app_usart_to_extender.h File Reference

application for USART to extender More...

Go to the source code of this file.

Data Structures

struct	**uart_msg_to_ext** Structure definition for UART msg object. More...
#define	**D_INVALID_BYTE** 0xFF
#define	**D_INVALID_QWORD** 0xFFFFFFFF
#define	**D_PROT_MESSAGE_VERSION** 0x01
#define	**D_PARAM_LEN** 0xFF
#define	**D_MinRxTelegramLen** (**PosPayloadData** + 2)
#define	**D_FlagHdrStart** 0x68
#define	**D_FlagHdrEnd** 0x69

enum	**eCmdId** {
	Cmd_PV_Heating = 1, **Cmd_PV_ActualSpd** = 309, **Cmd_PV_DrvCurrent** = 310, **Cmd_PV_OpHrsPump** = 311,
	Cmd_PV_OpHrsElec = 314, **Cmd_PV_TempElec** = 326, **Cmd_QL_StatusCCB** = 1024, **Cmd_MB_Status** = 1280,
	Cmd_MB_EnablePump = 1281, **Cmd_MB_LedOnOff** = 1282
	}
	Enum for 16bit command identifier in range 0..65535 (0x0 .. 0xFFFF) More...

enum	**eRxBytePos** {
	PosHdrFlagStart = 0, **PosHdrLen** = 1, **PosHdrVersion** = 3, **PosHdrRcvAddress** = 4,
	PosHdrOcd = 5, **PosHrdSeq** = 6, **PosHdrFlagEnd** = 10, **PosPayloadData** = 11
	}
	Enum for UART receive byte positions. More...

enum	**eMsgStates** {
	StHdrFlagStart = 0, **StHdrLen** = 1, **StHdrVersion** = 2, **StHdrDstAddress** = 3,
	StHdrOcd = 4, **StHrdSeq** = 5, **StHdrFlagEnd** = 6, **StPayloadData** = 7,
	StCrcCalc = 8, **StError** = 9
	}
	Enum for states of message interpretation. More...

enum	**eOcdType** {
	OCD_GET = 0, **OCD_MGET** = 1, **OCD_SET** = 2, **OCD_MSET** = 3,
	OCD_ACK = 4, **OCD_NAK** = 5, **OCD_DAT** = 6
	}
	enumeration for type of OCD (opcode) More...

enum	**eDataType** {
	DTYPE_BOOL_OLD = 0, **DTYPE_UINT** = 1, **DTYPE_REAL** = 2, **DTYPE_STRING** = 4,
	DTYPE_BOOL_NEW = 6, **DTYPE_USHORT** = 7, **DTYPE_EXPO_N** = 10
	}
	enumeration for data types More...

enum	**eAddrType** { **ADDR_IOTB** = 0, **ADDR_CCB** = 1 } enumeration for valid address types More...
typedef struct **uart_msg_to_ext**	**sUartMsg2Ext** Structure definition for UART msg object. More...
void	**uart_init** (void) UART initialization. More...

void	**uart_init_msg_obj** (**sUartMsg2Ext** *pMsg2Ext) UART initialization of msg object. More...
sUartMsg2Ext *	**uart_get_msg_obj** (void) get message object for user protocol handling More...
uint16_t	**uart_test_get_crc** (uint8_t idx, uint16_t *crc, bool *is_valid) UART test function: get crc of buffer of given index. More...
uint16_t	**crc_calc** (char *buf, uint16_t data_byte_len) calculate the CRC of given buffer with given data length in bytes More...
bool	**uart_is_valid_msg_version** (byte_t version_id) check for valid message version of telegram More...
bool	**uart_is_valid_msg_address** (byte_t dst_addr_id) check for valid message address in telegram More...
bool	**uart_is_valid_ocd** (byte_t ocd_id) check for valid OCD id in telegram More...
int	**uart_binary_data_interpreter** (byte_t *data, uint16_t data_len, **sUartMsg2Ext** *pMsg2Ext) UART binary data interpreter. More...

Detailed Description

application for USART to extender

Author

RichterT

Date

24.06.2020

Definition in file **app_usart_to_extender.h**.

Macro Definition Documentation

◆ D_FlagHdrEnd

```
#define D_FlagHdrEnd   0x69
```

byte to mark header end

Definition at line **93** of file **app_usart_to_extender.h**.

◆ D_FlagHdrStart

```
#define D_FlagHdrStart   0x68
```

byte to mark header start

Definition at line **92** of file **app_usart_to_extender.h**.

◆ D_INVALID_BYTE

```
#define D_INVALID_BYTE  0xFF
```

value to mark invalid byte

Definition at line **31** of file **app_usart_to_extender.h**.

◆ D_INVALID_QWORD

```
#define D_INVALID_QWORD  0xFFFFFFFF
```

value to mark invalid 32bit value

Definition at line **32** of file **app_usart_to_extender.h**.

◆ D_MinRxTelegramLen

```
#define D_MinRxTelegramLen  (PosPayloadData + 2)
```

minimum number of byte for RX telegram

Definition at line **74** of file **app_usart_to_extender.h**.

◆ D_PARAM_LEN

```
#define D_PARAM_LEN  0xFF
```

maximum size of parameter belonging to command

Definition at line **34** of file **app_usart_to_extender.h**.

◆ D_PROT_MESSAGE_VERSION

```
#define D_PROT_MESSAGE_VERSION  0x01
```

actual protocol version of current message

Definition at line **33** of file **app_usart_to_extender.h**.

Typedef Documentation

◆ sUartMsg2Ext

typedef struct **uart_msg_to_ext sUartMsg2Ext**

Structure definition for UART msg object.

Enumeration Type Documentation

◆ eAddrType

enum **eAddrType**

enumeration for valid address types

Enumerator	
ADDR_IOTB	address type: IoT board
ADDR_CCB	address type: Chip carrier board by Q-loud

Definition at line **117** of file **app_usart_to_extender.h**.

◆ eCmdId

enum **eCmdId**

Enum for 16bit command identifier in range 0..65535 (0x0 .. 0xFFFF)

Enumerator	
Cmd_PV_Heating	ID: 0x0001, Heating enable/disable, Type: bool, range: [0,1]
Cmd_PV_ActualSpd	ID: 0x0135, Actual Speed [Hz], Type: int, range: [0,999999]
Cmd_PV_DrvCurrent	ID: 0x0136, Current of driver [A], Type: float, range: [0,9999.99]
Cmd_PV_OpHrsPump	ID: 0x0137, Operating Hours Pump [h], Type: sign. int, range: [0,65535]
Cmd_PV_OpHrsElec	ID: 0x013A, Operating Hours Electronic [h], Type: sign. int, range: [0,65535]
Cmd_PV_TempElec	ID: 0x0146, Temperatur Electronic [°C], Type: sign. int, range: [0,999999]
Cmd_QL_StatusCCB	ID: 0x0400, Status Chip Carrier Board, Type: int, range: [0,1,2]=[Off,On,Standby]
Cmd_MB_Status	ID: 0x0500, Status Main Board, Type: sign. int, range: [-128,127]=[0x80,0x7F]
Cmd_MB_EnablePump	ID: 0x0501, Enable Pump, Type: unsig. int, range: [0,1,2]=[Off,On,Standby]
Cmd_MB_LedOnOff	ID: 0x0502, LED enable/disable, Type: unsig. int, range: [0,255]=[0,0xFF]

Definition at line **38** of file **app_usart_to_extender.h**.

◆ eDataType

enum **eDataType**

enumeration for data types

Enumerator	
DTYPE_BOOL_OLD	data type: old boolean
DTYPE_UINT	data type: unsigned integer (4 byte)
DTYPE_REAL	data type: real / floating
DTYPE_STRING	data type: string
DTYPE_BOOL_NEW	data type: new boolean
DTYPE_USHORT	data type: unsigned short (2byte)
DTYPE_EXPO_N	data type: exponent N

Definition at line **106** of file **app_usart_to_extender.h**.

◆ eMsgStates

enum **eMsgStates**

Enum for states of message interpretation.

Enumerator	
StHdrFlagStart	flag for start the header
StHdrLen	length of payload (2byte, unsigned 16bit), MSB + LSB
StHdrVersion	version byte of protocol
StHdrDstAddress	address byte of destination
StHdrOcd	operation code (1byte), e.g. GET, MGET, SET, MSET, DAT, ACK, NAK
StHrdSeq	sequential number (4byte), overflowing
StHdrFlagEnd	flag for end the header
StPayloadData	state to collect the payload data
StCrcCalc	state for CRC calculation
StError	state for error handling

Definition at line **78** of file **app_usart_to_extender.h**.

◆ eOcdType

enum **eOcdType**

enumeration for type of OCD (opcode)

Enumerator	
OCD_GET	opcode type: GET (get only 1 parameter with 1 byte)
OCD_MGET	opcode type: Multiple GET (get only 1 parameter with several bytes)
OCD_SET	opcode type: SET (set only 1 parameter with 1 byte)
OCD_MSET	opcode type: Multiple SET (set only 1 parameter with several bytes)
OCD_ACK	opcode type: acknowledge (positive response)
OCD_NAK	opcode type: neg. acknowlege (negative response)
OCD_DAT	opcode type: data (several bytes, stream data, etc.)

Definition at line **95** of file **app_usart_to_extender.h**.

◆ eRxBytePos

enum **eRxBytePos**

Enum for UART receive byte positions.

Enumerator	
PosHdrFlagStart	flag for start the header
PosHdrLen	length of payload (2byte, unsigned 16bit), MSB + LSB
PosHdrVersion	version byte of protocol
PosHdrRcvAddress	address byte of receiver
PosHdrOcd	operation code (1byte), e.g. GET, MGET, SET, MSET, DAT, ACK, NAK
PosHrdSeq	sequential number (4byte), overflowing
PosHdrFlagEnd	flag for end the header
PosPayloadData	state to collect the payload data

Definition at line **62** of file **app_usart_to_extender.h**.

Function Documentation

◆ crc_calc()

```
uint16_t crc_calc ( char *     buf,
                    uint16_t   data_byte_len
                  )
```

calculate the CRC of given buffer with given data length in bytes

Parameters

buf	pointer to signed char buffer
data_byte_len	length of data bytes

Returns

16bit unsigned CRC value

◆ uart_binary_data_interpreter()

```
int uart_binary_data_interpreter ( byte_t *          data,
                                   uint16_t          data_len,
                                   sUartMsg2Ext *    pMsg2Ext
                                 )
```

UART binary data interpreter.

Parameters

data	pointer to data block with unsigned characters of data_len bytes
data_len	number of bytes of given data block
pMsg2Ext	pointer to structure with message content to extender

Returns

0 for error-free, otherwise error

◆ uart_get_msg_obj()

```
sUartMsg2Ext* uart_get_msg_obj ( void  )
```

get message object for user protocol handling

Parameters

void

Returns

pMsg2Ext pointer to structure with content of message to extender

◆ uart_init()

void uart_init (void)

UART initialization.

Parameters

void

Returns

void

◆ uart_init_msg_obj()

void uart_init_msg_obj (**sUartMsg2Ext** * pMsg2Ext)

UART initialization of msg object.

Parameters

pMsg2Ext pointer to structure with content of message to extender

Returns

void

◆ uart_is_valid_msg_address()

bool uart_is_valid_msg_address (byte_t dst_addr_id)

check for valid message address in telegram

Parameters

dst_addr destination address identifier

Returns

true for valid version id, false otherwise

◆ uart_is_valid_msg_version()

bool uart_is_valid_msg_version (byte_t version_id)

check for valid message version of telegram

Parameters

version_id version identifier

Returns

true for valid version id, false otherwise

◆ uart_is_valid_ocd()

bool uart_is_valid_ocd (byte_t ocd_id)

check for valid OCD id in telegram

Parameters

ocd_id OCD identifier

Returns

true for valid OCD id, false otherwise

◆ uart_test_get_crc()

uint16_t uart_test_get_crc (uint8_t idx,
uint16_t * crc,
bool * is_valid
)

UART test function: get crc of buffer of given index.

Parameters

[in]	**idx**	index of buffer to test
[out]	**16bit**	unsigned integer of CRC
[out]	**is_valid**	boolean state of valid CRC

Returns

0 for error-free, otherwise error