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 DS3231 Real-Time Clock Driver

Real-Time Circuit Related Functions.

Functions

- uint8_t RTC_Init (void)

  Initialize The RTC DS3231.
- void RTC_Deinit (void)

  De-Initialize The RTC DS3231.
- static uint8_t RTC_ConvertToBCD (uint8_t dec)

  Converts Numbers From Decimal Base To BCD Base.
- static uint8_t RTC_ConvertToDEC (uint8_t bcd)

  Converts Numbers From BCD Base To Decimal Base.
- uint8_t RTC_GetState (void)

  This Function Checks If The Oscillator Is Still Running.
- void RTC_SetOscState (RTC_osc_state_t state)

  Sets The Oscillator Stop Flag (OSF).
- uint8_t RTC_Write (uint8_t regAddress, uint8_t val)

  Writes Value Into RTC Registers.
- uint8_t RTC_Read (uint8_t regAddress)

  Reads Value From RTC Register.
- void RTC_SetTime (RTC_time_t ∗pTime)

  Sets Time.
- void RTC_GetTime (RTC_time_t ∗pTime)

  Gets Time.
- void RTC_SetDate (RTC_date_t ∗pDate)

  Sets Date.
- void RTC_GetDate (RTC_date_t ∗pDate)

  Gets Date.
- void RTC_SetInterruptMode (RTC_interrupt_mode_t mode)

  Sets The INTCN Bit in Control Register.
- void RTC_CtrlAlarm1 (uint8_t enable)

  Enables/Disables The Alarm 1.

- void RTC_ClearFlagAlarm1 (void)

  Clears The A1F Flag in Control Register.
- void RTC_CtrlAlarm2 (uint8_t enable)

  Enables/Disables The Alarm 2.
- void RTC_ClearFlagAlarm2 (void)

  Clears The A1F Flag in Control Register.
- void RTC_SetTimeDefault (RTC_time_t ∗pTime)

  Sets Time To Default.
- void RTC_SetDateDefault (RTC_date_t ∗pDate)

  Sets Date To Default.

4.1.1 Detailed Description

Real-Time Circuit Related Functions.

4.1.2 Function Documentation

4.1.2.1 RTC_ClearFlagAlarm1()

```
void RTC_ClearFlagAlarm1 (
             void )
```

Clears The A1F Flag in Control Register.

4.1.2.2 RTC_ClearFlagAlarm2()

```
void RTC_ClearFlagAlarm2 (
             void )
```

Clears The A1F Flag in Control Register.

4.1.2.3 RTC_ConvertToBCD()

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

Converts Numbers From Decimal Base To BCD Base.

Parameters

dec	Decimal Number.

Returns

Number in Binary Coded Decimal.

4.1.2.4 RTC_ConvertToDEC()

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

Converts Numbers From BCD Base To Decimal Base.

Parameters

bcd	Binary-Coded Decimal Number.

Returns

Number in Decimal Base.

4.1.2.5 RTC_CtrlAlarm1()

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

Enables/Disables The Alarm 1.

Parameters

enable	Specifies If The Alarm Will Be Enabled or Not.

4.1.2.6 RTC_CtrlAlarm2()

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

Enables/Disables The Alarm 2.

Parameters

enable	Specifies If The Alarm Will Be Enabled or Not.

4.1.2.7 RTC_Deinit()

```
void RTC_Deinit (
            void )
```

De-Initialize The RTC DS3231.

Pins Should Also Be De-Initialised Lately.

Parameters

void	

4.1.2.8 RTC_GetDate()

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

Gets Date.

Parameters

pDate	Pointer To Date Structure.

4.1.2.9 RTC_GetState()

```
uint8_t RTC_GetState (
            void )
```

This Function Checks If The Oscillator Is Still Running.

Returns

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

4.1.2.10 RTC_GetTime()

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

Gets Time.

Parameters

pTime	Pointer To Time Structure.

4.1.2.11 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;

4.1.2.12 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.

4.1.2.13 RTC_SetDate()

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

Sets Date.

Parameters

pDate	Pointer To Date Structure.

4.1.2.14 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.

4.1.2.15 RTC_SetInterruptMode()

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

Sets The INTCN Bit in Control Register.

Parameters

mode	Interrupt Mode.

4.1.2.16 RTC_SetOscState()

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

Sets The Oscillator Stop Flag (OSF).

Parameters

state	State of The Flag.

4.1.2.17 RTC_SetTime()

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

Sets Time.

Parameters

pTime	Pointer To Time Structure.

4.1.2.18 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.

4.1.2.19 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.

```
/*
 *      Author:         Tomas Dolak
 *      File Name:      rtc_DS3231.h
 *      Description:    Header File to RTC DS3231 Driver.
 *      Created on:     Aug 7, 2024
 *
 *      @author         Tomas Dolak
 *      @brief          Header File to RTC DS3231 Driver.
 *      @filename       rtc_DS3231.h
 */

#ifndef RTC_DS3231_H_
#define RTC_DS3231_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include <stdint.h>

#include "defs.h"

#include "fsl_lpi2c.h"
#include "fsl_lpi2c_edma.h"
#include "fsl_edma.h"
/*******************************************************************************
 * Definitions
 ******************************************************************************/

#define E_FAULT                         1

#define APP_SUCCESS                     0


#define LPI2C_TX_DMA_CHANNEL            0U

#define LPI2C_RX_DMA_CHANNEL            1U

#define LPI2C_TX_CHANNEL                kDma0RequestMuxLpFlexcomm2Tx

#define LPI2C_RX_EDMA_CHANNEL           kDma0RequestMuxLpFlexcomm2Rx

#define I2C_MASTER                      ((LPI2C_Type *)LPI2C2_BASE)

#define DS3231_A1IE                     (0x00u)
#define DS3231_A2IE                     (0x1u)
#define DS3231_INTCN                    (0x2u)
#define DS3231_A1F                      (0x0u)
#define DS3231_A2F                      (0x1u)
```

```

#define ALARM_DISABLED              (0x0u)
#define OSC_STOPPED                 (0x00u)

#define DS3231_ADDR_SEC             (0x00U)
#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)

#define DS3231_REG_CTRL             (0x0Eu)
#define DS3231_ADDR_I2C             (0x68U) //<! The Slave Address Byte Contains In 7-bit: 1101000

#define SUNDAY                      (0x1)
#define MONDAY                      (0x2)
#define TUESDAY                     (0x3)
#define WEDNESDAY                   (0x4)
#define THURSDAY                    (0x5)
#define FRIDAY                      (0x6)
#define SATURDAY                    (0x7)

#define TIM_CYCLE_12H               (0x0)
#define TIM_CYCLE_12H_AM            (0x0)
#define TIM_CYCLE_12H_PM            (0x1)
#define TIM_CYCLE_24H               (0x2)




#define I2C_BAUDRATE                100000U

/*******************************************************************************
 * Structures
 *******************************************************************************/
typedef struct
{
    uint8_t format;
    uint8_t sec;
    uint8_t min;
    uint8_t hrs;

} RTC_time_t;

typedef struct
{
    uint8_t date;
    uint8_t day;
    uint8_t month;
    uint8_t year;

} RTC_date_t;

typedef enum
{
    OSC_OK = 0,
    OSC_INTERRUPTED

} RTC_osc_state_t;

/*
 * @brief An Enum For Interrupt Mode.
 */
typedef enum
{
    SQUARE_WAVE_INTERRUPT = 0,
    ALARM_INTERRUPT

}RTC_interrupt_mode_t;
/*******************************************************************************
 * Prototypes
 *******************************************************************************/


uint8_t RTC_Init(void);

void RTC_Deinit(void);

static uint8_t RTC_ConvertToBCD(uint8_t dec);

static uint8_t RTC_ConvertToDEC(uint8_t bcd);

```

```
uint8_t RTC_GetState(void);

void RTC_SetOscState(RTC_osc_state_t state);

uint8_t RTC_Write(uint8_t regAddress, uint8_t val);

uint8_t RTC_Read(uint8_t regAddress);


void RTC_SetTime(RTC_time_t *pTime);

void RTC_GetTime(RTC_time_t *pTime);

void RTC_SetDate(RTC_date_t *pDate);

void RTC_GetDate(RTC_date_t *pDate);

void RTC_SetInterruptMode(RTC_interrupt_mode_t mode);

void RTC_CtrlAlarm1(uint8_t enable);

void RTC_ClearFlagAlarm1(void);

void RTC_CtrlAlarm2(uint8_t enable);

void RTC_ClearFlagAlarm2(void);

void RTC_SetTimeDefault(RTC_time_t *pTime);

void RTC_SetDateDefault(RTC_date_t *pDate);
 // end of RTC group

#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.

```
/******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      init.c
 *  Author:         Tomas Dolak
 *  Date:           31.03.2025
 *  Description:    Implements Datalogger Application.
 *
 * ****************************/

/******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           init.c
 *  @author         Tomas Dolak
 *  @date           31.03.2025
 *  @brief          Implements Datalogger Application.
 * ****************************/

#ifndef APP_INIT_H_
#define APP_INIT_H_
/*******************************************************************************
 * Includes
 ******************************************************************************/
/* NXP Board Specific */
#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"

/* Application Includes */
#include "led.h"                // Control LEDs Module
#include "uart.h"              // UART Module
#include "time.h"              // Time Module
#include "error.h"                // Error Handling
#include "temperature.h"        // Temperature Measurement Module
#include "pwrloss_det.h"        // Power Loss Detection Module

/*******************************************************************************
 * Global Definitions
 ******************************************************************************/

/*******************************************************************************
 * Global Structures
 ******************************************************************************/

/*******************************************************************************
 * Function Prototypes
 ******************************************************************************/

void APP_InitBoard(void);


#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 ∗∗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

- TaskHandle_t g_xMscTaskHandle
- TaskHandle_t g_xRecordTaskHandle
- SemaphoreHandle_t g_xSemRecord
- SemaphoreHandle_t g_xSemMassStorage

6.6.1 Macro Definition Documentation

6.6.1.1 TASK_PRIO

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

6.6.2 Function Documentation

6.6.2.1 msc_task()

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

Task Responsible For Mass Storage Functionality in Device Mode.

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

Parameters

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.

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.

6.6.3 Variable Documentation

6.6.3.1 g_xMscTaskHandle

```
TaskHandle_t g_xMscTaskHandle  [extern]
```

6.6.3.2 g_xRecordTaskHandle

```
TaskHandle_t g_xRecordTaskHandle  [extern]
```

6.6.3.3 g_xSemMassStorage

```
SemaphoreHandle_t g_xSemMassStorage  [extern]
```

6.6.3.4 g_xSemRecord

```
SemaphoreHandle_t g_xSemRecord  [extern]
```

6.7 app_tasks.h

Go to the documentation of this file.

```
/******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      app_tasks.c
 *  Author:         Tomas Dolak
 *  Date:           14.09.2024
 *  Description:    Header File For Implementation of FreeRTOS Task.
 *
 * ***************************/

/******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           app_tasks.c
 *  @author         Tomas Dolak
 *  @date           14.09.2024
 *  @brief          Header File For Implementation of FreeRTOS Task.
 * ***************************/

#ifndef APP_TASKS_H_
#define APP_TASKS_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/

/* FreeRTOS Include */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "timers.h"

/* Mass Storage Includes */
#include <disk.h>
/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define TASK_PRIO       (configMAX_PRIORITIES - 1) //<! Task Priorities.

/*******************************************************************************
 * Global Variables
 ******************************************************************************/

/* Task Handles     */
extern TaskHandle_t g_xMscTaskHandle;

extern TaskHandle_t g_xRecordTaskHandle;

/* Semaphores       */
extern SemaphoreHandle_t g_xSemRecord;

extern SemaphoreHandle_t g_xSemMassStorage;

/*******************************************************************************
 * Prototypes
 ******************************************************************************/
```

```


void msc_task(void *handle);

void record_task(void *handle);

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

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

#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 16UL

    Constant 5 Tau, When Back-Up Power Capacitor is Charged To 99%.
- #define PWRLOSS_DET_ACTIVE_IN_TIME TAU5

    Time Interval When Power Loss Detection Became Active.
- #define UART_RECEIVE_PRIO (6U)

    Priority of LP_FLEXCOMM Interrupt (UART) For Rx Of Recorded Data.
- #define PWRLOSS_DET_PRIO (5U)

    Priority of Comparator Interrupt For Power Loss Detection.
- #define PWRLOSS_TIMER_PRIO (7U)

    Priority of Comparator Interrupt For Power Loss Detection.

- #define TEMPERATURE_MEAS_ENABLED (false)

  Enables/Disables Temperature Recording.
- #define CONTROL_LED_ENABLED (true)

  Enables/Disables Signaling By LEDs.
- #define DEFAULT_MAX_FILESIZE 8192

  Default Maximal File Size If The Configuration File Could Not Be Read Properly.
- #define CONFIG_FILE "config"

  Configuration File.
- #define DEFAULT_BAUDRATE 230400UL

  Default Baud Rate If The Configuration File Could Not Be Read Properly.
- #define DEFAULT_DATA_BITS kLPUART_EightDataBits

  Default Number of Data Bits If The Configuration File Could Not Be Read Properly.
- #define DEFAULT_STOP_BITS kLPUART_OneStopBit

  Default Number of Stop Bits If The Configuration File Could Not Be Read Properly.
- #define DEFAULT_PARITY kLPUART_ParityDisabled

  Default UART Parity If The Configuration File Could Not Be Read Properly.
- #define DEFAULT_FREE_SPACE 50UL

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

6.8.1 Macro Definition Documentation

6.8.1.1 CONFIG_FILE

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

Configuration File.

6.8.1.2 CONTROL_LED_ENABLED

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

Enables/Disables Signaling By LEDs.

6.8.1.3 DEBUG_ENABLED

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

Enables/Disables Debug Mode.

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

6.8.1.4 DEFAULT_BAUDRATE

```
#define DEFAULT_BAUDRATE 230400UL
```

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

6.8.1.5 DEFAULT_DATA_BITS

```
#define DEFAULT_DATA_BITS kLPUART_EightDataBits
```

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

6.8.1.6 DEFAULT_FREE_SPACE

```
#define DEFAULT_FREE_SPACE 50UL
```

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

In Megabytes.

6.8.1.7 DEFAULT_MAX_FILESIZE

```
#define DEFAULT_MAX_FILESIZE 8192
```

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

6.8.1.8 DEFAULT_PARITY

```
#define DEFAULT_PARITY kLPUART_ParityDisabled
```

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

6.8.1.9 DEFAULT_STOP_BITS

```
#define DEFAULT_STOP_BITS kLPUART_OneStopBit
```

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

6.8.1.10 INFO_ENABLED

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

Enables/Disables Debug Mode.

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

6.8.1.11 MSC_ENABLED

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

Enables/Disables Mass Storage Functionality.

6.8.1.12 MSC_STACK_SIZE

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

Defines The Stack Size For Mass Storage Task.

6.8.1.13 NOT_IMPLEMENTED

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

6.8.1.14 PWRLOSS_DET_ACTIVE_IN_TIME

```
#define PWRLOSS_DET_ACTIVE_IN_TIME TAU5
```

Time Interval When Power Loss Detection Became Active.

In Seconds.

6.8.1.15 PWRLOSS_DET_PRIO

```
#define PWRLOSS_DET_PRIO (5U)
```

Priority of Comparator Interrupt For Power Loss Detection.

6.8.1.16 PWRLOSS_DETECTION_ENABLED

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

Enables/Disables Power Loss Detection.

6.8.1.17 PWRLOSS_TEST_GPIOS

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

Enables/Disables GPIO For Testing Power Loss Detection.

6.8.1.18 PWRLOSS_TIMER_PRIO

```
#define PWRLOSS_TIMER_PRIO (7U)
```

Priority of Comparator Interrupt For Power Loss Detection.

6.8.1.19 RECORD_STACK_SIZE

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

Defines The Stack Size For Recording Task.

6.8.1.20 TAU5

```
#define TAU5 16UL
```

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

In Seconds (It's Actualy 16s).

6.8.1.21 TEMPERATURE_MEAS_ENABLED

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

Enables/Disables Temperature Recording.

6.8.1.22 UART_FIFO_ENABLED

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

Enables/Disables HW FIFO Queue on Application LPUART.

6.8.1.23 UART_FIFO_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.

```
/*******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      defs.h
 *  Author:         Tomas Dolak
 *  Date:           22.09.2024
 *  Description:    Header File Providing Definitions To Set Up The Project.
 *
 * ****************************/

/*******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           defs.h
 *  @author         Tomas Dolak
 *  @date           22.09.2024
 *  @brief          Header File Providing Definitions To Set Up The Project.
 * ****************************/

#ifndef DEFS_H_
#define DEFS_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include <stdbool.h>


/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define NOT_IMPLEMENTED             #error "This Feature Is Not Implemented!"

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

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

#define MSC_ENABLED                 (true)

#define INFO_ENABLED                (false)

#define DEBUG_ENABLED               (false)

#define UART_FIFO_ENABLED           (true)

#define UART_FIFO_LENGHT            (4u)

#define UART_PRINT_ENABLED          (false)

#define PWRLOSS_DETECTION_ENABLED   (true)

#define PWRLOSS_TEST_GPIOS          (false)

#define TAU5                        16UL

#define PWRLOSS_DET_ACTIVE_IN_TIME  TAU5

#define UART_RECEIVE_PRIO           (6U)

#define PWRLOSS_DET_PRIO            (5U)

#define PWRLOSS_TIMER_PRIO          (7U)


#define TEMPERATURE_MEAS_ENABLED    (false)

#define CONTROL_LED_ENABLED         (true)


#define DEFAULT_MAX_FILESIZE        8192

#define CONFIG_FILE                 "config"

#define DEFAULT_BAUDRATE            230400UL

#define DEFAULT_DATA_BITS           kLPUART_EightDataBits

#define DEFAULT_STOP_BITS           kLPUART_OneStopBit

#define DEFAULT_PARITY              kLPUART_ParityDisabled

#define DEFAULT_FREE_SPACE          50UL

#endif /* DEFS_H_ */
```

6.10 include/error.h File Reference

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

Macros

- #define ERROR_NONE 0x0000u

  Return Code When a Successful Operation is Performed.
- #define ERROR_IRTC 0x0001u

  Error Related To Internal RTC Integrated In MCXN947 MCU.
- #define ERROR_RECORD 0x0002u

  Error Related To Data Logging.
- #define ERROR_OPEN 0x0005u

  Error Related To Opening File.
- #define ERROR_READ 0x0006u

  Error Related To Reading File.
- #define ERROR_CLOSE 0x0004u

  Error Related To Closing File.
- #define ERROR_FILESYSTEM 0x0007u

  Error Related To File System (e.g. Mount of File System, Setting Up File Meta-Data,...).
- #define ERROR_CONFIG 0x0008u

  Error Related To Reading and Parsing Configuration File Stored On SDHC Card (config.txt).
- #define ERROR_ADMA 0x0003u

  Error Related To Storing Data on SDHC Card Using ADMA (Advance DMA).
- #define ERROR_OUT_OF_CYCLE 0xFFFEu

  Out Of The Main Cycle Error.
- #define ERROR_UNKNOWN 0xFFFFu

  Occurrence Of an Unclassified Error.

Typedefs

- typedef uint16_t error_t

Functions

- void ERR_HandleError (void)

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

6.10.1 Macro Definition Documentation

6.10.1.1 ERROR_ADMA

```
#define ERROR_ADMA 0x0003u
```

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

6.10.1.2 ERROR_CLOSE

```
#define ERROR_CLOSE 0x0004u
```

Error Related To Closing File.

6.10.1.3 ERROR_CONFIG

```
#define ERROR_CONFIG 0x0008u
```

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

6.10.1.4 ERROR_FILESYSTEM

```
#define ERROR_FILESYSTEM 0x0007u
```

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

6.10.1.5 ERROR_IRTC

```
#define ERROR_IRTC 0x0001u
```

Error Related To Internal RTC Integrated In MCXN947 MCU.

6.10.1.6 ERROR_NONE

```
#define ERROR_NONE 0x0000u
```

Return Code When a Successful Operation is Performed.

6.10.1.7 ERROR_OPEN

```
#define ERROR_OPEN 0x0005u
```

Error Related To Opening File.

6.10.1.8 ERROR_OUT_OF_CYCLE

```
#define ERROR_OUT_OF_CYCLE 0xFFFEu
```

Out Of The Main Cycle Error.

6.10.1.9 ERROR_READ

```
#define ERROR_READ 0x0006u
```

Error Related To Reading File.

6.10.1.10 ERROR_RECORD

```
#define ERROR_RECORD 0x0002u
```

Error Related To Data Logging.

6.10.1.11 ERROR_UNKNOWN

```
#define ERROR_UNKNOWN 0xFFFFu
```

Occurrence Of an Unclassified Error.

6.10.2 Typedef Documentation

6.10.2.1 error_t

```
typedef uint16_t error_t
```

6.10.3 Function Documentation

6.10.3.1 ERR_HandleError()

```
void ERR_HandleError (
             void )
```

Base Function For Error Handling.

6.10.3.2 ERR_Init()

```
void ERR_Init ()
```

6.10.3.3 ERR_SetState()

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

6.11 error.h

Go to the documentation of this file.

```
/******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      error.h
 *  Author:         Tomas Dolak
 *  Date:           22.01.2025
 *  Description:    Implements Error Handling Logic.
 *
 * ****************************/

/******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           error.h
 *  @author         Tomas Dolak
 *  @date           22.01.2025
 *  @brief          Implements Error Handling Logic.
 * ****************************/

#ifndef ERROR_H_
#define ERROR_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include <stdint.h>
/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define ERROR_NONE              0x0000u

#define ERROR_IRTC              0x0001u

#define ERROR_RECORD            0x0002u

#define ERROR_OPEN              0x0005u

#define ERROR_READ              0x0006u
#define ERROR_CLOSE             0x0004u

#define ERROR_FILESYSTEM        0x0007u

#define ERROR_CONFIG            0x0008u

#define ERROR_ADMA              0x0003u

#define ERROR_OUT_OF_CYCLE      0xFFFEu

#define ERROR_UNKNOWN           0xFFFFu
/*******************************************************************************
 * Structures
 ******************************************************************************/
typedef uint16_t error_t;

/*******************************************************************************
 * Prototypes
 ******************************************************************************/
void ERR_HandleError(void);

void ERR_Init();

void ERR_SetState(error_t err);


#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.

```
/*******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      leds.c
 *  Author:         Tomas Dolak
 *  Date:           06.02.2025
 *  Description:    Implements The Logic For LEDs Control.
 *
 * ****************************/

/*******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           leds.c
 *  @author         Tomas Dolak
 *  @date           06.02.2025
 *  @brief          Implements The Logic For LEDs Control.
 * ****************************/

#ifndef LED_H_
#define LED_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include "fsl_gpio.h"
#include "board.h"
/*******************************************************************************
 * Functions Macros
 ******************************************************************************/
#define ERROR_LED_PORT          GPIO0

#define ERROR_LED_PIN_RECORD    0x09


#define MEMORY_LOW_LED_PORT     GPIO0

#define MEMORY_LOW_LED_PIN      0x07


#define FLUSH_LED_PORT      GPIO0

#define RECORD_LED_PIN_FLUSH    0x0D
```

```

#define RECORD_LED_PORT         GPIO2

#define RECORD_LED_PIN          0x0B

#define BACKUP_POWER_LED_PORT   GPIO4

#define BACKUP_POWER_LED_PIN    0x11
/******************************************************************************
 * Functions Definitions
 ******************************************************************************/

void LED_SetHigh(GPIO_Type *port_base, uint32_t pin);

void LED_SetLow(GPIO_Type *port_base, uint32_t pin);

void LED_SignalReady(void);

void LED_SignalRecording(void);

void LED_SignalRecordingStop(void);

void LED_SignalBackUpPowerAvailable(void);

void LED_SignalLowMemory(void);

void LED_SignalError(void);

void LED_SignalFlush(void);

void LED_ClearSignalFlush(void);

#endif /* LED_H_ */
```

6.14 include/mass_storage.h File Reference

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

Functions

- void MSC_DeviceMscApp (void)

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

  Handles Mass Storage Application.

6.14.1 Function Documentation

6.14.1.1 MSC_DeviceMscApp()

```
void MSC_DeviceMscApp (
            void )
```

Process Extension to Mass Storage.

6.14.1.2 MSC_DeviceMscAppTask()

```
void MSC_DeviceMscAppTask (
            void )
```

Handles Mass Storage Application.

Communication and Data Transport Is Handled By USB1_HS ISR.

6.15 mass_storage.h

Go to the documentation of this file.

```
/*****************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      mass_storage.h
 *  Author:         Tomas Dolak
 *  Date:           06.02.2025
 *  Description:    Header File For USB Mass Storage.
 *
 * ***************************/

/*****************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           mass_storage.h
 *  @author         Tomas Dolak
 *  @date           06.02.2025
 *  @brief          Header File For USB Mass Storage.
 * ***************************/

#ifndef MASS_STORAGE_H_
#define MASS_STORAGE_H_


/*******************************************************************************
 * Includes
 ******************************************************************************/
#include "disk.h"

/*******************************************************************************
 * Global Variables
 ******************************************************************************/

/*******************************************************************************
 * Function Declarations
 ******************************************************************************/

void MSC_DeviceMscApp(void);

void MSC_DeviceMscAppTask(void);

#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.

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.

```
/******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      parser.h
 *  Author:         Tomas Dolak
 *  Date:           16.04.2025
 *  Description:    Header File Providing Definitions For Configuration File Parser.
 *
 * ****************************/

/******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           parser.h
 *  @author         Tomas Dolak
 *  @date           16.04.2025
 *  @brief          Header File Providing Definitions For Configuration File Parser.
 * ****************************/
#ifndef PARSER_H_
#define PARSER_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#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"
/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*******************************************************************************
 * Structures
 ******************************************************************************/
typedef enum
{
    WCT_UNKOWN = 0,
    WCT_AUTOS1,
    WCT_AUTOS2

} REC_version_t;

typedef struct
{
    /* Recorded NXP Device  */
    REC_version_t  version;

    /* UART Setup */
    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;

} REC_config_t;
/*******************************************************************************
 * Prototypes
 ***************************************************************************/
REC_config_t PARSER_GetConfig(void);

REC_version_t PARSER_GetVersion(void);

uint32_t PARSER_GetBaudrate(void);

uint32_t PARSER_GetFileSize(void);

lpuart_data_bits_t PARSER_GetDataBits(void);

lpuart_parity_mode_t PARSER_GetParity(void);

lpuart_stop_bit_count_t PARSER_GetStopBits(void);

uint32_t PARSER_GetFreeSpaceLimitMB(void);

uint32_t PARSER_GetMaxBytes(void);

void PARSER_ClearConfig(void);

error_t PARSER_ParseBaudrate(const char *chContent);

error_t PARSER_ParseFileSize(const char *chContent);

error_t PARSER_ParseParity(const char *chContent);

error_t PARSER_ParseStopBits(const char *chContent);

error_t PARSER_ParseDataBits(const char *chContent);

error_t PARSER_ParseFreeSpace(const char *chContent);

#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.

```
/*******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      pwrloss_det.h
 *  Author:         Tomas Dolak
 *  Date:           01.02.2024
 *  Description:    Implements The Logic Of Power Loss Detection.
 *
 * ***************************/

/*******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           pwrloss_det.h
 *  @author         Tomas Dolak
 *  @date           01.02.2024
 *  @brief          Implements The Logic Of Power Loss Detection.
 * ***************************/

#ifndef PWRLOSS_DET_H_
#define PWRLOSS_DET_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/

#include "fsl_lpcmp.h"
#include "fsl_spc.h"

#include "led.h"
#include "defs.h"


#include "fsl_ctimer.h"
/*******************************************************************************
 * Global Definitions
 ******************************************************************************/

#define LPCMP_BASE                      CMP1

#define LPCMP_USER_CHANNEL              0x02

#define LPCMP_DAC_CHANNEL               0x07

#define LPCMP_IRQ_ID                    HSCMP1_IRQn

#define SPC_BASE                        SPC0

#define CTIMER                          CTIMER4

#define CTIMER_MAT0_OUT                 kCTIMER_Match_0

#define CTIMER_EMT0_OUT                 (1u « kCTIMER_Match_0)

#define CTIMER_CLK_FREQ                 CLOCK_GetCTimerClkFreq(4U)

/*******************************************************************************
 * Prototypes
 ******************************************************************************/
void PWRLOSS_DetectionInit(void);


#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.

6.20.2.4 CONSOLELOG_CreateFile()

```
error_t CONSOLELOG_CreateFile (
             void )
```

Creates File Based Actual Date and Counter Value.

Returns

Returns Zero If File Creation Succeeded.

6.20.2.5 CONSOLELOG_Deinit()

```
error_t CONSOLELOG_Deinit (
             void )
```

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

Returns

error_t Returns 0 on Success, Otherwise E_FAULT.

6.20.2.6 CONSOLELOG_Flush()

```
error_t CONSOLELOG_Flush (
             void )
```

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

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

Returns

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

6.20.2.7 CONSOLELOG_GetFlushCompleted()

```
bool CONSOLELOG_GetFlushCompleted (
             void )
```

Returns If Flush Was Completed or Not.

Returns

If Recording Is Ongoing That Return False.

6.20.2.8 CONSOLELOG_GetFreeSpaceMB()

```
uint32_t CONSOLELOG_GetFreeSpaceMB (
             void )
```

Gets Free Space on SD Card.

Returns

Returns Free Space on SD Card.

6.20.2.9 CONSOLELOG_GetMaxBytes()

```
uint32_t CONSOLELOG_GetMaxBytes (
            void )
```

Maximal Received Bytes Between LED Blinking.

6.20.2.10 CONSOLELOG_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.

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.

6.21 record.h

Go to the documentation of this file.

```
/*****************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      fatfs.h
 *  Author:         Tomas Dolak
 *  Date:           18.11.2024
 *  Description:    File Includes Operation For Recoding Mode.
 *
 * ****************************/

/*****************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           fatfs.h
 *  @author         Tomas Dolak
 *  @date           18.11.2024
 *  @brief          File Includes Operation For Recoding Mode.
 * ****************************/

```

```
#ifndef RECORD_H_
#define RECORD_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include <led.h>
#include <string.h>
#include "fsl_sd.h"
#include "ff.h"                    /*<! File System */
#include "ffconf.h"                /*<! File System Configuration */
#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"
/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define FLUSH_TIMEOUT_TICKS pdMS_TO_TICKS(3000)

/*******************************************************************************
 * Structures
 ******************************************************************************/

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

uint32_t CONSOLELOG_GetFreeSpaceMB(void);

error_t CONSOLELOG_CreateDirectory(void);

error_t CONSOLELOG_CreateFile(void);

uint32_t CONSOLELOG_GetTransferedBytes(void);

bool CONSOLELOG_GetFlushCompleted(void);

void CONSOLELOG_ClearTransferedBytes(void);

uint32_t CONSOLELOG_GetMaxBytes(void);

FRESULT CONSOLELOG_CheckFileSystem(void);

error_t CONSOLELOG_Init(void);

error_t CONSOLELOG_Recording(uint32_t file_size);

error_t CONSOLELOG_Flush(void);

error_t CONSOLELOG_PowerLossFlush(void);

error_t CONSOLELOG_Deinit(void);

error_t CONSOLELOG_ReadConfig(void);

error_t CONSOLELOG_ProccessConfigFile(const char *content);


#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.

6.23 task_switching.h

Go to the documentation of this file.

```
/*******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      task_switching.c
 *  Author:         Tomas Dolak
 *  Date:           14.09.2024
 *  Description:    Includes Implementation of Task Switching (Record / USB Mass Storage).
 *
 * ****************************/

/*******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           task_switching.c
 *  @author         Tomas Dolak
 *  @date           14.09.2024
 *  @brief          Includes Implementation of Task Switching (Record / USB Mass Storage).
 * ****************************/

#ifndef TASK_SWITCHING_H_
#define TASK_SWITCHING_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include "usb_device_dci.h"

/*******************************************************************************
 * Global Variables
 ******************************************************************************/
extern usb_msc_struct_t g_msc;
/*******************************************************************************
 * Prototypes
 ******************************************************************************/
usb_device_notification_t USB_State(usb_device_struct_t *pDeviceHandle);


#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.

```
/*******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      temperature.h
 *  Author:         Tomas Dolak
 *  Date:           11.02.2025
 *  Description:    Implements The Logic For Temperature Measurement.
 *
 * ****************************/

/*******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           temperature.h
 *  @author         Tomas Dolak
 *  @date           11.02.2025
 *  @brief          Implements The Logic For Temperature Measurement.
 * ****************************/

#ifndef TEMPERATURE_H_
#define TEMPERATURE_H_

/*******************************************************************************
 * Includes
 ******************************************************************************/

#include "error.h"

#include "fsl_lpi2c.h"
#include "fsl_lpi2c_edma.h"
#include "fsl_edma.h"
/*******************************************************************************
 * Global Definitions
 ******************************************************************************/


/*******************************************************************************
 * Prototypes
 ******************************************************************************/
uint8_t Write(uint8_t regAddress, uint8_t val[]);

uint16_t Read(uint8_t regAddress);

float TMP_GetTemperature(void);

uint8_t TMP_Init(void);

#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.

```
/*****************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      mainc.c
 *  Author:         Tomas Dolak
 *  Date:           07.08.2024
 *  Description:    Implements Datalogger Application.
 *
 * ****************************/

/*****************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           main.c
 *  @author         Tomas Dolak
 *  @date           07.08.2024
 *  @brief          Implements Datalogger Application.
 * ****************************/


#ifndef TIME_H_
#define TIME_H_

/******************************************************************************
 * Includes
 ******************************************************************************/
#include "fsl_irtc.h"
#include "rtc_ds3231.h"
#include "fsl_debug_console.h"
#include "error.h"

/******************************************************************************
 * Global Definitions
 ******************************************************************************/

#define LPI2C_DMA_BASEADDR  (DMA0)

/******************************************************************************
 * Prototypes
 ******************************************************************************/

error_t TIME_InitIRTC(void);

#endif /* TIME_H_ */
```

6.28 include/uart.h File Reference

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

Macros

- #define LPUART3_CLK_FREQ CLOCK_GetLPFlexCommClkFreq(3u)

  Frequency of LPUART7.

Functions

- void UART_Init (uint32_t baudrate)

  Initializes LPUART For Recording.
- void UART_Print (uint8_t ch)

  Prints Character on The Terminal.
- void UART_Enable (void)

  Enables Interrupt For Application LPUART.
- void UART_Disable (void)

  Disables Interrupt For Application LPUART.
- void UART_Deinit (void)

  De-Initialize LPUART.

6.28.1 Macro Definition Documentation

6.28.1.1 LPUART3_CLK_FREQ

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

Frequency of LPUART7.

6.28.2 Function Documentation

6.28.2.1 UART_Deinit()

```
void UART_Deinit (
             void )
```

De-Initialize LPUART.

The Pins Should Be De-Initialized After This Function.

6.28.2.2 UART_Disable()

```
void UART_Disable (
             void )
```

Disables Interrupt For Application LPUART.

6.28.2.3 UART_Enable()

```
void UART_Enable (
             void )
```

Enables Interrupt For Application LPUART.

6.28.2.4 UART_Init()

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

Initializes LPUART For Recording.

6.28.2.5 UART_Print()

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

Prints Character on The Terminal.

Parameters

in	*ch*	Character in uint8_t.

6.29 uart.h

Go to the documentation of this file.

```
/******************************
 *  Project:        NXP MCXN947 Datalogger
 *  File Name:      uart.h
 *  Author:         Tomas Dolak
 *  Date:           25.11.2024
 *  Description:    Header File For The UART Peripheral Support.
 *
 * ****************************/

/******************************
 *  @package        NXP MCXN947 Datalogger
 *  @file           uart.h
 *  @author         Tomas Dolak
 *  @date           25.11.2024
 *  @brief          Header File For The UART Peripheral Support.
 * ****************************/

#ifndef UART_H_
#define UART_H_
```

```

/*******************************************************************************
 * Includes
 ******************************************************************************/
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_lpuart.h"

#include "fsl_debug_console.h"
#include "fsl_clock.h"
/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define LPUART3_CLK_FREQ   CLOCK_GetLPFlexCommClkFreq(3u)


/*******************************************************************************
 * Declarations
 ******************************************************************************/

void UART_Init(uint32_t baudrate);

void UART_Print(uint8_t ch);

void UART_Enable(void);

void UART_Disable(void);

void UART_Deinit(void);

#endif /* UART_H_ */

```

Index