

My Project

Generated by Doxygen 1.13.2

1 Topic Index	1
1.1 Topics	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Topic Documentation	7
4.1 Task Management	7
4.1.1 Detailed Description	7
4.1.2 Variable Documentation	7
4.1.2.1 uxIdleTaskStack	7
4.1.2.2 uxTimerTaskStack	8
4.1.2.3 xIdleTaskTCB	8
4.1.2.4 xTimerTaskTCB	8
4.2 Buffers and Recording Management	8
4.2.1 Detailed Description	9
4.2.2 Function Documentation	9
4.2.2.1 SDK_ALIGN()	9
4.2.3 Variable Documentation	9
4.2.3.1 g_bBackDmaBufferReady	9
4.2.3.2 g_bFlushCompleted	9
4.2.3.3 g_fileObject	10
4.2.3.4 g_fileSystem	10
4.2.3.5 g_lastDataTick	10
4.2.3.6 g_pu8BackDmaBuffer	10
4.2.3.7 g_pu8FrontDmaBuffer	10
4.2.3.8 g_u16BackDmaBufferIdx	10
4.2.3.9 g_u32BytesTransferred	10
4.2.3.10 g_u32CurrentFileSize	11
4.2.3.11 g_u32FileCounter	11
4.2.3.12 g_u8CurrentDirectory	11
4.3 Management	11
4.3.1 Detailed Description	11
4.3.2 Function Documentation	12
4.3.2.1 SDK_ALIGN()	12
4.3.3 Variable Documentation	12
4.3.3.1 g_u32ReadIndex	12
4.3.3.2 g_u32WriteIndex	12
4.4 DS3231 Real-Time Clock Driver	12
4.4.1 Detailed Description	13

4.4.2 Function Documentation	13
4.4.2.1 RTC_ClearFlagAlarm1()	13
4.4.2.2 RTC_ClearFlagAlarm2()	13
4.4.2.3 RTC_ConvertToBCD()	13
4.4.2.4 RTC_ConvertToDEC()	14
4.4.2.5 RTC_CtrlAlarm1()	14
4.4.2.6 RTC_CtrlAlarm2()	14
4.4.2.7 RTC_Deinit()	14
4.4.2.8 RTC_GetDate()	15
4.4.2.9 RTC_GetState()	15
4.4.2.10 RTC_GetTime()	15
4.4.2.11 RTC_Init()	15
4.4.2.12 RTC_Read()	16
4.4.2.13 RTC_SetDate()	16
4.4.2.14 RTC_SetDateDefault()	16
4.4.2.15 RTC_SetInterruptMode()	16
4.4.2.16 RTC_SetOscState()	17
4.4.2.17 RTC_SetTime()	17
4.4.2.18 RTC_SetTimeDefault()	17
4.4.2.19 RTC_Write()	17
5 Class Documentation	19
5.1 REC_config_t Struct Reference	19
5.1.1 Detailed Description	19
5.1.2 Member Data Documentation	19
5.1.2.1 baudrate	19
5.1.2.2 data_bits	20
5.1.2.3 free_space_limit_mb	20
5.1.2.4 max_bytes	20
5.1.2.5 parity	20
5.1.2.6 size	20
5.1.2.7 stop_bits	20
5.1.2.8 version	20
5.2 RTC_date_t Struct Reference	21
5.2.1 Detailed Description	21
5.2.2 Member Data Documentation	21
5.2.2.1 date	21
5.2.2.2 day	21
5.2.2.3 month	21
5.2.2.4 year	21
5.3 RTC_time_t Struct Reference	22
5.3.1 Detailed Description	22

5.3.2 Member Data Documentation	22
5.3.2.1 format	22
5.3.2.2 hrs	22
5.3.2.3 min	22
5.3.2.4 sec	22
6 File Documentation	23
6.1 drivers/rtc_ds3231.c File Reference	23
6.1.1 Macro Definition Documentation	24
6.1.1.1 BYTE_1	24
6.1.1.2 BYTE_2	24
6.1.1.3 CLK_STATE	25
6.1.1.4 I2C_DATA_LENGTH	25
6.1.1.5 OSF_STATE	25
6.1.2 Function Documentation	25
6.1.2.1 AT_NONCACHEABLE_SECTION() [1/2]	25
6.1.2.2 AT_NONCACHEABLE_SECTION() [2/2]	25
6.1.2.3 lpi2c_callback()	25
6.1.2.4 RTC_ConvertToBCD()	25
6.1.2.5 RTC_ConvertToDEC()	26
6.1.3 Variable Documentation	26
6.1.3.1 gCompletionFlag	26
6.1.3.2 gEdmaRxHandle	26
6.1.3.3 gEdmaTxHandle	26
6.1.3.4 xfer	26
6.2 drivers/rtc_ds3231.h File Reference	26
6.2.1 Macro Definition Documentation	28
6.2.1.1 ALARM_DISABLED	28
6.2.1.2 APP_SUCCESS	28
6.2.1.3 DS3231_A1F	29
6.2.1.4 DS3231_A1IE	29
6.2.1.5 DS3231_A2F	29
6.2.1.6 DS3231_A2IE	29
6.2.1.7 DS3231_ADDR_CENT	29
6.2.1.8 DS3231_ADDR_DATE	29
6.2.1.9 DS3231_ADDR_DAY	29
6.2.1.10 DS3231_ADDR_HRS	29
6.2.1.11 DS3231_ADDR_I2C	29
6.2.1.12 DS3231_ADDR_MIN	29
6.2.1.13 DS3231_ADDR_MONTH	30
6.2.1.14 DS3231_ADDR_SEC	30
6.2.1.15 DS3231_ADDR_YEAR	30

6.2.1.16 DS3231_INTCN	30
6.2.1.17 DS3231_REG_CTRL	30
6.2.1.18 DS3231_REG_STATUS	30
6.2.1.19 E_FAULT	30
6.2.1.20 FRIDAY	30
6.2.1.21 I2C_BAUDRATE	30
6.2.1.22 I2C_MASTER	31
6.2.1.23 LPI2C_RX_DMA_CHANNEL	31
6.2.1.24 LPI2C_RX_EDMA_CHANNEL	31
6.2.1.25 LPI2C_TX_CHANNEL	31
6.2.1.26 LPI2C_TX_DMA_CHANNEL	31
6.2.1.27 MONDAY	31
6.2.1.28 OSC_STOPPED	31
6.2.1.29 SATURDAY	31
6.2.1.30 SUNDAY	32
6.2.1.31 THURSDAY	32
6.2.1.32 TIM_CYCLE_12H	32
6.2.1.33 TIM_CYCLE_12H_AM	32
6.2.1.34 TIM_CYCLE_12H_PM	32
6.2.1.35 TIM_CYCLE_24H	32
6.2.1.36 TUESDAY	32
6.2.1.37 WEDNESDAY	32
6.2.2 Enumeration Type Documentation	32
6.2.2.1 RTC_interrupt_mode_t	32
6.2.2.2 RTC_osc_state_t	32
6.3 rtc_ds3231.h	33
6.4 include/app_init.h File Reference	35
6.4.1 Function Documentation	36
6.4.1.1 APP_InitBoard()	36
6.5 app_init.h	36
6.6 include/app_tasks.h File Reference	37
6.6.1 Macro Definition Documentation	37
6.6.1.1 TASK_PRIO	37
6.6.2 Function Documentation	37
6.6.2.1 msc_task()	37
6.6.2.2 record_task()	38
6.6.2.3 vApplicationGetIdleTaskMemory()	38
6.6.2.4 vApplicationGetTimerTaskMemory()	38
6.6.3 Variable Documentation	39
6.6.3.1 g_xMscTaskHandle	39
6.6.3.2 g_xRecordTaskHandle	39
6.6.3.3 g_xSemMassStorage	39

6.6.3.4 g_xSemRecord	39
6.7 app_tasks.h	39
6.8 include/defs.h File Reference	40
6.8.1 Macro Definition Documentation	41
6.8.1.1 CONFIG_FILE	41
6.8.1.2 CONTROL_LED_ENABLED	41
6.8.1.3 DEBUG_ENABLED	42
6.8.1.4 DEFAULT_BAUDRATE	42
6.8.1.5 DEFAULT_DATA_BITS	42
6.8.1.6 DEFAULT_FREE_SPACE	42
6.8.1.7 DEFAULT_MAX_FILESIZE	42
6.8.1.8 DEFAULT_PARITY	42
6.8.1.9 DEFAULT_STOP_BITS	42
6.8.1.10 INFO_ENABLED	43
6.8.1.11 MSC_ENABLED	43
6.8.1.12 MSC_STACK_SIZE	43
6.8.1.13 NOT_IMPLEMENTED	43
6.8.1.14 PWRLOSS_DET_ACTIVE_IN_TIME	43
6.8.1.15 PWRLOSS_DET_PRIO	43
6.8.1.16 PWRLOSS_DETECTION_ENABLED	43
6.8.1.17 PWRLOSS_TEST_GPIOS	44
6.8.1.18 PWRLOSS_TIMER_PRIO	44
6.8.1.19 RECORD_STACK_SIZE	44
6.8.1.20 TAU	44
6.8.1.21 TEMPERATURE_MEAS_ENABLED	44
6.8.1.22 UART_FIFO_ENABLED	44
6.8.1.23 UART_FIFO_LENHT	44
6.8.1.24 UART_PRINT_ENABLED	45
6.8.1.25 UART_RECEIVE_PRIO	45
6.9 defs.h	45
6.10 include/error.h File Reference	46
6.10.1 Macro Definition Documentation	47
6.10.1.1 ERROR_ADMA	47
6.10.1.2 ERROR_CLOSE	47
6.10.1.3 ERROR_CONFIG	47
6.10.1.4 ERROR_FILESYSTEM	47
6.10.1.5 ERROR_IRTC	47
6.10.1.6 ERROR_NONE	47
6.10.1.7 ERROR_OPEN	48
6.10.1.8 ERROR_OUT_OF_CYCLE	48
6.10.1.9 ERROR_READ	48
6.10.1.10 ERROR_RECORD	48

6.10.1.11 ERROR_UNKNOWN	48
6.10.2 Typedef Documentation	48
6.10.2.1 error_t	48
6.10.3 Function Documentation	48
6.10.3.1 ERR_HandleError()	48
6.10.3.2 ERR_Init()	49
6.10.3.3 ERR_SetState()	49
6.11 error.h	49
6.12 include/led.h File Reference	50
6.12.1 Macro Definition Documentation	50
6.12.1.1 BACKUP_POWER_LED_PIN	50
6.12.1.2 BACKUP_POWER_LED_PORT	50
6.12.1.3 ERROR_LED_PIN_RECORD	50
6.12.1.4 ERROR_LED_PORT	51
6.12.1.5 FLUSH_LED_PORT	51
6.12.1.6 MEMORY_LOW_LED_PIN	51
6.12.1.7 MEMORY_LOW_LED_PORT	51
6.12.1.8 RECORD_LED_PIN	51
6.12.1.9 RECORD_LED_PIN_FLUSH	51
6.12.1.10 RECORD_LED_PORT	51
6.12.2 Function Documentation	51
6.12.2.1 LED_ClearSignalFlush()	51
6.12.2.2 LED_SetHigh()	51
6.12.2.3 LED_SetLow()	52
6.12.2.4 LED_SignalBackUpPowerAvailable()	52
6.12.2.5 LED_SignalError()	52
6.12.2.6 LED_SignalFlush()	52
6.12.2.7 LED_SignalLowMemory()	52
6.12.2.8 LED_SignalReady()	52
6.12.2.9 LED_SignalRecording()	52
6.12.2.10 LED_SignalRecordingStop()	52
6.13 led.h	53
6.14 include/mass_storage.h File Reference	54
6.14.1 Function Documentation	55
6.14.1.1 MSC_DeviceMscApp()	55
6.14.1.2 MSC_DeviceMscAppTask()	55
6.15 mass_storage.h	55
6.16 include/parser.h File Reference	56
6.16.1 Enumeration Type Documentation	57
6.16.1.1 REC_version_t	57
6.16.2 Function Documentation	57
6.16.2.1 PARSER_ClearConfig()	57

6.16.2.2 PARSER_GetBaudrate()	57
6.16.2.3 PARSER_GetConfig()	58
6.16.2.4 PARSER_GetDataBits()	58
6.16.2.5 PARSER_GetFileSize()	58
6.16.2.6 PARSER_GetFreeSpaceLimitMB()	58
6.16.2.7 PARSER_GetMaxBytes()	59
6.16.2.8 PARSER_GetParity()	59
6.16.2.9 PARSER_GetStopBits()	59
6.16.2.10 PARSER_GetVersion()	59
6.16.2.11 PARSER_ParseBaudrate()	59
6.16.2.12 PARSER_ParseDataBits()	60
6.16.2.13 PARSER_ParseFileSize()	60
6.16.2.14 PARSER_ParseFreeSpace()	60
6.16.2.15 PARSER_ParseParity()	61
6.16.2.16 PARSER_ParseStopBits()	61
6.17 parser.h	62
6.18 include/pwrloss_det.h File Reference	63
6.18.1 Macro Definition Documentation	63
6.18.1.1 CTIMER	63
6.18.1.2 CTIMER_CLK_FREQ	63
6.18.1.3 CTIMER_EMT0_OUT	64
6.18.1.4 CTIMER_MAT0_OUT	64
6.18.1.5 DEMO_LPCMP_BASE	64
6.18.1.6 DEMO_LPCMP_DAC_CHANNEL	64
6.18.1.7 DEMO_LPCMP_IRQ_HANDLER_FUNC	64
6.18.1.8 DEMO_LPCMP_IRQ_ID	64
6.18.1.9 DEMO_LPCMP_USER_CHANNEL	64
6.18.1.10 DEMO_SPC_BASE	64
6.18.1.11 DEMO_VREF_BASE	64
6.18.2 Function Documentation	64
6.18.2.1 PWRLOSS_DetectionInit()	64
6.19 pwrloss_det.h	65
6.20 include/record.h File Reference	65
6.20.1 Macro Definition Documentation	67
6.20.1.1 FLUSH_TIMEOUT_TICKS	67
6.20.2 Function Documentation	67
6.20.2.1 CONSOLELOG_CheckFileSystem()	67
6.20.2.2 CONSOLELOG_ClearTransferredBytes()	67
6.20.2.3 CONSOLELOG_CreateDirectory()	67
6.20.2.4 CONSOLELOG_CreateFile()	67
6.20.2.5 CONSOLELOG_Deinit()	68
6.20.2.6 CONSOLELOG_Flush()	68

6.20.2.7	CONSOLELOG_GetFlushCompleted()	68
6.20.2.8	CONSOLELOG_GetFreeSpaceMB()	68
6.20.2.9	CONSOLELOG_GetMaxBytes()	69
6.20.2.10	CONSOLELOG_GetTransferredBytes()	69
6.20.2.11	CONSOLELOG_Init()	69
6.20.2.12	CONSOLELOG_PowerLossFlush()	69
6.20.2.13	CONSOLELOG_ProccessConfigFile()	69
6.20.2.14	CONSOLELOG_ReadConfig()	70
6.20.2.15	CONSOLELOG_Recording()	70
6.21	record.h	71
6.22	include/task_switching.h File Reference	72
6.22.1	Function Documentation	72
6.22.1.1	USB_State()	72
6.22.2	Variable Documentation	72
6.22.2.1	g_msc	72
6.23	task_switching.h	73
6.24	include/temperature.h File Reference	73
6.24.1	Macro Definition Documentation	74
6.24.1.1	SENSOR_DYNAMIC_ADDR	74
6.24.1.2	SENSOR_STATIC_ADDR	74
6.24.2	Function Documentation	74
6.24.2.1	Read()	74
6.24.2.2	TMP_GetTemperature()	74
6.24.2.3	TMP_Init()	74
6.24.2.4	Write()	74
6.25	temperature.h	75
6.26	include/time.h File Reference	75
6.26.1	Macro Definition Documentation	76
6.26.1.1	LPI2C_DMA_BASEADDR	76
6.26.2	Function Documentation	76
6.26.2.1	TIME_InitIRTC()	76
6.27	time.h	76
6.28	include/uart.h File Reference	77
6.28.1	Macro Definition Documentation	77
6.28.1.1	LPUART3_CLK_FREQ	77
6.28.2	Function Documentation	78
6.28.2.1	UART_Deinit()	78
6.28.2.2	UART_Disable()	78
6.28.2.3	UART_Enable()	78
6.28.2.4	UART_Init()	78
6.28.2.5	UART_Print()	78
6.29	uart.h	79

6.30 source/app_init.c File Reference	79
6.30.1 Function Documentation	80
6.30.1.1 APP_InitBoard()	80
6.31 source/app_tasks.c File Reference	80
6.31.1 Function Documentation	81
6.31.1.1 msc_task()	81
6.31.1.2 record_task()	81
6.31.1.3 vApplicationGetIdleTaskMemory()	81
6.31.1.4 vApplicationGetTimerTaskMemory()	81
6.32 source/error.c File Reference	82
6.32.1 Function Documentation	82
6.32.1.1 ERR_HandleError()	82
6.32.1.2 ERR_Init()	82
6.32.1.3 ERR_SetState()	82
6.32.2 Variable Documentation	83
6.32.2.1 error	83
6.33 source/led.c File Reference	83
6.33.1 Function Documentation	83
6.33.1.1 LED_ClearSignalFlush()	83
6.33.1.2 LED_SetHigh()	83
6.33.1.3 LED_SetLow()	83
6.33.1.4 LED_SignalBackUpPowerAvailable()	84
6.33.1.5 LED_SignalError()	84
6.33.1.6 LED_SignalFlush()	84
6.33.1.7 LED_SignalLowMemory()	84
6.33.1.8 LED_SignalReady()	84
6.33.1.9 LED_SignalRecording()	84
6.33.1.10 LED_SignalRecordingStop()	84
6.34 source/main.c File Reference	84
6.34.1 Function Documentation	85
6.34.1.1 main()	85
6.34.2 Variable Documentation	85
6.34.2.1 g_msc	85
6.34.2.2 g_xMscTaskHandle	86
6.34.2.3 g_xMscTaskStack	86
6.34.2.4 g_xMscTaskTCB	86
6.34.2.5 g_xRecordTaskHandle	86
6.34.2.6 g_xRecordTaskStack	86
6.34.2.7 g_xRecordTaskTCB	86
6.34.2.8 g_xSemMassStorage	86
6.34.2.9 g_xSemRecord	87
6.35 source/mass_storage.c File Reference	87

6.35.1 Function Documentation	87
6.35.1.1 MSC_DeviceMscApp()	87
6.35.1.2 MSC_DeviceMscAppTask()	87
6.35.1.3 USB1_HS_IRQHandler()	88
6.35.2 Variable Documentation	88
6.35.2.1 g_xSemMassStorage	88
6.35.2.2 g_xSemRecord	88
6.36 source/parser.c File Reference	88
6.36.1 Macro Definition Documentation	89
6.36.1.1 RECORD_LED_TIME_INTERVAL	89
6.36.2 Function Documentation	89
6.36.2.1 PARSER_ClearConfig()	89
6.36.2.2 PARSER_GetBaudrate()	89
6.36.2.3 PARSER_GetConfig()	90
6.36.2.4 PARSER_GetDataBits()	90
6.36.2.5 PARSER_GetFileSize()	90
6.36.2.6 PARSER_GetFreeSpaceLimitMB()	90
6.36.2.7 PARSER_GetMaxBytes()	91
6.36.2.8 PARSER_GetParity()	91
6.36.2.9 PARSER_GetStopBits()	91
6.36.2.10 PARSER_GetVersion()	91
6.36.2.11 PARSER_ParseBaudrate()	91
6.36.2.12 PARSER_ParseDataBits()	92
6.36.2.13 PARSER_ParseFileSize()	92
6.36.2.14 PARSER_ParseFreeSpace()	92
6.36.2.15 PARSER_ParseParity()	93
6.36.2.16 PARSER_ParseStopBits()	93
6.36.3 Variable Documentation	93
6.36.3.1 g_config	93
6.37 source/pwrloss_det.c File Reference	94
6.37.1 Macro Definition Documentation	94
6.37.1.1 REF_VOLTAGE	94
6.37.1.2 TRIGGER_VOLTAGE	94
6.37.2 Function Documentation	94
6.37.2.1 CTIMER4_IRQHandler()	94
6.37.2.2 HSCMP1_IRQHandler()	94
6.37.2.3 PWRLOSS_DetectionInit()	95
6.38 source/record.c File Reference	95
6.38.1 Macro Definition Documentation	97
6.38.1.1 BLOCK_SIZE	97
6.38.1.2 CIRCULAR_BUFFER_SIZE	97
6.38.1.3 GET_CURRENT_TIME_MS	97

6.38.1.4 GET_WAIT_INTERVAL	97
6.38.2 Function Documentation	97
6.38.2.1 CONSOLELOG_CheckFileSystem()	97
6.38.2.2 CONSOLELOG_ClearTransferredBytes()	98
6.38.2.3 CONSOLELOG_CreateDirectory()	98
6.38.2.4 CONSOLELOG_CreateFile()	98
6.38.2.5 CONSOLELOG_Deinit()	98
6.38.2.6 CONSOLELOG_Flush()	99
6.38.2.7 CONSOLELOG_GetFlushCompleted()	99
6.38.2.8 CONSOLELOG_GetFreeSpaceMB()	99
6.38.2.9 CONSOLELOG_GetTransferredBytes()	99
6.38.2.10 CONSOLELOG_Init()	100
6.38.2.11 CONSOLELOG_PowerLossFlush()	100
6.38.2.12 CONSOLELOG_ProccessConfigFile()	100
6.38.2.13 CONSOLELOG_ReadConfig()	101
6.38.2.14 CONSOLELOG_Recording()	101
6.38.2.15 get_fatime()	101
6.38.2.16 LP_FLEXCOMM3_IRQHandler()	101
6.39 source/semihost_hardfault.c File Reference	102
6.39.1 Function Documentation	102
6.39.1.1 __attribute__()	102
6.40 source/task_switching.c File Reference	102
6.40.1 Function Documentation	102
6.40.1.1 USB_State()	102
6.41 source/temperature.c File Reference	103
6.41.1 Macro Definition Documentation	104
6.41.1.1 BUFF_SIZE	104
6.41.1.2 I2C_BAUDRATE	104
6.41.1.3 I2C_MASTER_I2C5	104
6.41.1.4 LPI2C_RX_DMA_CHANNEL	104
6.41.1.5 LPI2C_RX_EDMA_CHANNEL	104
6.41.1.6 LPI2C_TX_CHANNEL	105
6.41.1.7 LPI2C_TX_DMA_CHANNEL	105
6.41.1.8 P3T1755_ADDR_7BIT	105
6.41.1.9 PRINT_REG_OUTPUT	105
6.41.1.10 REGISTER_CONFIG	105
6.41.1.11 REGISTER_TEMPERATURE	105
6.41.1.12 REGISTER_THVST	105
6.41.1.13 REGISTER_TOS	105
6.41.2 Function Documentation	106
6.41.2.1 AT_NONCACHEABLE_SECTION() [1/2]	106
6.41.2.2 AT_NONCACHEABLE_SECTION() [2/2]	106

6.41.2.3 CTIMER0_IRQHandler()	106
6.41.2.4 Ipi2c_callback()	106
6.41.2.5 Read()	106
6.41.2.6 TMP_GetTemperature()	106
6.41.2.7 TMP_Init()	107
6.41.2.8 Write()	107
6.41.3 Variable Documentation	107
6.41.3.1 g_bCompletionFlag_I2C5	107
6.41.3.2 g_EdmaRxHandle_I2C5	107
6.41.3.3 g_EdmaTxHandle_I2C5	107
6.41.3.4 g_Xfer_I2C5	108
6.42 source/time.c File Reference	108
6.42.1 Macro Definition Documentation	108
6.42.1.1 EXTERNAL_RTC_TIME_OFFSET	108
6.42.1.2 INTERNAL_RTC_BASE_YEAR	108
6.42.2 Function Documentation	108
6.42.2.1 TIME_InitIRTC()	108
6.43 source/uart.c File Reference	109
6.43.1 Macro Definition Documentation	109
6.43.1.1 BUFFER_SIZE	109
6.43.2 Function Documentation	109
6.43.2.1 UART_Deinit()	109
6.43.2.2 UART_Disable()	109
6.43.2.3 UART_Enable()	110
6.43.2.4 UART_Init()	110
6.43.2.5 UART_Print()	110
Index	111

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Task Management	7
Buffers and Recording Management	8
Management	11
DS3231 Real-Time Clock Driver	12

Chapter 2

Class Index

2.1 Class List

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

REC_config_t	Configuration structure for the recording system	19
RTC_date_t	Structure for Keeping Date	21
RTC_time_t	Structure for Keeping Time	22

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

drivers/ rtc_ds3231.c	23
drivers/ rtc_ds3231.h	26
include/ app_init.h	35
include/ app_tasks.h	37
include/ defs.h	40
include/ error.h	46
include/ led.h	50
include/ mass_storage.h	54
include/ parser.h	56
include/ pwrloss_det.h	63
include/ record.h	65
include/ task_switching.h	72
include/ temperature.h	73
include/ time.h	75
include/ uart.h	77
source/ app_init.c	79
source/ app_tasks.c	80
source/ error.c	82
source/ led.c	83
source/ main.c	84
source/ mass_storage.c	87
source/ parser.c	88
source/ pwrloss_det.c	94
source/ record.c	95
source/ semihost_hardfault.c	102
source/ task_switching.c	102
source/ temperature.c	103
source/ time.c	108
source/ uart.c	109

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_u32BytesTransferred` = 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_u32BytesTransferred

```
uint32_t g_u32BytesTransferred = 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

<i>dec</i>	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

<i>bcd</i>	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

<i>enable</i>	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

<i>enable</i>	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

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

4.4.2.8 RTC_GetDate()

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

Gets Date.

Parameters

<i>pDate</i>	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

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

4.4.2.11 RTC_Init()

```
uint8_t RTC_Init (
    void )
```

Initialize The RTC DS3231.

Parameters

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

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

4.4.2.12 RTC_Read()

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

Reads Value From RTC Register.

Parameters

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

Returns

Read Value.

4.4.2.13 RTC_SetDate()

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

Sets Date.

Parameters

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

4.4.2.14 RTC_SetDateDefault()

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

Sets Date To Default.

Parameters

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

4.4.2.15 RTC_SetInterruptMode()

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

Sets The INTCN Bit in Control Register.

Parameters

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

4.4.2.16 RTC_SetOscState()

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

Sets The Oscillator Stop Flag (OSF).

Parameters

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

4.4.2.17 RTC_SetTime()

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

Sets Time.

Parameters

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

4.4.2.18 RTC_SetTimeDefault()

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

Sets Time To Default.

Parameters

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

4.4.2.19 RTC_Write()

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

Writes Value Into RTC Registers.

Parameters

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

Returns

void

Chapter 5

Class Documentation

5.1 REC_config_t Struct Reference

Configuration structure for the recording system.

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

Public Attributes

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

5.1.1 Detailed Description

Configuration structure for the recording system.

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

5.1.2 Member Data Documentation

5.1.2.1 baudrate

```
uint32_t REC_config_t::baudrate
```

Desired Baudrate

5.1.2.2 data_bits

```
lpuart_data_bits_t REC_config_t::data_bits
```

Number of Data Bit

5.1.2.3 free_space_limit_mb

```
uint32_t REC_config_t::free_space_limit_mb
```

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

5.1.2.4 max_bytes

```
uint32_t REC_config_t::max_bytes
```

Number of Bytes Between LED Signal

5.1.2.5 parity

```
lpuart_parity_mode_t REC_config_t::parity
```

Parity Bit

5.1.2.6 size

```
uint32_t REC_config_t::size
```

Maximum File Size Maximal Log. Time In Per File

5.1.2.7 stop_bits

```
lpuart_stop_bit_count_t REC_config_t::stop_bits
```

Number of Stop Bit

5.1.2.8 version

```
REC_version_t REC_config_t::version
```

NXP Board That Will Be Recorded

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

- [include/parser.h](#)

5.2 RTC_date_t Struct Reference

Structure for Keeping Date.

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

Public Attributes

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

5.2.1 Detailed Description

Structure for Keeping Date.

5.2.2 Member Data Documentation

5.2.2.1 date

```
uint8_t RTC_date_t::date
```

Day 1..31

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 */
- #define [BYTE_1](#) (0U)
- #define [BYTE_2](#) (1U)
- #define [CLK_STATE](#)(x)
- #define [OSF_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}

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 */
```

6.1.1.5 OSF_STATE

```
#define OSF_STATE(
    x)
```

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.

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}
```

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 `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)
- #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)
- #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
Desired Baud Rate For I2C Bus.
- #define `APP_SUCCESS` 0

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
```

6.2.1.3 DS3231_A1F

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

6.2.1.4 DS3231_A1IE

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

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)
```

6.2.1.8 DS3231_ADDR_DATE

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

6.2.1.9 DS3231_ADDR_DAY

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

6.2.1.10 DS3231_ADDR_HRS

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

6.2.1.11 DS3231_ADDR_I2C

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

6.2.1.12 DS3231_ADDR_MIN

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

6.2.1.13 DS3231_ADDR_MONTH

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

6.2.1.14 DS3231_ADDR_SEC

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

6.2.1.15 DS3231_ADDR_YEAR

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

6.2.1.16 DS3231_INTCN

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

6.2.1.17 DS3231_REG_CTRL

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

6.2.1.18 DS3231_REG_STATUS

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

6.2.1.19 E_FAULT

```
#define E_FAULT 1
```

General Error Return Code.

6.2.1.20 FRIDAY

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

6.2.1.21 I2C_BAUDRATE

```
#define I2C_BAUDRATE 100000U
```

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)
```

6.2.1.28 OSC_STOPPED

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

6.2.1.29 SATURDAY

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

6.2.1.30 SUNDAY

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

6.2.1.31 THURSDAY

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

6.2.1.32 TIM_CYCLE_12H

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

6.2.1.33 TIM_CYCLE_12H_AM

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

6.2.1.34 TIM_CYCLE_12H_PM

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

6.2.1.35 TIM_CYCLE_24H

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

6.2.1.36 TUESDAY

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

6.2.1.37 WEDNESDAY

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

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
00036 #define LPI2C_TX_DMA_CHANNEL 0U
00037
00041 #define LPI2C_RX_DMA_CHANNEL 1U
00042
00046 #define LPI2C_TX_CHANNEL kDma0RequestMuxLpFlexcomm2Tx
00047
00051 #define LPI2C_RX_EDMA_CHANNEL kDma0RequestMuxLpFlexcomm2Rx
00052
00056 #define I2C_MASTER ((LPI2C_Type *)LPI2C2_BASE)
00057
00058 /*
00059  * @brief Alarm Interrupt Enable Bits.
00060  */
00061 #define DS3231_A1IE (0x00u)
00062 #define DS3231_A2IE (0x1u)
00063 /*
00064  * @brief Interrupt Control Bit.
00065  */
00066 #define DS3231_INTCN (0x2u)
00067 /*
00068  * @brief Alarm 1 & Alarm 2 Flags.
00069  * @details A Logic 1 in The Alarm 'x' Flag Bit Indicates That The Time Matched The Alarm 'x'
00070  * Registers.
00071  */
00071 #define DS3231_A1F (0x0u)
00072 #define DS3231_A2F (0x1u)
00073
00074 #define ALARM_DISABLED (0x0u)
00075 /*
00076  * @brief Indicates That Oscillator Has Stopped & Time Has To Updated.
00077  */
00078 #define OSC_STOPPED (0x00u)
00079
00080 /*
00081  * @brief Registers Addresses of DS3231.
00082  */
00083 #define DS3231_ADDR_SEC (0x00U)
00084 #define DS3231_ADDR_MIN (0x01U)
00085 #define DS3231_ADDR_HRS (0x02U)
00086 #define DS3231_ADDR_DAY (0x03U)

```

```

00087
00088 #define DS3231_ADDR_DATE          (0x04U)
00089 #define DS3231_ADDR_MONTH        (0x05U)
00090 #define DS3231_ADDR_YEAR         (0x06U)
00091 #define DS3231_ADDR_CENT         (0x07U)
00092 /*
00093  * @brief Address of Status Register.
00094  */
00095 #define DS3231_REG_STATUS          (0x0Fu)
00096 /*
00097  * @brief Address of Control Register.
00098  */
00099
00100 #define DS3231_REG_CTRL            (0x0Eu)
00101 /*
00102  * @brief Address of I2C Slave.
00103  */
00104 #define DS3231_ADDR_I2C            (0x68U) //<! The Slave Address Byte Contains In 7-bit: 1101000
00105
00106 /*
00107  * @brief Definition of Dates.
00108  */
00109 #define SUNDAY                     (0x1)
00110 #define MONDAY                     (0x2)
00111 #define TUESDAY                    (0x3)
00112 #define WEDNESDAY                  (0x4)
00113 #define THURSDAY                   (0x5)
00114 #define FRIDAY                     (0x6)
00115 #define SATURDAY                   (0x7)
00116
00117 /*
00118  * @brief Definitions For Time Format Handling.
00119  */
00120 #define TIM_CYCLE_12H              (0x0)
00121 #define TIM_CYCLE_12H_AM           (0x0)
00122 #define TIM_CYCLE_12H_PM           (0x1)
00123 #define TIM_CYCLE_24H              (0x2)
00124
00125 /*
00126  * @brief Application Configurable Items.
00127  */
00128
00129
00134 #define I2C_BAUDRATE                100000U
00135
00136
00137 #define APP_SUCCESS                 0
00138 /*****
00139  * Structures
00140  *****/
00144 typedef struct
00145 {
00146     uint8_t format;
00147     uint8_t sec;
00148     uint8_t min;
00149     uint8_t hrs;
00150 } RTC_time_t;
00151
00152
00156 typedef struct
00157 {
00158     uint8_t date;
00159     uint8_t day;
00160     uint8_t month;
00161     uint8_t year;
00162 } RTC_date_t;
00163
00164
00168 typedef enum
00169 {
00170     OSC_OK = 0,
00171     OSC_INTERRUPTED
00172 } RTC_osc_state_t;
00173
00174
00175 /*
00176  * @brief An Enum For Interrupt Mode.
00177  */
00178 typedef enum
00179 {
00180     SQUARE_WAVE_INTERRUPT = 0,
00181     ALARM_INTERRUPT
00182 } RTC_interrupt_mode_t;
00183
00184 /*****
00185  * Prototypes
00186  *****/

```



```

00187
00193
00201 uint8_t RTC_Init(void);
00202
00208 void RTC_Deinit(void);
00209
00215 static uint8_t RTC_ConvertToBCD(uint8_t dec);
00216
00222 static uint8_t RTC_ConvertToDEC(uint8_t bcd);
00223
00229 uint8_t RTC_GetState(void);
00230
00235 void RTC_SetOscState(RTC_osc_state_t state);
00236
00243 uint8_t RTC_Write(uint8_t regAddress, uint8_t val);
00244
00250 uint8_t RTC_Read(uint8_t regAddress);
00251
00252
00257 void RTC_SetTime(RTC_time_t *pTime);
00258
00263 void RTC_GetTime(RTC_time_t *pTime);
00264
00269 void RTC_SetDate(RTC_date_t *pDate);
00270
00275 void RTC_GetDate(RTC_date_t *pDate);
00276
00281 void RTC_SetInterruptMode(RTC_interrupt_mode_t mode);
00282
00287 void RTC_CtrlAlarm1(uint8_t enable);
00288
00292 void RTC_ClearFlagAlarm1(void);
00293
00298 void RTC_CtrlAlarm2(uint8_t enable);
00299
00303 void RTC_ClearFlagAlarm2(void);
00304
00309 void RTC_SetTimeDefault(RTC_time_t *pTime);
00310
00315 void RTC_SetDateDefault(RTC_date_t *pDate);
00316 // end of RTC group
00318
00319 #endif /* RTC_DS3231_H_*/

```

6.4 include/app_init.h File Reference

```

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

```

Functions

- void [APP_InitBoard](#) (void)

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

6.4.1 Function Documentation

6.4.1.1 APP_InitBoard()

```
void APP_InitBoard (
    void )
```

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

6.5 app_init.h

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

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

6.6 include/app_tasks.h File Reference

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

Macros

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

Functions

- void [msc_task](#) (void *handle)
Task Responsible For Mass Storage Functionality in Device Mode.
- void [record_task](#) (void *handle)
- 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

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

6.6.2.2 record_task()

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

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

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

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:  Includes Implementation of Task For FreeRTOS.
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       Includes Implementation of Task For FreeRTOS.
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 /* Task Handles */
00043 extern TaskHandle_t g_xMscTaskHandle;
00044
00045 extern TaskHandle_t g_xRecordTaskHandle;
00046
00047 extern SemaphoreHandle_t g_xSemRecord;
00048
00049 extern SemaphoreHandle_t g_xSemMassStorage;
00050 /*****
00051  * Prototypes
00052  *****/
00053
00054
00055 void msc_task(void *handle);
00056
00057 void record_task(void *handle);
00058 void vApplicationGetIdleTaskMemory(StaticTask_t **ppxIdleTaskTCBBuffer,
00059                                     StackType_t **ppxIdleTaskStackBuffer,
00060                                     uint32_t *pulIdleTaskStackSize);
00061
00062 void vApplicationGetTimerTaskMemory(StaticTask_t **ppxTimerTaskTCBBuffer,
00063                                     StackType_t **ppxTimerTaskStackBuffer,
00064                                     uint32_t *pulTimerTaskStackSize);
00065
00066 #endif /* APP_TASKS_H_ */

```

6.8 include/defs.h File Reference

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

Macros

- #define `NOT_IMPLEMENTED` `#error "This Feature Is Not Implemented!"`
- #define `MSC_STACK_SIZE` `((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))`
Defines The Stack Size For Mass Storage Task.
- #define `RECORD_STACK_SIZE` `((uint32_t)(5000UL / (uint32_t)sizeof(portSTACK_TYPE)))`
Defines The Stack Size For Recording Task.
- #define `MSC_ENABLED` `(true)`
Enables/Disables Mass Storage Functionality.
- #define `INFO_ENABLED` `(false)`
Enables/Disables Debug Mode.
- #define `DEBUG_ENABLED` `(false)`
Enables/Disables Debug Mode.
- #define `UART_FIFO_ENABLED` `(true)`
Enables/Disables HW FIFO Queue on Application LPUART.
- #define `UART_FIFO_LENHT` `(4u)`
Defines The Size of HW FIFO Queue.
- #define `UART_PRINT_ENABLED` `(false)`
Enables/Disables Print of Received Bytes From Application LPUART To Console.
- #define `PWRLOSS_DETECTION_ENABLED` `(true)`
Enables/Disables Power Loss Detection.

- #define `PWRLOSS_TEST_GPIOS` (false)
Enables/Disables GPIO For Testing Power Loss Detection.
- #define `TAU` 16UL
Constant Tau, When Back-Up Power Capacitor is Charged To 99%.
- #define `PWRLOSS_DET_ACTIVE_IN_TIME_TAU`
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 TAU
```

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 TAU

```
#define TAU 16UL
```

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

In Seconds (It's Actually 16s).

6.8.1.21 TEMPERATURE_MEAS_ENABLED

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

Enables/Disables Temperature Recording.

6.8.1.22 UART_FIFO_ENABLED

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

Enables/Disables HW FIFO Queue on Application LPUART.

6.8.1.23 UART_FIFO_LENHT

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

Defines The Size of HW FIFO Queue.

6.8.1.24 UART_PRINT_ENABLED

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

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

6.8.1.25 UART_RECEIVE_PRIO

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

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

6.9 defs.h

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

```
00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    defs.h
00004  * Author:       Tomas Dolak
00005  * Date:         22.09.2024
00006  * Description:   Header File Providing Definitions To Set Up The Project.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          defs.h
00013  * @author        Tomas Dolak
00014  * @date          22.09.2024
00015  * @brief         Header File Providing Definitions To Set Up The Project.
00016  * *****/
00017
00018 #ifndef DEFS_H_
00019 #define DEFS_H_
00020
00021 /*****
00022  * Includes
00023  *****/
00024 #include <stdbool.h>
00025
00026
00027 /*****
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_LENHT     (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 TAU                   16UL
00089
00094 #define PWRLOSS_DET_ACTIVE_IN_TIME TAU
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
- #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

- 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)

6.12.1 Macro Definition Documentation

6.12.1.1 BACKUP_POWER_LED_PIN

```
#define BACKUP_POWER_LED_PIN 0x11
```

6.12.1.2 BACKUP_POWER_LED_PORT

```
#define BACKUP_POWER_LED_PORT GPIO4
```

6.12.1.3 ERROR_LED_PIN_RECORD

```
#define ERROR_LED_PIN_RECORD 0x09
```


6.12.1.4 ERROR_LED_PORT

```
#define ERROR_LED_PORT GPIO0
```

6.12.1.5 FLUSH_LED_PORT

```
#define FLUSH_LED_PORT GPIO0
```

6.12.1.6 MEMORY_LOW_LED_PIN

```
#define MEMORY_LOW_LED_PIN 0x07
```

6.12.1.7 MEMORY_LOW_LED_PORT

```
#define MEMORY_LOW_LED_PORT GPIO0
```

6.12.1.8 RECORD_LED_PIN

```
#define RECORD_LED_PIN 0x0B
```

6.12.1.9 RECORD_LED_PIN_FLUSH

```
#define RECORD_LED_PIN_FLUSH 0x0D
```

6.12.1.10 RECORD_LED_PORT

```
#define RECORD_LED_PORT GPIO2
```

6.12.2 Function Documentation

6.12.2.1 LED_ClearSignalFlush()

```
void LED_ClearSignalFlush (  
    void )
```

6.12.2.2 LED_SetHigh()

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

6.12.2.3 LED_SetLow()

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

6.12.2.4 LED_SignalBackUpPowerAvailable()

```
void LED_SignalBackUpPowerAvailable (
    void )
```

6.12.2.5 LED_SignalError()

```
void LED_SignalError (
    void )
```

6.12.2.6 LED_SignalFlush()

```
void LED_SignalFlush (
    void )
```

6.12.2.7 LED_SignalLowMemory()

```
void LED_SignalLowMemory (
    void )
```

6.12.2.8 LED_SignalReady()

```
void LED_SignalReady (
    void )
```

6.12.2.9 LED_SignalRecording()

```
void LED_SignalRecording (
    void )
```

6.12.2.10 LED_SignalRecordingStop()

```
void LED_SignalRecordingStop (
    void )
```

6.13 led.h

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

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    leds.c
00004  * Author:       Tomas Dolak
00005  * Date:         06.02.2025
00006  * Description:  Implements The Logic For LEDs Control.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file         leds.c
00013  * @author       Tomas Dolak
00014  * @date         06.02.2025
00015  * @brief        Implements The Logic For LEDs Control.
00016  * *****/
00017
00018 #ifndef LED_H_
00019 #define LED_H_
00020
00021 /*****
00022  * Includes
00023  * *****/
00024 #include "fsl_gpio.h"
00025 #include "board.h"
00026 /*****
00027  * Functions Macros
00028  * *****/
00029 /*
00030  * @brief Error LED's Port Number.
00031  */
00032 #define ERROR_LED_PORT      GPIO0
00033
00034 /*
00035  * @brief Error LED's Pin 2.
00036  * @details P0_9
00037  */
00038 #define ERROR_LED_PIN_RECORD    0x09
00039
00040
00041 /*
00042  * @brief Low Memory LED's Port Number.
00043  */
00044 #define MEMORY_LOW_LED_PORT    GPIO0
00045
00046 /*
00047  * @brief Error LED's Pin 1.
00048  * @details P0_7
00049  */
00050 #define MEMORY_LOW_LED_PIN      0x07
00051
00052
00053 /*
00054  * @brief Flush LED's Port Number.
00055  */
00056 #define FLUSH_LED_PORT          GPIO0
00057
00058 /*
00059  * @brief Error LED's Pin 3.
00060  * @details P0_13
00061  */
00062 #define RECORD_LED_PIN_FLUSH    0x0D
00063
00064 /*
00065  * @brief Recording LED's Port Number.
00066  * @details Signals That Device Records.
00067  */
00068 #define RECORD_LED_PORT          GPIO2
00069
00070 /*
00071  * @brief Record LED's Pin.
00072  * @details P2_11
00073  */
00074 #define RECORD_LED_PIN          0x0B
00075
00076 /*
00077  * @brief Recording LED's Port Number.
00078  * @details Signals That Device Records.
00079  */
00080 #define BACKUP_POWER_LED_PORT    GPIO4
00081
00082 */

```

```

00083  * @brief   Record LED's Pin.
00084  * @details P4_17
00085  */
00086 #define BACKUP_POWER_LED_PIN    0x11
00087 /*****
00088  * Functions Definitions
00089  *****/
00090
00091 /*
00092  * @brief   Sets Logic 1 on GPIO Pin.
00093  * @param port_base Pointer to GPIO Instance.
00094  * @param pin      Pin.
00095  */
00096 void LED_SetHigh(GPIO_Type *port_base, uint32_t pin);
00097
00098 /*
00099  * @brief   Sets Logic 0 on GPIO Pin.
00100  * @param port_base Pointer to GPIO Instance.
00101  * @param pin      Pin.
00102  */
00103 void LED_SetLow(GPIO_Type *port_base, uint32_t pin);
00104
00105 /*
00106  * @brief Indicates Ready State of The Recording Device.
00107  */
00108 void LED_SignalReady(void);
00109
00110 /*
00111  * @brief Signals That Device Is Currently Receiving Bytes (Recording).
00112  */
00113 void LED_SignalRecording(void);
00114
00115 /*
00116  * @brief Signals That Device Is Stopped Receiving Bytes (Recording).
00117  */
00118 void LED_SignalRecordingStop(void);
00119
00120 /*
00121  * @brief Signals Configuration File Missing or Contains Unexpected Data.
00122  */
00123 void LED_SignalBackUpPowerAvailable(void);
00124
00125 /*
00126  * @brief Signals That There Will Be Empty Space on SD Card.
00127  */
00128 void LED_SignalLowMemory(void);
00129
00130 /*
00131  * @brief Signals Error During Recording.
00132  */
00133 void LED_SignalError(void);
00134
00135 /*
00136  * @brief Signals That Flush Has Been Activated.
00137  */
00138 void LED_SignalFlush(void);
00139
00140 /*
00141  * @brief Clears Flush LED After UART Re-Initialization.
00142  */
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	<i>chContent</i>	Pointer To Content of Configuration File.
----	------------------	---

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.12 PARSER_ParseDataBits()

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

Parse The Number of Data Bits From Configuration File.

Parameters

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

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.13 PARSER_ParseFileSize()

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

Parse Record File Size From Configuration File.

Parameters

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

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.14 PARSER_ParseFreeSpace()

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

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

Parameters

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

Returns

ERROR_NONE If The Parsing Succeed.

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

Returns

ERROR_NONE If The Parsing Succeed.

6.16.2.16 PARSER_ParseStopBits()

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

Parse The Number of Stop Bits From Configuration File.

Parameters

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

Returns

ERROR_NONE If The Parsing Succeed.

6.17 parser.h

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

```

00001 /*****
00002  * Project:      NXP MCXN947 Datalogger
00003  * File Name:    parser.h
00004  * Author:       Tomas Dolak
00005  * Date:         16.04.2025
00006  * Description:  Header File Providing Definitions For Configuration File Parser.
00007  *
00008  * *****/
00009
00010 /*****
00011  * @package      NXP MCXN947 Datalogger
00012  * @file          parser.h
00013  * @author        Tomas Dolak
00014  * @date          16.04.2025
00015  * @brief         Header File Providing Definitions For Configuration File Parser.
00016  * *****/
00017 #ifndef 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  *****/
00039 typedef enum
00040 {
00041     WCT_UNKOWN = 0,
00042     WCT_AUTOS1,
00043     WCT_AUTOS2
00044 } REC_version_t;
00045
00046 typedef struct
00047 {
00048     /* Recorded NXP Device */
00049     REC_version_t version;
00050
00051     /* UART Setup */
00052     uint32_t baudrate;
00053     lpuart_stop_bit_count_t stop_bits;
00054     lpuart_data_bits_t data_bits;
00055     lpuart_parity_mode_t parity;
00056
00057     uint32_t size;
00058     uint32_t max_bytes;
00059     uint32_t free_space_limit_mb;
00060 } REC_config_t;
00061
00062 /*****
00063  * Prototypes
00064  *****/
00065 REC_config_t PARSER_GetConfig(void);
00066
00067 REC_version_t PARSER_GetVersion(void);
00068
00069 uint32_t PARSER_GetBaudrate(void);
00070
00071 uint32_t PARSER_GetFileSize(void);
00072
00073 lpuart_data_bits_t PARSER_GetDataBits(void);
00074
00075 lpuart_parity_mode_t PARSER_GetParity(void);
00076
00077 lpuart_stop_bit_count_t PARSER_GetStopBits(void);
00078
00079 uint32_t PARSER_GetFreeSpaceLimitMB(void);
00080

```

```

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 [DEMO_LPCMP_BASE](#) CMP1
- #define [DEMO_LPCMP_USER_CHANNEL](#) 2U
- #define [DEMO_LPCMP_DAC_CHANNEL](#) 7U
- #define [DEMO_LPCMP_IRQ_ID](#) HSCMP1_IRQn
- #define [DEMO_LPCMP_IRQ_HANDLER_FUNC](#) HSCMP1_IRQHandler
- #define [DEMO_VREF_BASE](#) VREF0
- #define [DEMO_SPC_BASE](#) SPC0
- #define [CTIMER](#) CTIMER4 /* Timer 4 */
- #define [CTIMER_MAT0_OUT](#) kCTIMER_Match_0 /* Match output 0 */
- #define [CTIMER_EMT0_OUT](#) (1u << kCTIMER_Match_0)
- #define [CTIMER_CLK_FREQ](#) CLOCK_GetCTimerClkFreq(4U)

Functions

- void [PWRLOSS_DetectionInit](#) (void)

6.18.1 Macro Definition Documentation

6.18.1.1 CTIMER

```
#define CTIMER CTIMER4 /* Timer 4 */
```

6.18.1.2 CTIMER_CLK_FREQ

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

6.18.1.3 CTIMER_EMT0_OUT

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

6.18.1.4 CTIMER_MAT0_OUT

```
#define CTIMER_MAT0_OUT kCTIMER_Match_0 /* Match output 0 */
```

6.18.1.5 DEMO_LPCMP_BASE

```
#define DEMO_LPCMP_BASE CMP1
```

6.18.1.6 DEMO_LPCMP_DAC_CHANNEL

```
#define DEMO_LPCMP_DAC_CHANNEL 7U
```

6.18.1.7 DEMO_LPCMP_IRQ_HANDLER_FUNC

```
#define DEMO_LPCMP_IRQ_HANDLER_FUNC HSCMP1_IRQHandler
```

6.18.1.8 DEMO_LPCMP_IRQ_ID

```
#define DEMO_LPCMP_IRQ_ID HSCMP1_IRQn
```

6.18.1.9 DEMO_LPCMP_USER_CHANNEL

```
#define DEMO_LPCMP_USER_CHANNEL 2U
```

6.18.1.10 DEMO_SPC_BASE

```
#define DEMO_SPC_BASE SPC0
```

6.18.1.11 DEMO_VREF_BASE

```
#define DEMO_VREF_BASE VREF0
```

6.18.2 Function Documentation

6.18.2.1 PWRLOSS_DetectionInit()

```
void PWRLOSS_DetectionInit (  
    void )
```

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"          /*TODO: */
00029 #include "defs.h"
00030
00031
00032 #include "fsl_ctimer.h"
00033 /*****
00034  * Global Definitions
00035  * *****/
00036 #define DEMO_LPCMP_BASE          CMP1
00037 #define DEMO_LPCMP_USER_CHANNEL  2U
00038 #define DEMO_LPCMP_DAC_CHANNEL  7U
00039 #define DEMO_LPCMP_IRQ_ID        HSCMP1_IRQn
00040 #define DEMO_LPCMP_IRQ_HANDLER_FUNC HSCMP1_IRQHandler
00041 #define DEMO_VREF_BASE           VREF0
00042 #define DEMO_SPC_BASE            SPC0
00043
00044 #define CTIMER                    CTIMER4          /* Timer 4 */
00045 #define CTIMER_MAT0_OUT            kCTIMER_Match_0 /* Match output 0 */
00046 #define CTIMER_EMT0_OUT            (1u < kCTIMER_Match_0)
00047 #define CTIMER_CLK_FREQ            CLOCK_GetCTimerClkFreq(4U)
00048
00049 /*****
00050  * Prototypes
00051  * *****/
00052 void PWRLOSS_DetectionInit(void);
00053
00054
00055 #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 Milli-Seconds].

Functions

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

6.20.1 Macro Definition Documentation

6.20.1.1 FLUSH_TIMEOUT_TICKS

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

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

6.20.2 Function Documentation

6.20.2.1 CONSOLELOG_CheckFileSystem()

```
FRESULT CONSOLELOG_CheckFileSystem (  
    void )
```

Checks If The File System Is Initialized.

return F_OK If File System Initialized.

6.20.2.2 CONSOLELOG_ClearTransferredBytes()

```
void CONSOLELOG_ClearTransferredBytes (  
    void )
```

Clears Currently Received Bytes After LED Blinking.

6.20.2.3 CONSOLELOG_CreateDirectory()

```
error_t CONSOLELOG_CreateDirectory (  
    void )
```

Creates Directory Based Actual Date.

Returns

Returns Zero If Directory Creation Succeeded.

6.20.2.4 CONSOLELOG_CreateFile()

```
error_t CONSOLELOG_CreateFile (  
    void )
```

Creates File Based Actual Date and Counter Value.

Returns

Returns Zero If File Creation Succeeded.

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.

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_GetTransferredBytes()

```
uint32_t CONSOLELOG_GetTransferredBytes (
    void )
```

Returns Currently Received Bytes Between LED Blinking.

Returns

uint32_t Received Bytes Between LED Blinking.

6.20.2.11 CONSOLELOG_Init()

```
error_t CONSOLELOG_Init (
    void )
```

Initializes The Recording System and Mounts The File System.

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

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

Returns

error_t Returns ERROR_NONE on Success, Otherwise ERROR_FILESYSTEM.

6.20.2.12 CONSOLELOG_PowerLossFlush()

```
error_t CONSOLELOG_PowerLossFlush (
    void )
```

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

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

Returns

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

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

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

Returns

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

6.20.2.14 CONSOLELOG_ReadConfig()

```
error_t CONSOLELOG_ReadConfig (  
    void )
```

Reads and Processes The Configuration File From The Root directory.

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

Returns

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

6.20.2.15 CONSOLELOG_Recording()

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

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

Function Uses CONSOLELOG_Init To Initialize The Recording System.

Returns

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

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_GetTransferredBytes(void);
00089
00095 bool  CONSOLELOG_GetFlushCompleted(void);
00099 void  CONSOLELOG_ClearTransferredBytes(void);
00100
00104 uint32_t  CONSOLELOG_GetMaxBytes(void);
00105
00111 FRESULT  CONSOLELOG_CheckFileSystem(void);
00112
00126 error_t  CONSOLELOG_Init(void);
00127
00137 error_t  CONSOLELOG_Recording(uint32_t file_size);
00138
00148 error_t  CONSOLELOG_Flush(void);
00149
00157 error_t  CONSOLELOG_PowerLossFlush(void);
00158
00164 error_t  CONSOLELOG_Deinit(void);

```

```

00165
00175 error_t CONSOLELOG_ReadConfig(void);
00176
00187 error_t CONSOLELOG_ProccessConfigFile(const char *content);
00188
00189
00190 #endif /* RECORD_H_ */

```

6.22 include/task_switching.h File Reference

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

Functions

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

Variables

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

6.22.1 Function Documentation

6.22.1.1 USB_State()

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

Checks If Application USB Is Attached To Digital Data Logger.

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

Parameters

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

Returns

kUSB_DeviceNotifyAttach if USB Is Attached Otherwise Returns kUSB_DeviceNotifyDetach.

6.22.2 Variable Documentation

6.22.2.1 g_msc

```
usb_msc_struct_t g_msc [extern]
```

Mass Storage Descriptor.

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 For FreeRTOS.
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 For FreeRTOS.
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"

```

Macros

- #define [SENSOR_STATIC_ADDR](#) 0x48U
- #define [SENSOR_DYNAMIC_ADDR](#) 0x08U

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 Macro Definition Documentation

6.24.1.1 SENSOR_DYNAMIC_ADDR

```
#define SENSOR_DYNAMIC_ADDR 0x08U
```

6.24.1.2 SENSOR_STATIC_ADDR

```
#define SENSOR_STATIC_ADDR 0x48U
```

6.24.2 Function Documentation

6.24.2.1 Read()

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

I2C Read Function.

Parameters

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

Return values

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

6.24.2.2 TMP_GetTemperature()

```
float TMP_GetTemperature (
    void )
```

Gets Temperature From On-Board P3T1755 Temperature Sensor.

Return values

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

6.24.2.3 TMP_Init()

```
uint8_t TMP_Init (
    void )
```

Initialize On-Board P3T1755 Temperature Sensor.

6.24.2.4 Write()

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

I2C Write Function.

Parameters

in	<i>regAddress</i>	Address of The Register To Be Written To.
in	<i>val</i>	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 Of Power Loss Detection.
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 Of Power Loss Detection.
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 #define SENSOR_STATIC_ADDR 0x48U // Statická adresa senzoru (I2C mód)
00034 #define SENSOR_DYNAMIC_ADDR 0x08U // Dynamická adresa (přiřazená)
00035
00036 /*****
00037  * Prototypes
00038  * *****/
00047 uint8_t Write(uint8_t regAddress, uint8_t val[]);
00048
00056 uint16_t Read(uint8_t regAddress);
00057
00063 float TMP_GetTemperature(void);
00064
00068 uint8_t TMP_Init(void);
00069
00070 #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)`

Functions

- `error_t TIME_InitIRTC (void)`

6.26.1 Macro Definition Documentation

6.26.1.1 LPI2C_DMA_BASEADDR

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

6.26.2 Function Documentation

6.26.2.1 TIME_InitIRTC()

```
error_t TIME_InitIRTC (
    void )
```

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
00034 /*
00035  * @brief I2C Definitions.
00036  */
00037 #define LPI2C_DMA_BASEADDR (DMA0)
00038
```

```

00039 /*****
00040  * Prototypes
00041  *****/
00042
00043 /*
00044  * @brief   Initialize Internal And External Real-Time Circuits And Passes Timestamp
00045  *          Information From The External To The Internal.
00046  */
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 LPUART7 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 LPUART7 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 source/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 source/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)
- 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

- 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

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

6.31.1.2 record_task()

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

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

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

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 source/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 source/led.c File Reference

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

Functions

- 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)

6.33.1 Function Documentation

6.33.1.1 LED_ClearSignalFlush()

```
void LED_ClearSignalFlush (  
    void )
```

6.33.1.2 LED_SetHigh()

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

6.33.1.3 LED_SetLow()

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

6.33.1.4 LED_SignalBackUpPowerAvailable()

```
void LED_SignalBackUpPowerAvailable (  
    void )
```

6.33.1.5 LED_SignalError()

```
void LED_SignalError (  
    void )
```

6.33.1.6 LED_SignalFlush()

```
void LED_SignalFlush (  
    void )
```

6.33.1.7 LED_SignalLowMemory()

```
void LED_SignalLowMemory (  
    void )
```

6.33.1.8 LED_SignalReady()

```
void LED_SignalReady (  
    void )
```

6.33.1.9 LED_SignalRecording()

```
void LED_SignalRecording (  
    void )
```

6.33.1.10 LED_SignalRecordingStop()

```
void LED_SignalRecordingStop (  
    void )
```

6.34 source/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 source/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 source/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	<i>chContent</i>	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	<i>chContent</i>	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	<i>chContent</i>	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	<i>chContent</i>	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	<i>chContent</i>	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	<i>chContent</i>	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 source/pwrloss_det.c File Reference

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

Macros

- #define [REF_VOLTAGE](#) 0xFFu
- #define [TRIGGER_VOLTAGE](#) 0xE1u

Functions

- void [CTIMER4_IRQHandler](#) (void)
- void [HSCMP1_IRQHandler](#) (void)
- void [PWRLOSS_DetectionInit](#) (void)

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 0xE1u
```

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 )
```

6.38 source/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_GetTransferredBytes` (void)
Returns Currently Received Bytes Between LED Blinking.
- bool `CONSOLELOG_GetFlushCompleted` (void)
Returns If Flush Was Completed or Not.
- void `CONSOLELOG_ClearTransferredBytes` (void)
Clears Currently Received Bytes After LED Blinking.
- 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_u32BytesTransferred](#) = 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

<i>seconds</i>	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_ClearTransferredBytes()**

```
void CONSOLELOG_ClearTransferredBytes (
    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.

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.

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_GetTransferredBytes()**

```
uint32_t CONSOLELOG_GetTransferredBytes (
    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	<i>content</i>	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 source/semihost_hardfault.c File Reference

Functions

- [__attribute__](#) ((naked))

6.39.1 Function Documentation

6.39.1.1 __attribute__()

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

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

Returns

kUSB_DeviceNotifyAttach if USB Is Attached Otherwise Returns kUSB_DeviceNotifyDetach.

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

Return values

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

6.41.2.6 TMP_GetTemperature()

```
float TMP_GetTemperature (
    void )
```

Gets Temperature From On-Board P3T1755 Temperature Sensor.

Return values

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

Return values

<i>If</i>	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 source/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_OFFSET 2000`
The Year Range Is Between 0 and 99.

Functions

- `error_t TIME_InitIRTC (void)`

6.42.1 Macro Definition Documentation

6.42.1.1 EXTERNAL_RTC_TIME_OFFSET

```
#define EXTERNAL_RTC_TIME_OFFSET 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 )
```

6.43 source/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 LPUART7 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 LPUART7 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

- `__attribute__`
 - `semihost_hardfault.c`, [102](#)
- `ALARM_DISABLED`
 - `rtc_ds3231.h`, [28](#)
- `ALARM_INTERRUPT`
 - `rtc_ds3231.h`, [32](#)
- `app_init.c`
 - `APP_InitBoard`, [80](#)
- `app_init.h`
 - `APP_InitBoard`, [36](#)
- `APP_InitBoard`
 - `app_init.c`, [80](#)
 - `app_init.h`, [36](#)
- `APP_SUCCESS`
 - `rtc_ds3231.h`, [28](#)
- `app_tasks.c`
 - `msc_task`, [81](#)
 - `record_task`, [81](#)
 - `vApplicationGetIdleTaskMemory`, [81](#)
 - `vApplicationGetTimerTaskMemory`, [81](#)
- `app_tasks.h`
 - `g_xMscTaskHandle`, [39](#)
 - `g_xRecordTaskHandle`, [39](#)
 - `g_xSemMassStorage`, [39](#)
 - `g_xSemRecord`, [39](#)
 - `msc_task`, [37](#)
 - `record_task`, [38](#)
 - `TASK_PRIO`, [37](#)
 - `vApplicationGetIdleTaskMemory`, [38](#)
 - `vApplicationGetTimerTaskMemory`, [38](#)
- `AT_NONCACHEABLE_SECTION`
 - `rtc_ds3231.c`, [25](#)
 - `temperature.c`, [106](#)
- `BACKUP_POWER_LED_PIN`
 - `led.h`, [50](#)
- `BACKUP_POWER_LED_PORT`
 - `led.h`, [50](#)
- `baudrate`
 - `REC_config_t`, [19](#)
- `BLOCK_SIZE`
 - `record.c`, [97](#)
- `BUFF_SIZE`
 - `temperature.c`, [104](#)
- `BUFFER_SIZE`
 - `uart.c`, [109](#)
- `Buffers and Recording Management`, [8](#)
 - `g_bBackDmaBufferReady`, [9](#)
 - `g_bFlushCompleted`, [9](#)
 - `g_fileObject`, [9](#)
 - `g_fileSystem`, [10](#)
 - `g_lastDataTick`, [10](#)
 - `g_pu8BackDmaBuffer`, [10](#)
 - `g_pu8FrontDmaBuffer`, [10](#)
 - `g_u16BackDmaBufferIdx`, [10](#)
 - `g_u32BytesTransferred`, [10](#)
 - `g_u32CurrentFileSize`, [10](#)
 - `g_u32FileCounter`, [11](#)
 - `g_u8CurrentDirectory`, [11](#)
 - `SDK_ALIGN`, [9](#)
- `BYTE_1`
 - `rtc_ds3231.c`, [24](#)
- `BYTE_2`
 - `rtc_ds3231.c`, [24](#)
- `CIRCULAR_BUFFER_SIZE`
 - `record.c`, [97](#)
- `CLK_STATE`
 - `rtc_ds3231.c`, [24](#)
- `CONFIG_FILE`
 - `defs.h`, [41](#)
- `CONSOLELOG_CheckFileSystem`
 - `record.c`, [97](#)
 - `record.h`, [67](#)
- `CONSOLELOG_ClearTransferredBytes`
 - `record.c`, [97](#)
 - `record.h`, [67](#)
- `CONSOLELOG_CreateDirectory`
 - `record.c`, [98](#)
 - `record.h`, [67](#)
- `CONSOLELOG_CreateFile`
 - `record.c`, [98](#)
 - `record.h`, [67](#)
- `CONSOLELOG_Deinit`
 - `record.c`, [98](#)
 - `record.h`, [67](#)
- `CONSOLELOG_Flush`
 - `record.c`, [98](#)
 - `record.h`, [68](#)
- `CONSOLELOG_GetFlushCompleted`
 - `record.c`, [99](#)
 - `record.h`, [68](#)
- `CONSOLELOG_GetFreeSpaceMB`
 - `record.c`, [99](#)
 - `record.h`, [68](#)
- `CONSOLELOG_GetMaxBytes`
 - `record.h`, [68](#)
- `CONSOLELOG_GetTransferredBytes`
 - `record.c`, [99](#)

- record.h, [69](#)
- CONSOLELOG_Init
 - record.c, [99](#)
 - record.h, [69](#)
- CONSOLELOG_PowerLossFlush
 - record.c, [100](#)
 - record.h, [69](#)
- CONSOLELOG_ProccessConfigFile
 - record.c, [100](#)
 - record.h, [69](#)
- CONSOLELOG_ReadConfig
 - record.c, [100](#)
 - record.h, [70](#)
- CONSOLELOG_Recording
 - record.c, [101](#)
 - record.h, [70](#)
- CONTROL_LED_ENABLED
 - defs.h, [41](#)
- CTIMER
 - pwrloss_det.h, [63](#)
- CTIMER0_IRQHandler
 - temperature.c, [106](#)
- CTIMER4_IRQHandler
 - pwrloss_det.c, [94](#)
- CTIMER_CLK_FREQ
 - pwrloss_det.h, [63](#)
- CTIMER_EMT0_OUT
 - pwrloss_det.h, [63](#)
- CTIMER_MAT0_OUT
 - pwrloss_det.h, [64](#)
- data_bits
 - REC_config_t, [19](#)
- date
 - RTC_date_t, [21](#)
- day
 - RTC_date_t, [21](#)
- DEBUG_ENABLED
 - defs.h, [41](#)
- DEFAULT_BAUDRATE
 - defs.h, [42](#)
- DEFAULT_DATA_BITS
 - defs.h, [42](#)
- DEFAULT_FREE_SPACE
 - defs.h, [42](#)
- DEFAULT_MAX_FILESIZE
 - defs.h, [42](#)
- DEFAULT_PARITY
 - defs.h, [42](#)
- DEFAULT_STOP_BITS
 - defs.h, [42](#)
- defs.h
 - CONFIG_FILE, [41](#)
 - CONTROL_LED_ENABLED, [41](#)
 - DEBUG_ENABLED, [41](#)
 - DEFAULT_BAUDRATE, [42](#)
 - DEFAULT_DATA_BITS, [42](#)
 - DEFAULT_FREE_SPACE, [42](#)
 - DEFAULT_MAX_FILESIZE, [42](#)
 - DEFAULT_PARITY, [42](#)
 - DEFAULT_STOP_BITS, [42](#)
 - INFO_ENABLED, [42](#)
 - MSC_ENABLED, [43](#)
 - MSC_STACK_SIZE, [43](#)
 - NOT_IMPLEMENTED, [43](#)
 - PWRLOSS_DET_ACTIVE_IN_TIME, [43](#)
 - PWRLOSS_DET_PRIO, [43](#)
 - PWRLOSS_DETECTION_ENABLED, [43](#)
 - PWRLOSS_TEST_GPIOS, [43](#)
 - PWRLOSS_TIMER_PRIO, [44](#)
 - RECORD_STACK_SIZE, [44](#)
 - TAU, [44](#)
 - TEMPERATURE_MEAS_ENABLED, [44](#)
 - UART_FIFO_ENABLED, [44](#)
 - UART_FIFO_LENHT, [44](#)
 - UART_PRINT_ENABLED, [44](#)
 - UART_RECEIVE_PRIO, [45](#)
- DEMO_LPCMP_BASE
 - pwrloss_det.h, [64](#)
- DEMO_LPCMP_DAC_CHANNEL
 - pwrloss_det.h, [64](#)
- DEMO_LPCMP_IRQ_HANDLER_FUNC
 - pwrloss_det.h, [64](#)
- DEMO_LPCMP_IRQ_ID
 - pwrloss_det.h, [64](#)
- DEMO_LPCMP_USER_CHANNEL
 - pwrloss_det.h, [64](#)
- DEMO_SPC_BASE
 - pwrloss_det.h, [64](#)
- DEMO_VREF_BASE
 - pwrloss_det.h, [64](#)
- drivers/rtc_ds3231.c, [23](#)
- drivers/rtc_ds3231.h, [26](#), [33](#)
- DS3231 Real-Time Clock Driver, [12](#)
 - RTC_ClearFlagAlarm1, [13](#)
 - RTC_ClearFlagAlarm2, [13](#)
 - RTC_ConvertToBCD, [13](#)
 - RTC_ConvertToDEC, [14](#)
 - RTC_CtrlAlarm1, [14](#)
 - RTC_CtrlAlarm2, [14](#)
 - RTC_Deinit, [14](#)
 - RTC_GetDate, [15](#)
 - RTC_GetState, [15](#)
 - RTC_GetTime, [15](#)
 - RTC_Init, [15](#)
 - RTC_Read, [16](#)
 - RTC_SetDate, [16](#)
 - RTC_SetDateDefault, [16](#)
 - RTC_SetInterruptMode, [16](#)
 - RTC_SetOscState, [17](#)
 - RTC_SetTime, [17](#)
 - RTC_SetTimeDefault, [17](#)
 - RTC_Write, [17](#)
- DS3231_A1F
 - rtc_ds3231.h, [28](#)
- DS3231_A1IE
 - rtc_ds3231.h, [29](#)

- DS3231_A2F
 - rtc_ds3231.h, [29](#)
- DS3231_A2IE
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_CENT
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_DATE
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_DAY
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_HRS
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_I2C
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_MIN
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_MONTH
 - rtc_ds3231.h, [29](#)
- DS3231_ADDR_SEC
 - rtc_ds3231.h, [30](#)
- DS3231_ADDR_YEAR
 - rtc_ds3231.h, [30](#)
- DS3231_INTCN
 - rtc_ds3231.h, [30](#)
- DS3231_REG_CTRL
 - rtc_ds3231.h, [30](#)
- DS3231_REG_STATUS
 - rtc_ds3231.h, [30](#)
- E_FAULT
 - rtc_ds3231.h, [30](#)
- ERR_HandleError
 - error.c, [82](#)
 - error.h, [48](#)
- ERR_Init
 - error.c, [82](#)
 - error.h, [48](#)
- ERR_SetState
 - error.c, [82](#)
 - error.h, [49](#)
- error
 - error.c, [83](#)
- error.c
 - ERR_HandleError, [82](#)
 - ERR_Init, [82](#)
 - ERR_SetState, [82](#)
 - error, [83](#)
- error.h
 - ERR_HandleError, [48](#)
 - ERR_Init, [48](#)
 - ERR_SetState, [49](#)
 - ERROR_ADMA, [47](#)
 - ERROR_CLOSE, [47](#)
 - ERROR_CONFIG, [47](#)
 - ERROR_FILESYSTEM, [47](#)
 - ERROR_IRTC, [47](#)
 - ERROR_NONE, [47](#)
 - ERROR_OPEN, [47](#)
 - ERROR_OUT_OF_CYCLE, [48](#)
 - ERROR_READ, [48](#)
 - ERROR_RECORD, [48](#)
 - error_t, [48](#)
 - ERROR_UNKNOWN, [48](#)
 - EXTERNAL_RTC_TIME_OFFSET, [108](#)
 - FLUSH_LED_PORT, [51](#)
 - FLUSH_TIMEOUT_TICKS, [67](#)
 - format
 - RTC_time_t, [22](#)
 - free_space_limit_mb
 - REC_config_t, [20](#)
 - FRIDAY
 - rtc_ds3231.h, [30](#)
 - g_bBackDmaBufferReady
 - Buffers and Recording Management, [9](#)
 - g_bCompletionFlag_I2C5
 - temperature.c, [107](#)
 - g_bFlushCompleted
 - Buffers and Recording Management, [9](#)
 - g_config
 - parser.c, [93](#)
 - g_EdmaRxHandle_I2C5
 - temperature.c, [107](#)
 - g_EdmaTxHandle_I2C5
 - temperature.c, [107](#)
 - g_fileObject

- Buffers and Recording Management, 9
- g_fileSystem
 - Buffers and Recording Management, 10
- g_lastDataTick
 - Buffers and Recording Management, 10
- g_msc
 - main.c, 85
 - task_switching.h, 72
- g_pu8BackDmaBuffer
 - Buffers and Recording Management, 10
- g_pu8FrontDmaBuffer
 - Buffers and Recording Management, 10
- g_u16BackDmaBufferIdx
 - Buffers and Recording Management, 10
- g_u32BytesTransferred
 - Buffers and Recording Management, 10
- g_u32CurrentFileSize
 - Buffers and Recording Management, 10
- g_u32FileCounter
 - Buffers and Recording Management, 11
- g_u32ReadIndex
 - Management, 12
- g_u32WriteIndex
 - Management, 12
- g_u8CurrentDirectory
 - Buffers and Recording Management, 11
- g_Xfer_I2C5
 - temperature.c, 107
- g_xMscTaskHandle
 - app_tasks.h, 39
 - main.c, 85
- g_xMscTaskStack
 - main.c, 86
- g_xMscTaskTCB
 - main.c, 86
- g_xRecordTaskHandle
 - app_tasks.h, 39
 - main.c, 86
- g_xRecordTaskStack
 - main.c, 86
- g_xRecordTaskTCB
 - main.c, 86
- g_xSemMassStorage
 - app_tasks.h, 39
 - main.c, 86
 - mass_storage.c, 88
- g_xSemRecord
 - app_tasks.h, 39
 - main.c, 86
 - mass_storage.c, 88
- gCompletionFlag
 - rtc_ds3231.c, 26
- gEdmaRxHandle
 - rtc_ds3231.c, 26
- gEdmaTxHandle
 - rtc_ds3231.c, 26
- GET_CURRENT_TIME_MS
 - record.c, 97
- get_fatime
 - record.c, 101
- GET_WAIT_INTERVAL
 - record.c, 97
- hrs
 - RTC_time_t, 22
- HSCMP1_IRQHandler
 - pwrloss_det.c, 94
- I2C_BAUDRATE
 - rtc_ds3231.h, 30
 - temperature.c, 104
- I2C_DATA_LENGTH
 - rtc_ds3231.c, 25
- I2C_MASTER
 - rtc_ds3231.h, 30
- I2C_MASTER_I2C5
 - temperature.c, 104
- include/app_init.h, 35, 36
- include/app_tasks.h, 37, 39
- include/defs.h, 40, 45
- include/error.h, 46, 49
- include/led.h, 50, 53
- include/mass_storage.h, 54, 55
- include/parser.h, 56, 62
- include/pwrloss_det.h, 63, 65
- include/record.h, 65, 71
- include/task_switching.h, 72, 73
- include/temperature.h, 73, 75
- include/time.h, 75, 76
- include/uart.h, 77, 79
- INFO_ENABLED
 - defs.h, 42
- INTERNAL_RTC_BASE_YEAR
 - time.c, 108
- led.c
 - LED_ClearSignalFlush, 83
 - LED_SetHigh, 83
 - LED_SetLow, 83
 - LED_SignalBackUpPowerAvailable, 83
 - LED_SignalError, 84
 - LED_SignalFlush, 84
 - LED_SignalLowMemory, 84
 - LED_SignalReady, 84
 - LED_SignalRecording, 84
 - LED_SignalRecordingStop, 84
- led.h
 - BACKUP_POWER_LED_PIN, 50
 - BACKUP_POWER_LED_PORT, 50
 - ERROR_LED_PIN_RECORD, 50
 - ERROR_LED_PORT, 50
 - FLUSH_LED_PORT, 51
 - LED_ClearSignalFlush, 51
 - LED_SetHigh, 51
 - LED_SetLow, 51
 - LED_SignalBackUpPowerAvailable, 52
 - LED_SignalError, 52

- LED_SignalFlush, [52](#)
- LED_SignalLowMemory, [52](#)
- LED_SignalReady, [52](#)
- LED_SignalRecording, [52](#)
- LED_SignalRecordingStop, [52](#)
- MEMORY_LOW_LED_PIN, [51](#)
- MEMORY_LOW_LED_PORT, [51](#)
- RECORD_LED_PIN, [51](#)
- RECORD_LED_PIN_FLUSH, [51](#)
- RECORD_LED_PORT, [51](#)
- LED_ClearSignalFlush
 - led.c, [83](#)
 - led.h, [51](#)
- LED_SetHigh
 - led.c, [83](#)
 - led.h, [51](#)
- LED_SetLow
 - led.c, [83](#)
 - led.h, [51](#)
- LED_SignalBackUpPowerAvailable
 - led.c, [83](#)
 - led.h, [52](#)
- LED_SignalError
 - led.c, [84](#)
 - led.h, [52](#)
- LED_SignalFlush
 - led.c, [84](#)
 - led.h, [52](#)
- LED_SignalLowMemory
 - led.c, [84](#)
 - led.h, [52](#)
- LED_SignalReady
 - led.c, [84](#)
 - led.h, [52](#)
- LED_SignalRecording
 - led.c, [84](#)
 - led.h, [52](#)
- LED_SignalRecordingStop
 - led.c, [84](#)
 - led.h, [52](#)
- LP_FLEXCOMM3_IRQHandler
 - record.c, [101](#)
- lpi2c_callback
 - rtc_ds3231.c, [25](#)
 - temperature.c, [106](#)
- LPI2C_DMA_BASEADDR
 - time.h, [76](#)
- LPI2C_RX_DMA_CHANNEL
 - rtc_ds3231.h, [31](#)
 - temperature.c, [104](#)
- LPI2C_RX_EDMA_CHANNEL
 - rtc_ds3231.h, [31](#)
 - temperature.c, [104](#)
- LPI2C_TX_CHANNEL
 - rtc_ds3231.h, [31](#)
 - temperature.c, [104](#)
- LPI2C_TX_DMA_CHANNEL
 - rtc_ds3231.h, [31](#)
- temperature.c, [105](#)
- LPUART3_CLK_FREQ
 - uart.h, [77](#)
- main
 - main.c, [85](#)
- main.c
 - g_msc, [85](#)
 - g_xMscTaskHandle, [85](#)
 - g_xMscTaskStack, [86](#)
 - g_xMscTaskTCB, [86](#)
 - g_xRecordTaskHandle, [86](#)
 - g_xRecordTaskStack, [86](#)
 - g_xRecordTaskTCB, [86](#)
 - g_xSemMassStorage, [86](#)
 - g_xSemRecord, [86](#)
 - main, [85](#)
- Management, [11](#)
 - g_u32ReadIndex, [12](#)
 - g_u32WriteIndex, [12](#)
 - SDK_ALIGN, [12](#)
- mass_storage.c
 - g_xSemMassStorage, [88](#)
 - g_xSemRecord, [88](#)
 - MSC_DeviceMscApp, [87](#)
 - MSC_DeviceMscAppTask, [87](#)
 - USB1_HS_IRQHandler, [87](#)
- mass_storage.h
 - MSC_DeviceMscApp, [55](#)
 - MSC_DeviceMscAppTask, [55](#)
- max_bytes
 - REC_config_t, [20](#)
- MEMORY_LOW_LED_PIN
 - led.h, [51](#)
- MEMORY_LOW_LED_PORT
 - led.h, [51](#)
- min
 - RTC_time_t, [22](#)
- MONDAY
 - rtc_ds3231.h, [31](#)
- month
 - RTC_date_t, [21](#)
- MSC_DeviceMscApp
 - mass_storage.c, [87](#)
 - mass_storage.h, [55](#)
- MSC_DeviceMscAppTask
 - mass_storage.c, [87](#)
 - mass_storage.h, [55](#)
- MSC_ENABLED
 - defs.h, [43](#)
- MSC_STACK_SIZE
 - defs.h, [43](#)
- msc_task
 - app_tasks.c, [81](#)
 - app_tasks.h, [37](#)
- NOT_IMPLEMENTED
 - defs.h, [43](#)

- OSC_INTERRUPTED
 - rtc_ds3231.h, 33
- OSC_OK
 - rtc_ds3231.h, 33
- OSC_STOPPED
 - rtc_ds3231.h, 31
- OSF_STATE
 - rtc_ds3231.c, 25
- P3T1755_ADDR_7BIT
 - temperature.c, 105
- parity
 - REC_config_t, 20
- parser.c
 - g_config, 93
 - PARSER_ClearConfig, 89
 - PARSER_GetBaudrate, 89
 - PARSER_GetConfig, 89
 - PARSER_GetDataBits, 90
 - PARSER_GetFileSize, 90
 - PARSER_GetFreeSpaceLimitMB, 90
 - PARSER_GetMaxBytes, 90
 - PARSER_GetParity, 91
 - PARSER_GetStopBits, 91
 - PARSER_GetVersion, 91
 - PARSER_ParseBaudrate, 91
 - PARSER_ParseDataBits, 92
 - PARSER_ParseFileSize, 92
 - PARSER_ParseFreeSpace, 92
 - PARSER_ParseParity, 93
 - PARSER_ParseStopBits, 93
 - RECORD_LED_TIME_INTERVAL, 89
- parser.h
 - PARSER_ClearConfig, 57
 - PARSER_GetBaudrate, 57
 - PARSER_GetConfig, 57
 - PARSER_GetDataBits, 58
 - PARSER_GetFileSize, 58
 - PARSER_GetFreeSpaceLimitMB, 58
 - PARSER_GetMaxBytes, 58
 - PARSER_GetParity, 59
 - PARSER_GetStopBits, 59
 - PARSER_GetVersion, 59
 - PARSER_ParseBaudrate, 59
 - PARSER_ParseDataBits, 60
 - PARSER_ParseFileSize, 60
 - PARSER_ParseFreeSpace, 60
 - PARSER_ParseParity, 61
 - PARSER_ParseStopBits, 61
 - REC_version_t, 57
 - WCT_AUTOS1, 57
 - WCT_AUTOS2, 57
 - WCT_UNKOWN, 57
- PARSER_ClearConfig
 - parser.c, 89
 - parser.h, 57
- PARSER_GetBaudrate
 - parser.c, 89
 - parser.h, 57
- PARSER_GetConfig
 - parser.c, 89
 - parser.h, 57
- PARSER_GetDataBits
 - parser.c, 90
 - parser.h, 58
- PARSER_GetFileSize
 - parser.c, 90
 - parser.h, 58
- PARSER_GetFreeSpaceLimitMB
 - parser.c, 90
 - parser.h, 58
- PARSER_GetMaxBytes
 - parser.c, 90
 - parser.h, 58
- PARSER_GetParity
 - parser.c, 91
 - parser.h, 59
- PARSER_GetStopBits
 - parser.c, 91
 - parser.h, 59
- PARSER_GetVersion
 - parser.c, 91
 - parser.h, 59
- PARSER_ParseBaudrate
 - parser.c, 91
 - parser.h, 59
- PARSER_ParseDataBits
 - parser.c, 92
 - parser.h, 60
- PARSER_ParseFileSize
 - parser.c, 92
 - parser.h, 60
- PARSER_ParseFreeSpace
 - parser.c, 92
 - parser.h, 60
- PARSER_ParseParity
 - parser.c, 93
 - parser.h, 61
- PARSER_ParseStopBits
 - parser.c, 93
 - parser.h, 61
- PRINT_REG_OUTPUT
 - temperature.c, 105
- pwrloss_det.c
 - CTIMER4_IRQHandler, 94
 - HSCMP1_IRQHandler, 94
 - PWRLOSS_DetectionInit, 94
 - REF_VOLTAGE, 94
 - TRIGGER_VOLTAGE, 94
- pwrloss_det.h
 - CTIMER, 63
 - CTIMER_CLK_FREQ, 63
 - CTIMER_EMT0_OUT, 63
 - CTIMER_MAT0_OUT, 64
 - DEMO_LPCMP_BASE, 64
 - DEMO_LPCMP_DAC_CHANNEL, 64
 - DEMO_LPCMP_IRQ_HANDLER_FUNC, 64

- DEMO_LPCMP_IRQ_ID, 64
- DEMO_LPCMP_USER_CHANNEL, 64
- DEMO_SPC_BASE, 64
- DEMO_VREF_BASE, 64
- PWRLOSS_DetectionInit, 64
- PWRLOSS_DET_ACTIVE_IN_TIME
 - defs.h, 43
- PWRLOSS_DET_PRIO
 - defs.h, 43
- PWRLOSS_DETECTION_ENABLED
 - defs.h, 43
- PWRLOSS_DetectionInit
 - pwrloss_det.c, 94
 - pwrloss_det.h, 64
- PWRLOSS_TEST_GPIO
 - defs.h, 43
- PWRLOSS_TIMER_PRIO
 - defs.h, 44
- Read
 - temperature.c, 106
 - temperature.h, 74
- REC_config_t, 19
 - baudrate, 19
 - data_bits, 19
 - free_space_limit_mb, 20
 - max_bytes, 20
 - parity, 20
 - size, 20
 - stop_bits, 20
 - version, 20
- REC_version_t
 - parser.h, 57
- record.c
 - BLOCK_SIZE, 97
 - CIRCULAR_BUFFER_SIZE, 97
 - CONSOLELOG_CheckFileSystem, 97
 - CONSOLELOG_ClearTransferredBytes, 97
 - CONSOLELOG_CreateDirectory, 98
 - CONSOLELOG_CreateFile, 98
 - CONSOLELOG_Deinit, 98
 - CONSOLELOG_Flush, 98
 - CONSOLELOG_GetFlushCompleted, 99
 - CONSOLELOG_GetFreeSpaceMB, 99
 - CONSOLELOG_GetTransferredBytes, 99
 - CONSOLELOG_Init, 99
 - CONSOLELOG_PowerLossFlush, 100
 - CONSOLELOG_ProccessConfigFile, 100
 - CONSOLELOG_ReadConfig, 100
 - CONSOLELOG_Recording, 101
 - GET_CURRENT_TIME_MS, 97
 - get_fatime, 101
 - GET_WAIT_INTERVAL, 97
 - LP_FLEXCOMM3_IRQHandler, 101
- record.h
 - CONSOLELOG_CheckFileSystem, 67
 - CONSOLELOG_ClearTransferredBytes, 67
 - CONSOLELOG_CreateDirectory, 67
 - CONSOLELOG_CreateFile, 67
 - CONSOLELOG_Deinit, 67
 - CONSOLELOG_Flush, 68
 - CONSOLELOG_GetFlushCompleted, 68
 - CONSOLELOG_GetFreeSpaceMB, 68
 - CONSOLELOG_GetMaxBytes, 68
 - CONSOLELOG_GetTransferredBytes, 69
 - CONSOLELOG_Init, 69
 - CONSOLELOG_PowerLossFlush, 69
 - CONSOLELOG_ProccessConfigFile, 69
 - CONSOLELOG_ReadConfig, 70
 - CONSOLELOG_Recording, 70
 - FLUSH_TIMEOUT_TICKS, 67
- RECORD_LED_PIN
 - led.h, 51
- RECORD_LED_PIN_FLUSH
 - led.h, 51
- RECORD_LED_PORT
 - led.h, 51
- RECORD_LED_TIME_INTERVAL
 - parser.c, 89
- RECORD_STACK_SIZE
 - defs.h, 44
- record_task
 - app_tasks.c, 81
 - app_tasks.h, 38
- REF_VOLTAGE
 - pwrloss_det.c, 94
- REGISTER_CONFIG
 - temperature.c, 105
- REGISTER_TEMPERATURE
 - temperature.c, 105
- REGISTER_THVST
 - temperature.c, 105
- REGISTER_TOS
 - temperature.c, 105
- RTC_ClearFlagAlarm1
 - DS3231 Real-Time Clock Driver, 13
- RTC_ClearFlagAlarm2
 - DS3231 Real-Time Clock Driver, 13
- RTC_ConvertToBCD
 - DS3231 Real-Time Clock Driver, 13
 - rtc_ds3231.c, 25
- RTC_ConvertToDEC
 - DS3231 Real-Time Clock Driver, 14
 - rtc_ds3231.c, 25
- RTC_CtrlAlarm1
 - DS3231 Real-Time Clock Driver, 14
- RTC_CtrlAlarm2
 - DS3231 Real-Time Clock Driver, 14
- RTC_date_t, 21
 - date, 21
 - day, 21
 - month, 21
 - year, 21
- RTC_Deinit
 - DS3231 Real-Time Clock Driver, 14
- rtc_ds3231.c
 - AT_NONCACHEABLE_SECTION, 25

- BYTE_1, [24](#)
- BYTE_2, [24](#)
- CLK_STATE, [24](#)
- gCompletionFlag, [26](#)
- gEdmaRxHandle, [26](#)
- gEdmaTxHandle, [26](#)
- I2C_DATA_LENGTH, [25](#)
- lpi2c_callback, [25](#)
- OSF_STATE, [25](#)
- RTC_ConvertToBCD, [25](#)
- RTC_ConvertToDEC, [25](#)
- xfer, [26](#)
- rtc_ds3231.h
 - ALARM_DISABLED, [28](#)
 - ALARM_INTERRUPT, [32](#)
 - APP_SUCCESS, [28](#)
 - DS3231_A1F, [28](#)
 - DS3231_A1IE, [29](#)
 - DS3231_A2F, [29](#)
 - DS3231_A2IE, [29](#)
 - DS3231_ADDR_CENT, [29](#)
 - DS3231_ADDR_DATE, [29](#)
 - DS3231_ADDR_DAY, [29](#)
 - DS3231_ADDR_HRS, [29](#)
 - DS3231_ADDR_I2C, [29](#)
 - DS3231_ADDR_MIN, [29](#)
 - DS3231_ADDR_MONTH, [29](#)
 - DS3231_ADDR_SEC, [30](#)
 - DS3231_ADDR_YEAR, [30](#)
 - DS3231_INTCN, [30](#)
 - DS3231_REG_CTRL, [30](#)
 - DS3231_REG_STATUS, [30](#)
 - E_FAULT, [30](#)
 - FRIDAY, [30](#)
 - I2C_BAUDRATE, [30](#)
 - I2C_MASTER, [30](#)
 - LPI2C_RX_DMA_CHANNEL, [31](#)
 - LPI2C_RX_EDMA_CHANNEL, [31](#)
 - LPI2C_TX_CHANNEL, [31](#)
 - LPI2C_TX_DMA_CHANNEL, [31](#)
 - MONDAY, [31](#)
 - OSC_INTERRUPTED, [33](#)
 - OSC_OK, [33](#)
 - OSC_STOPPED, [31](#)
 - RTC_interrupt_mode_t, [32](#)
 - RTC_osc_state_t, [32](#)
 - SATURDAY, [31](#)
 - SQUARE_WAVE_INTERRUPT, [32](#)
 - SUNDAY, [31](#)
 - THURSDAY, [32](#)
 - TIM_CYCLE_12H, [32](#)
 - TIM_CYCLE_12H_AM, [32](#)
 - TIM_CYCLE_12H_PM, [32](#)
 - TIM_CYCLE_24H, [32](#)
 - TUESDAY, [32](#)
 - WEDNESDAY, [32](#)
- RTC_GetDate
 - DS3231 Real-Time Clock Driver, [15](#)
- RTC_GetState
 - DS3231 Real-Time Clock Driver, [15](#)
- RTC_GetTime
 - DS3231 Real-Time Clock Driver, [15](#)
- RTC_Init
 - DS3231 Real-Time Clock Driver, [15](#)
- RTC_interrupt_mode_t
 - rtc_ds3231.h, [32](#)
- RTC_osc_state_t
 - rtc_ds3231.h, [32](#)
- RTC_Read
 - DS3231 Real-Time Clock Driver, [16](#)
- RTC_SetDate
 - DS3231 Real-Time Clock Driver, [16](#)
- RTC_SetDateDefault
 - DS3231 Real-Time Clock Driver, [16](#)
- RTC_SetInterruptMode
 - DS3231 Real-Time Clock Driver, [16](#)
- RTC_SetOscState
 - DS3231 Real-Time Clock Driver, [17](#)
- RTC_SetTime
 - DS3231 Real-Time Clock Driver, [17](#)
- RTC_SetTimeDefault
 - DS3231 Real-Time Clock Driver, [17](#)
- RTC_time_t, [22](#)
 - format, [22](#)
 - hrs, [22](#)
 - min, [22](#)
 - sec, [22](#)
- RTC_Write
 - DS3231 Real-Time Clock Driver, [17](#)
- SATURDAY
 - rtc_ds3231.h, [31](#)
- SDK_ALIGN
 - Buffers and Recording Management, [9](#)
 - Management, [12](#)
- sec
 - RTC_time_t, [22](#)
- semihost_hardfault.c
 - __attribute__, [102](#)
- SENSOR_DYNAMIC_ADDR
 - temperature.h, [74](#)
- SENSOR_STATIC_ADDR
 - temperature.h, [74](#)
- size
 - REC_config_t, [20](#)
- source/app_init.c, [79](#)
- source/app_tasks.c, [80](#)
- source/error.c, [82](#)
- source/led.c, [83](#)
- source/main.c, [84](#)
- source/mass_storage.c, [87](#)
- source/parser.c, [88](#)
- source/pwrloss_det.c, [94](#)
- source/record.c, [95](#)
- source/semihost_hardfault.c, [102](#)
- source/task_switching.c, [102](#)
- source/temperature.c, [103](#)

- source/time.c, [108](#)
- source/uart.c, [109](#)
- SQUARE_WAVE_INTERRUPT
 - rtc_ds3231.h, [32](#)
- stop_bits
 - REC_config_t, [20](#)
- SUNDAY
 - rtc_ds3231.h, [31](#)
- Task Management, [7](#)
 - uxIdleTaskStack, [7](#)
 - uxTimerTaskStack, [7](#)
 - xIdleTaskTCB, [8](#)
 - xTimerTaskTCB, [8](#)
- TASK_PRIO
 - app_tasks.h, [37](#)
- task_switching.c
 - USB_State, [102](#)
- task_switching.h
 - g_msc, [72](#)
 - USB_State, [72](#)
- TAU
 - defs.h, [44](#)
- temperature.c
 - AT_NONCACHEABLE_SECTION, [106](#)
 - BUFF_SIZE, [104](#)
 - CTIMER0_IRQHandler, [106](#)
 - g_bCompletionFlag_I2C5, [107](#)
 - g_EdmaRxHandle_I2C5, [107](#)
 - g_EdmaTxHandle_I2C5, [107](#)
 - g_Xfer_I2C5, [107](#)
 - I2C_BAUDRATE, [104](#)
 - I2C_MASTER_I2C5, [104](#)
 - lpi2c_callback, [106](#)
 - LPI2C_RX_DMA_CHANNEL, [104](#)
 - LPI2C_RX_EDMA_CHANNEL, [104](#)
 - LPI2C_TX_CHANNEL, [104](#)
 - LPI2C_TX_DMA_CHANNEL, [105](#)
 - P3T1755_ADDR_7BIT, [105](#)
 - PRINT_REG_OUTPUT, [105](#)
 - Read, [106](#)
 - REGISTER_CONFIG, [105](#)
 - REGISTER_TEMPERATURE, [105](#)
 - REGISTER_THVST, [105](#)
 - REGISTER_TOS, [105](#)
 - TMP_GetTemperature, [106](#)
 - TMP_Init, [107](#)
 - Write, [107](#)
- temperature.h
 - Read, [74](#)
 - SENSOR_DYNAMIC_ADDR, [74](#)
 - SENSOR_STATIC_ADDR, [74](#)
 - TMP_GetTemperature, [74](#)
 - TMP_Init, [74](#)
 - Write, [74](#)
- TEMPERATURE_MEAS_ENABLED
 - defs.h, [44](#)
- THURSDAY
 - rtc_ds3231.h, [32](#)
- TIM_CYCLE_12H
 - rtc_ds3231.h, [32](#)
- TIM_CYCLE_12H_AM
 - rtc_ds3231.h, [32](#)
- TIM_CYCLE_12H_PM
 - rtc_ds3231.h, [32](#)
- TIM_CYCLE_24H
 - rtc_ds3231.h, [32](#)
- time.c
 - EXTERNAL_RTC_TIME_OFFSET, [108](#)
 - INTERNAL_RTC_BASE_YEAR, [108](#)
 - TIME_InitIRTC, [108](#)
- time.h
 - LPI2C_DMA_BASEADDR, [76](#)
 - TIME_InitIRTC, [76](#)
- TIME_InitIRTC
 - time.c, [108](#)
 - time.h, [76](#)
- TMP_GetTemperature
 - temperature.c, [106](#)
 - temperature.h, [74](#)
- TMP_Init
 - temperature.c, [107](#)
 - temperature.h, [74](#)
- TRIGGER_VOLTAGE
 - pwrloss_det.c, [94](#)
- TUESDAY
 - rtc_ds3231.h, [32](#)
- uart.c
 - BUFFER_SIZE, [109](#)
 - UART_Deinit, [109](#)
 - UART_Disable, [109](#)
 - UART_Enable, [109](#)
 - UART_Init, [110](#)
 - UART_Print, [110](#)
- uart.h
 - LPUART3_CLK_FREQ, [77](#)
 - UART_Deinit, [78](#)
 - UART_Disable, [78](#)
 - UART_Enable, [78](#)
 - UART_Init, [78](#)
 - UART_Print, [78](#)
- UART_Deinit
 - uart.c, [109](#)
 - uart.h, [78](#)
- UART_Disable
 - uart.c, [109](#)
 - uart.h, [78](#)
- UART_Enable
 - uart.c, [109](#)
 - uart.h, [78](#)
- UART_FIFO_ENABLED
 - defs.h, [44](#)
- UART_FIFO_LENGTH
 - defs.h, [44](#)
- UART_Init
 - uart.c, [110](#)
 - uart.h, [78](#)

- UART_Print
 - uart.c, [110](#)
 - uart.h, [78](#)
- UART_PRINT_ENABLED
 - defs.h, [44](#)
- UART_RECEIVE_PRIO
 - defs.h, [45](#)
- USB1_HS_IRQHandler
 - mass_storage.c, [87](#)
- USB_State
 - task_switching.c, [102](#)
 - task_switching.h, [72](#)
- uxIdleTaskStack
 - Task Management, [7](#)
- uxTimerTaskStack
 - Task Management, [7](#)
- vApplicationGetIdleTaskMemory
 - app_tasks.c, [81](#)
 - app_tasks.h, [38](#)
- vApplicationGetTimerTaskMemory
 - app_tasks.c, [81](#)
 - app_tasks.h, [38](#)
- version
 - REC_config_t, [20](#)
- WCT_AUTOS1
 - parser.h, [57](#)
- WCT_AUTOS2
 - parser.h, [57](#)
- WCT_UNKOWN
 - parser.h, [57](#)
- WEDNESDAY
 - rtc_ds3231.h, [32](#)
- Write
 - temperature.c, [107](#)
 - temperature.h, [74](#)
- xfer
 - rtc_ds3231.c, [26](#)
- xIdleTaskTCB
 - Task Management, [8](#)
- xTimerTaskTCB
 - Task Management, [8](#)
- year
 - RTC_date_t, [21](#)