRQHandler (void)
- void MSC_DeviceMscApp (void)

  Process Extension to Mass Storage.
- void MSC_DeviceMscAppTask (void)

  Handles Mass Storage Application.

Variables

- SemaphoreHandle_t g_xSemRecord

  Semaphore For Record Task Management.
- SemaphoreHandle_t g_xSemMassStorage

  Semaphore For USB Mass Storage Task Management.

6.35.1 Function Documentation

6.35.1.1 MSC_DeviceMscApp()

```
void MSC_DeviceMscApp (
            void )
```

Process Extension to Mass Storage.

6.35.1.2 MSC_DeviceMscAppTask()

```
void MSC_DeviceMscAppTask (
            void )
```

Handles Mass Storage Application.

Communication and Data Transport Is Handled By USB1_HS ISR. MISRA Deviation: Rule 2.2 Justification: Function 'MSC_DeviceMscApp' is Called Periodically and Allows To Expand USB Mass Storage.

6.35.1.3 USB1_HS_IRQHandler()

```
void USB1_HS_IRQHandler (
            void )
```

Switch The Context From ISR To a Higher Priority Task, Without Waiting For The Next Scheduler Tick.

6.35.2 Variable Documentation

6.35.2.1 g_xSemMassStorage

```
SemaphoreHandle_t g_xSemMassStorage  [extern]
```

Semaphore For USB Mass Storage Task Management.

6.35.2.2 g_xSemRecord

```
SemaphoreHandle_t g_xSemRecord  [extern]
```

Semaphore For Record Task Management.

6.36 src/parser.c File Reference

```
#include "parser.h"
```

Macros

- #define RECORD_LED_TIME_INTERVAL (uint32_t)(10U)

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.

Variables

- static REC_config_t g_config

  Configuration of The Data Logger on The Basis of Data Obtained From The Configuration File.

6.36.1 Macro Definition Documentation

6.36.1.1 RECORD_LED_TIME_INTERVAL

```
#define RECORD_LED_TIME_INTERVAL (uint32_t)(10U)
```

6.36.2 Function Documentation

6.36.2.1 PARSER_ClearConfig()

```
void PARSER_ClearConfig (
            void )
```

Clears The Configuration To Default.

6.36.2.2 PARSER_GetBaudrate()

```
uint32_t PARSER_GetBaudrate (
            void )
```

Returns the Baudrate of The Device Being Recorded.

Returns

uint32_t Baud Rate of Recorded Device.

6.36.2.3 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.

6.36.2.4 PARSER_GetDataBits()

```
lpuart_data_bits_t PARSER_GetDataBits (
            void )
```

Returns The Number of Data Bits.

Returns

lpuart_data_bits_t Number of Data Bits.

6.36.2.5 PARSER_GetFileSize()

```
uint32_t PARSER_GetFileSize (
            void )
```

Returns the Maximal File Size.

Returns

uint32_t Maximal File Size.

6.36.2.6 PARSER_GetFreeSpaceLimitMB()

```
uint32_t PARSER_GetFreeSpaceLimitMB (
            void )
```

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

Returns

uint32_t Free Space Limit.

6.36.2.7 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.

6.36.2.8 PARSER_GetParity()

```
lpuart_parity_mode_t PARSER_GetParity (
            void )
```

Returns The Parity.

Returns

lpuart_parity_mode_t kLPUART_ParityDisabled, kLPUART_ParityEven or kLPUART_ParityOdd.

6.36.2.9 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.

6.36.2.10 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).

6.36.2.11 PARSER_ParseBaudrate()

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

Parse Baud Rate From Configuration File.

Parameters

in	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

6.36.2.12 PARSER_ParseDataBits()

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

Parse The Number of Data Bits From Configuration File.

Parameters

in	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

6.36.2.13 PARSER_ParseFileSize()

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

Parse Record File Size From Configuration File.

Parameters

in	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

6.36.2.14 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	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

MISRA Deviation: Rule 21.6 [Required] Suppress: Use of Standard Library Function 'strtoul'. Justification: 'strtoul' is Used With Trusted Input For Converting a String to an Unsigned Long Value. The Usage is Controlled and Verified, Ensuring That The Input Cannot Cause Unexpected Behavior.

6.36.2.15 PARSER_ParseParity()

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

Parse Parity From Configuration File.

Parameters

in	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

6.36.2.16 PARSER_ParseStopBits()

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

Parse The Number of Stop Bits From Configuration File.

Parameters

in	*chContent*	Pointer To Content of Configuration File.

Returns

ERROR_NONE If The Parsing Succeed.

6.36.3 Variable Documentation

6.36.3.1 g_config

```
REC_config_t g_config  [static]
```

Configuration of The Data Logger on The Basis of Data Obtained From The Configuration File.

6.37 src/pwrloss_det.c File Reference

```
#include "pwrloss_det.h"
#include "record.h"
```

Macros

- #define REF_VOLTAGE 0xFFu
- #define TRIGGER_VOLTAGE 0xE1

Functions

- void CTIMER4_IRQHandler (void)
- void HSCMP1_IRQHandler (void)
- void PWRLOSS_DetectionInit (void)

  Initializes Power Loss Detection Components.

6.37.1 Macro Definition Documentation

6.37.1.1 REF_VOLTAGE

```
#define REF_VOLTAGE 0xFFu
```

6.37.1.2 TRIGGER_VOLTAGE

```
#define TRIGGER_VOLTAGE 0xE1
```

6.37.2 Function Documentation

6.37.2.1 CTIMER4_IRQHandler()

```
void CTIMER4_IRQHandler (
            void )
```

MISRA Deviation: Rule 10.3 [Required] Suppress: Conversion From enum To unsigned32. Justification: The Enum Value is Part of The NXP SDK and is Intentionally Used in Context as a Bitmask Flag For Hardware Status Registers.

6.37.2.2 HSCMP1_IRQHandler()

```
void HSCMP1_IRQHandler (
            void )
```

6.37.2.3 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 (∼5tau). After Initialization, The Timer is Started and The Comparator Will Be Enabled Upon Timer Match Event.

6.38 src/record.c File Reference

```
#include <record.h>
#include "fsl_irtc.h"
#include <limits.h>
```

Macros

- #define CIRCULAR_BUFFER_SIZE 1024U

  Software FIFO Size.
- #define BLOCK_SIZE 512U

  Block Size For Write To SDHC Card Operation (In Bytes).
- #define GET_WAIT_INTERVAL(seconds)

  Convert Time In Seconds To Number of Ticks.
- #define GET_CURRENT_TIME_MS()

Functions

- SDK_ALIGN (static uint8_t g_au8DmaBuffer1[BLOCK_SIZE], BOARD_SDMMC_DATA_BUFFER_ALIGN_↩ SIZE)

  Buffer For Multi-Buffering - In Particular Dual-Buffering, One Is Always Filled, The Other Is Processed.
- SDK_ALIGN (static volatile uint8_t g_au8CircBuffer[CIRCULAR_BUFFER_SIZE], BOARD_SDMMC_DATA↩ _BUFFER_ALIGN_SIZE)

  Circular Buffer For Reception of Data From UART Interrupt Service Routine.
- void LP_FLEXCOMM3_IRQHandler (void)

  LPUART3 IRQ Handler.
- DWORD get_fattime (void)
- uint32_t CONSOLELOG_GetFreeSpaceMB (void)

  Gets Free Space on SD Card.
- error_t CONSOLELOG_CreateFile (void)

  Creates File Based Actual Date and Counter Value.
- error_t CONSOLELOG_CreateDirectory (void)

  Creates Directory Based Actual Date.
- uint32_t CONSOLELOG_GetTransferedBytes (void)

  Returns Currently Received Bytes Between LED Blinking.
- bool CONSOLELOG_GetFlushCompleted (void)

  Returns If Flush Was Completed or Not.

- void CONSOLELOG_ClearTransferedBytes (void)

  Clears Currently Received Bytes After 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.

Variables

- static FATFS g_fileSystem

  File System Object.
- static FIL g_fileObject

  File Object.
- static char g_u8CurrentDirectory [32]

  Name Of The Folder Where The Files (Logs) From The Current Session Are Stored.
- static uint8_t ∗ g_pu8BackDmaBuffer = g_au8DmaBuffer1

  Back Buffer Which Serves For Data Collection From Circular Buffer And Is Used For Data-Processing (Time Stamps Are Inserted To This Buffer).
- static uint8_t ∗ g_pu8FrontDmaBuffer = NULL

  Front Buffer Which Serves For Storing Data Into SD Card.
- static uint16_t g_u16BackDmaBufferIdx = 0

  Pointer on Current Back DMA Buffer Into Which The Time Stamps Are Inserted.
- static bool g_bBackDmaBufferReady = false

  Indicates That Collection Buffer (Back Buffer) Is Full and Ready To Swap.
- static TickType_t g_lastDataTick = 0

  Value of Ticks When Last Character Was Received Thru LPUART.
- static uint32_t g_u32CurrentFileSize = 0

  Tracks Current File Size.
- static uint16_t g_u32FileCounter = 1

  Counter For Unique File Names.
- static bool g_bFlushCompleted = false

  Flush Completed Flag.
- static uint32_t g_u32BytesTransfered = 0U

  Transferred Bytes Between Blinking LEDs.
- static volatile uint32_t g_u32WriteIndex = 0

  Index For Writing Into FIFO.
- static volatile uint32_t g_u32ReadIndex = 0

  Index For Reading From FIFO.

6.38.1 Macro Definition Documentation

6.38.1.1 BLOCK_SIZE

```
#define BLOCK_SIZE 512U
```

Block Size For Write To SDHC Card Operation (In Bytes).

6.38.1.2 CIRCULAR_BUFFER_SIZE

```
#define CIRCULAR_BUFFER_SIZE 1024U
```

Software FIFO Size.

6.38.1.3 GET_CURRENT_TIME_MS

```
#define GET_CURRENT_TIME_MS()
```

Value:

```
(xTaskGetTickCount() * portTICK_PERIOD_MS)
```

6.38.1.4 GET_WAIT_INTERVAL

```
#define GET_WAIT_INTERVAL(
              seconds)
```

Value:

```
((seconds) * 1000 / configTICK_RATE_HZ)
```

Convert Time In Seconds To Number of Ticks.

Parameters

seconds	Number of Seconds To Transfer.

Returns

Number of Ticks Corresponding To The Specified Number of Seconds.

6.38.2 Function Documentation

6.38.2.1 CONSOLELOG_CheckFileSystem()

```
FRESULT CONSOLELOG_CheckFileSystem (
             void )
```

Checks If The File System Is Initialized.

return F_OK If File System Initialized.

6.38.2.2 CONSOLELOG_ClearTransferedBytes()

```
void CONSOLELOG_ClearTransferedBytes (
            void )
```

Clears Currently Received Bytes After LED Blinking.

6.38.2.3 CONSOLELOG_CreateDirectory()

```
error_t CONSOLELOG_CreateDirectory (
            void )
```

Creates Directory Based Actual Date.

Returns

Returns Zero If Directory Creation Succeeded.

MISRA Deviation: Rule 21.6 [Advisory] Suppress: Use Of Standard Library Function 'snprintf()' Which Is Not Fully Bounded In All Environments. Justification: The Use Of 'snprintf()' Is Intentional And Acceptable In This Context, The Format String And All Input Values Are Controlled and Predictable.

MISRA Deviation: Rule 21.6 [Advisory] Suppress: Use Of Standard Library Function 'snprintf()' Which Is Not Fully Bounded In All Environments. Justification: The Use Of 'snprintf()' Is Intentional And Acceptable In This Context, The Format String And All Input Values Are Controlled and Predictable.

6.38.2.4 CONSOLELOG_CreateFile()

```
error_t CONSOLELOG_CreateFile (
            void )
```

Creates File Based Actual Date and Counter Value.

Returns

Returns Zero If File Creation Succeeded.

MISRA Deviation: Rule 21.6 Suppress: Use Of Standard Library Function 'Snprintf'. Justification: 'Snprintf' Is Deprecated But There Is No Better Equivalent For Safe String Formatting.

MISRA Deviation: Rule 10.1 [Required] Suppress: Bitwise Operation on Composite Constant Expression. Justification: FA_WRITE and FA_CREATE_ALWAYS are Standard Bitmask Flags Defined By The FatFs Library. These Constants Are Designed To Be Combined Using Bitwise OR, and The Use is Safe and Intentional.

MISRA Deviation Note: The Following Expression is Intentionally Written in a Compact and Readable Form For Setting FAT File Time Stamps. All Shifts and Bitwise Operations Are Used Correctly, Even Though They Trigger MISRA Rule 10.1, 10.3, 10.4, 10.7, 12.2 Warnings. These are Suppressed Here For Clarity and Maintainability.

6.38.2.5 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.

6.38.2.6 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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

6.38.2.7 CONSOLELOG_GetFlushCompleted()

```
bool CONSOLELOG_GetFlushCompleted (
            void )
```

Returns If Flush Was Completed or Not.

Returns

If Recording Is Ongoing That Return False.

6.38.2.8 CONSOLELOG_GetFreeSpaceMB()

```
uint32_t CONSOLELOG_GetFreeSpaceMB (
            void )
```

Gets Free Space on SD Card.

Returns

Returns Free Space on SD Card.

6.38.2.9 CONSOLELOG_GetTransferedBytes()

```
uint32_t CONSOLELOG_GetTransferedBytes (
            void )
```

Returns Currently Received Bytes Between LED Blinking.

Returns

uint32_t Received Bytes Between LED Blinking.

6.38.2.10 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.

6.38.2.11 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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

6.38.2.12 CONSOLELOG_ProccessConfigFile()

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

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

Parameters

in	*content*	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.

6.38.2.13 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.

6.38.2.14 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.

ADMA Error Status (ADMA_ERR_STATUS) 3 bit -> ADMA Descriptor Error 2 bit -> ADMA Length Mismatch Error 1-0 bit -> ADMA Error State (When ADMA Error Occurred) Field Indicates The State of The ADMA When An Error Has Occurred During An ADMA Data Transfer.

6.38.2.15 get_fattime()

```
DWORD get_fattime (
            void )
```

MISRA Deviation: Rule 10.8 Suppress: Conversion From Smaller Unsigned Integer Types to a Wider Unsigned Type. Justification: The Cast From 'Uint8_t' and 'Uint16_t' to 'Uint32_t' is Intentional and Safe in This Context. The Values are Combined Using Bitwise OR Into a FAT Time Stamp Format That Expects 32-Bit Result. All Input Ranges are Within Limits.

6.38.2.16 LP_FLEXCOMM3_IRQHandler()

```
void LP_FLEXCOMM3_IRQHandler (
            void )
```

LPUART3 IRQ Handler.

6.39 src/semihost_hardfault.c File Reference

Functions

- __attribute__ ((naked))

6.39.1 Function Documentation

6.39.1.1 __attribute__()

```
__attribute__ (
            (naked) )
```

6.40 src/task_switching.c File Reference

```
#include "mass_storage.h"
```

Functions

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

6.40.1 Function Documentation

6.40.1.1 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	*pDeviceHandle*	Pointer to USB Device Handle (e.g. Mass Storage Handle).

Returns

kUSB_DeviceNotifyAttach if USB Is Attached Otherwise Returns kUSB_DeviceNotifyDetach.

6.41 src/temperature.c File Reference

```
#include <led.h>
#include "fsl_debug_console.h"
#include "fsl_ctimer.h"
#include "temperature.h"
#include "defs.h"
```

Macros

- #define LPI2C_TX_DMA_CHANNEL 0UL

  I2C DMA Channel For Transmission.
- #define LPI2C_RX_DMA_CHANNEL 1UL

  I2C DMA Channel For Reception.
- #define LPI2C_TX_CHANNEL (int32_t)kDma1RequestMuxLpFlexcomm5Tx

  Connection Between DMA Channel 1 and LP_FLEXCOMM5 Tx.
- #define LPI2C_RX_EDMA_CHANNEL (int32_t)kDma1RequestMuxLpFlexcomm5Rx

  Connection Between DMA Channel 1 and LP_FLEXCOMM5 Rx.
- #define I2C_MASTER_I2C5 ((LPI2C_Type ∗)LPI2C5_BASE)

  Points To I2C Peripheral Unit (Specifically LPI2C5 Instance).
- #define P3T1755_ADDR_7BIT 0x48U

  I2C Address of P3T1755 Temperature Sensor.
- #define I2C_BAUDRATE 100000U

  Desired Baud Rate For I2C.
- #define PRINT_REG_OUTPUT true

  Enables Printout of P3T1755 Register.
- #define BUFF_SIZE 2

  Receive Buffer Size.
- #define REGISTER_TEMPERATURE 0x00

  Temperature Register Address of P3T1755.
- #define REGISTER_CONFIG 0x01

  Configuration Register Address of P3T1755.
- #define REGISTER_THVST 0x02
- #define REGISTER_TOS 0x03

Functions

- AT_NONCACHEABLE_SECTION (static uint8_t g_aRxBuff_I2C5[BUFF_SIZE])

  Reception Buffer.
- AT_NONCACHEABLE_SECTION (static lpi2c_master_edma_handle_t g_EdmaHandle_I2C5)

  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)
- void CTIMER0_IRQHandler (void)
- 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.

Variables

- static edma_handle_t g_EdmaTxHandle_I2C5

  Tx eDMA Handle.
- static edma_handle_t g_EdmaRxHandle_I2C5

  Rx eDMA Handle.
- static volatile bool g_bCompletionFlag_I2C5 = false

  Flag Indicating Whether The Transfer Has Finished.
- static lpi2c_master_transfer_t g_Xfer_I2C5 = {0}

6.41.1 Macro Definition Documentation

6.41.1.1 BUFF_SIZE

```
#define BUFF_SIZE 2
```

Receive Buffer Size.

6.41.1.2 I2C_BAUDRATE

```
#define I2C_BAUDRATE 100000U
```

Desired Baud Rate For I2C.

Frequency - 100kHz.

6.41.1.3 I2C_MASTER_I2C5

```
#define I2C_MASTER_I2C5 ((LPI2C_Type *)LPI2C5_BASE)
```

Points To I2C Peripheral Unit (Specifically LPI2C5 Instance).

6.41.1.4 LPI2C_RX_DMA_CHANNEL

```
#define LPI2C_RX_DMA_CHANNEL 1UL
```

I2C DMA Channel For Reception.

6.41.1.5 LPI2C_RX_EDMA_CHANNEL

```
#define LPI2C_RX_EDMA_CHANNEL (int32_t)kDma1RequestMuxLpFlexcomm5Rx
```

Connection Between DMA Channel 1 and LP_FLEXCOMM5 Rx.

6.41.1.6 LPI2C_TX_CHANNEL

```
#define LPI2C_TX_CHANNEL (int32_t)kDma1RequestMuxLpFlexcomm5Tx
```

Connection Between DMA Channel 1 and LP_FLEXCOMM5 Tx.

6.41.1.7 LPI2C_TX_DMA_CHANNEL

```
#define LPI2C_TX_DMA_CHANNEL 0UL
```

I2C DMA Channel For Transmission.

6.41.1.8 P3T1755_ADDR_7BIT

```
#define P3T1755_ADDR_7BIT 0x48U
```

I2C Address of P3T1755 Temperature Sensor.

6.41.1.9 PRINT_REG_OUTPUT

```
#define PRINT_REG_OUTPUT true
```

Enables Printout of P3T1755 Register.

6.41.1.10 REGISTER_CONFIG

```
#define REGISTER_CONFIG 0x01
```

Configuration Register Address of P3T1755.

6.41.1.11 REGISTER_TEMPERATURE

```
#define REGISTER_TEMPERATURE 0x00
```

Temperature Register Address of P3T1755.

6.41.1.12 REGISTER_THVST

```
#define REGISTER_THVST 0x02
```

6.41.1.13 REGISTER_TOS

```
#define REGISTER_TOS 0x03
```

6.41.2 Function Documentation

6.41.2.1 AT_NONCACHEABLE_SECTION() [1/2]

```
AT_NONCACHEABLE_SECTION (
            static lpi2c_master_edma_handle_t g_EdmaHandle_I2C5)
```

eDMA Driver Handle Used For Non-Blocking DMA Transfer.

6.41.2.2 AT_NONCACHEABLE_SECTION() [2/2]

```
AT_NONCACHEABLE_SECTION (
            static uint8_t g_aRxBuff_I2C5[BUFF_SIZE])
```

Reception Buffer.

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

6.41.2.3 CTIMER0_IRQHandler()

```
void CTIMER0_IRQHandler (
            void )
```

6.41.2.4 lpi2c_callback()

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

6.41.2.5 Read()

```
uint16_t Read (
            uint8_t regAddress)
```

I2C Read Function.

Parameters

in	*regAddress*	Address of The Register From Which The Reading Will Take Place.

Return values

Returns	The Read Value From The Registry.

6.41.2.6 TMP_GetTemperature()

```
float TMP_GetTemperature (
            void )
```

Gets Temperature From On-Board P3T1755 Temperature Sensor.

Return values

Returns	Temperature As Float Number.

6.41.2.7 TMP_Init()

```
uint8_t TMP_Init (
            void )
```

Initialize On-Board P3T1755 Temperature Sensor.

6.41.2.8 Write()

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

I2C Write Function.

Parameters

in	*regAddress*	Address of The Register To Be Written To.
in	*val*	Array of Values To Be Written To The Register.

Return values

If	The Write Succeeds, Returns 0.

6.41.3 Variable Documentation

6.41.3.1 g_bCompletionFlag_I2C5

```
volatile bool g_bCompletionFlag_I2C5 = false  [static]
```

Flag Indicating Whether The Transfer Has Finished.

6.41.3.2 g_EdmaRxHandle_I2C5

```
edma_handle_t g_EdmaRxHandle_I2C5  [static]
```

Rx eDMA Handle.

6.41.3.3 g_EdmaTxHandle_I2C5

```
edma_handle_t g_EdmaTxHandle_I2C5  [static]
```

Tx eDMA Handle.

6.41.3.4 g_Xfer_I2C5

```
lpi2c_master_transfer_t g_Xfer_I2C5 = {0}  [static]
```

6.42 src/time.c File Reference

```
#include "fsl_irtc.h"
#include "rtc_ds3231.h"
#include "fsl_debug_console.h"
#include "pin_mux.h"
#include "time.h"
```

Macros

- #define INTERNAL_RTC_BASE_YEAR 2112

  Base Year of IRTC.
- #define EXTERNAL_RTC_TIME_OFSET 2000

  The Year Range Is Between 0 and 99.

Functions

- error_t TIME_InitIRTC (void)

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

6.42.1 Macro Definition Documentation

6.42.1.1 EXTERNAL_RTC_TIME_OFSET

```
#define EXTERNAL_RTC_TIME_OFSET 2000
```

The Year Range Is Between 0 and 99.

6.42.1.2 INTERNAL_RTC_BASE_YEAR

```
#define INTERNAL_RTC_BASE_YEAR 2112
```

Base Year of IRTC.

6.42.2 Function Documentation

6.42.2.1 TIME_InitIRTC()

```
error_t TIME_InitIRTC (
            void )
```

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

6.43 src/uart.c File Reference

```
#include "uart.h"
#include "defs.h"
#include "parser.h"
```

Macros

- #define BUFFER_SIZE (200)

Functions

- void UART_Print (uint8_t ch)

  Prints Character on The Terminal.
- void UART_Init (uint32_t baudrate)

  Initializes LPUART For Recording.
- 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.

6.43.1 Macro Definition Documentation

6.43.1.1 BUFFER_SIZE

```
#define BUFFER_SIZE (200)
```

6.43.2 Function Documentation

6.43.2.1 UART_Deinit()

```
void UART_Deinit (
            void )
```

De-Initialize LPUART.

The Pins Should Be De-Initialized After This Function.

6.43.2.2 UART_Disable()

```
void UART_Disable (
            void )
```

Disables Interrupt For Application LPUART.

6.43.2.3 UART_Enable()

```
void UART_Enable (
            void )
```

Enables Interrupt For Application LPUART.

MISRA Deviation Note: Rule 10.3: Cannot Assign 'enum' to a Different Essential Type Such As 'unsigned32'. [Required] Rule 11.4: Conversion Between Object Pointer Type and Integer Type. [Required] Justification: This Code Follows The Usage Pattern Provided By The NXP SDK.

6.43.2.4 UART_Init()

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

Initializes LPUART For Recording.

MISRA Deviation: Rule 11.4 [Required] Suppress: Conversion Between Object Pointer Type 'LPUART_Type ∗' and Integer Type 'Unsigned Int'. Justification: LPUART3 Is a Hardware Peripheral Base Address Defined In The NXP SDK. This Code Follows The Usage Pattern Provided By The NXP SDK.

6.43.2.5 UART_Print()

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

Prints Character on The Terminal.

Parameters

in	*ch*	Character in uint8_t.

Index