Designed By: Michael N.
Document Type: Requirements
Date: 29.09.2022
Project: Alnicko

ALNICKO Lab
Estonia, Tallinn, 10151,
Kesklinna linnaosa, Ahtri tn 6-401
hello@alnicko.com

Requirements

Communication is based on request-response messages. Only the Host can initiate communication by sending a request frame. The slave should response on all valid frames.

Implement library that will provide:

for Host:

1. can create request frame
2. can parse received bytes stream and detect frame
3. can detect valid and invalid frame

for Slave:

1. can parse received bytes stream and detect frame
2. can detect valid and invalid frame
3. can create response frame

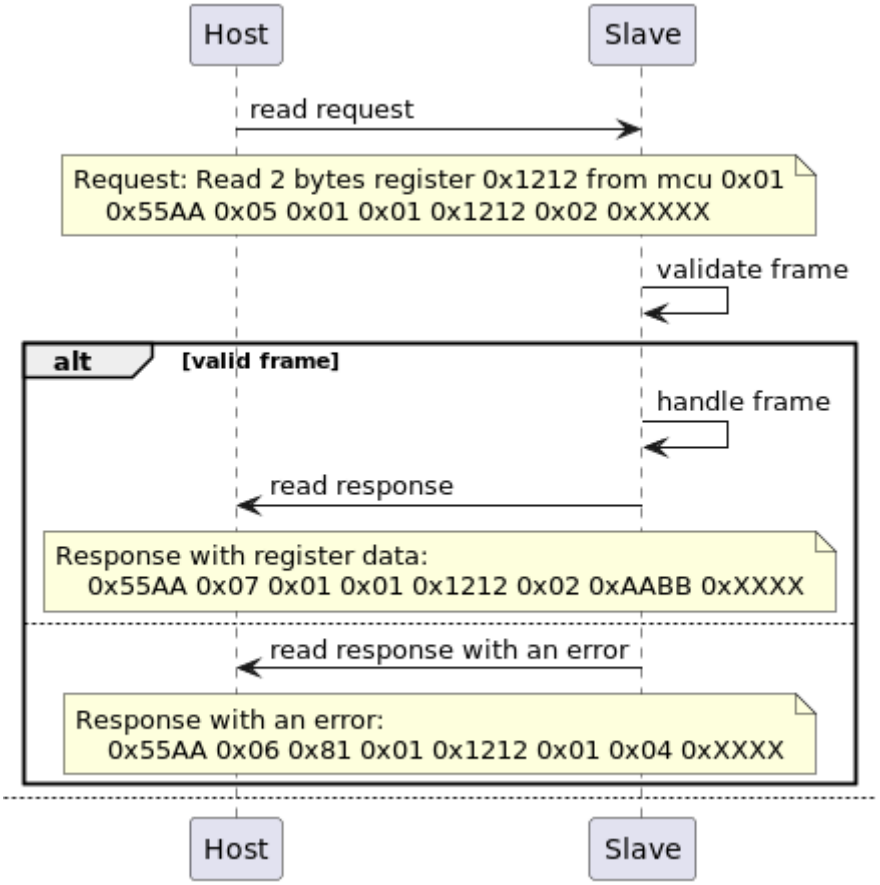
Implementation requirements:

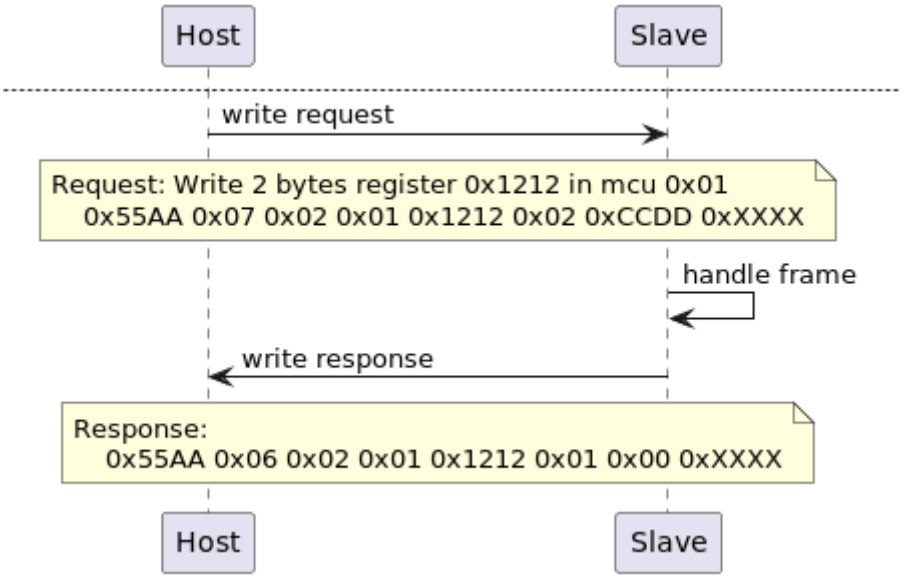
1. do not use dynamic memory
2. must work on little and big endian platforms
3. should be multi instanceable
4. use cmake
5. should be covered by unit tests

The information contained in this document is strictly confidential and is intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

ALNICKO Lab
Estonia, Tallinn, 10151,
Kesklinna linnaosa, Ahtri tn 6-401
hello@alnicko.com

Communication examples:





The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

ALNICKO Lab
Estonia, Tallinn, 10151,
Kesklinna linnaosa, Ahtri tn 6-401
hello@alnicko.com

Serial Communication Protocol Description.

Version 1.0

Revision History:

Date	Rev No.	Description	By
14.01.2020	0.1	Initial version	Michael Novotny

This document describes serial communication protocol.
That protocol is used to communicate between the following nodes:

- Linux to Main board
- Linux to Supervisor
- Supervisor to Peripheral boards
- Supervisor to Linux

UART Transport

Configuration

The following default UART couration is used:

- Baud rate: 115200
- No hardware (RTS/CTS) flow control.
- 8-N-1 byte format.

Signal Operation

UART transport sends and receives data synchronously. Data can not be sent and received simultaneously, transfer of frames can be initiated only by the host (Master) processor.

Transport Frame Format

The UART transport frame format is shown in the following figure. The left-most field is transmitted first over the wire. As shown, valid frames can range from **6 to 6+N** bytes in length, depending on the Length field.

Bytes: 2	2	0-N	2
SOF	Length	Payload	FCS

The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

SOF: Start of frame indicator, which is always set to 0x55AA.
Length: Length of the Payload field. N is maximum payload size - 1024 Bytes.
Payload: This is the package format data.
FCS: Frame check sequence, computed as a CRC16 of all the bytes in 'Length' and 'Payload' frame fields.

Package Format

The interface defines the common type of package format. The package starts with a 5-byte Header field, consisting of an 8-bit Type and CMD0 fields, followed by 16-bit CMD1 and CMD2 fields. After the Header bytes, a variable length Data field may be appended to form a complete frame of up to 1024 bytes.
Header and Data elements are packed on consecutive one-byte boundaries - there is no padding between elements of different sizes and data types. **For multi-byte elements, the lowest order byte is buffered first. For example, a 16-bit value will have its least significant byte (LSB) sent first, followed by its most significant byte (MSB).** As shown in the following sections, a valid Data block can range from 0 to 1018 bytes in length, depending on the specific command and the type of frame in use.

Common frame format

Header					Package data
1 byte	1 byte	2 byte	2 byte		Len (0-1018) bytes
Type	CMD0	CMD1	CMD2		DATA

Header field codes

Type field codes:

Bit	Alias	Description
7	T_ERROR_FLAG	Error flag. Valid only for response. 0 - Request was processed successfully. 1 - Request was not processed successfully. The DATA field contains error code.
6-4	-	Reserved.
3-0	T_CMD_CODE	Frame type

Code	Alias	Description
0x01	REQR	Request to read a register.
0x02	REQW	Request to write a register.

The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

ALNICKO Lab
Estonia, Tallinn, 10151,
Kesklinna linnaosa, Ahtri tn 6-401
hello@alnicko.com

Type code 0x01, 0x02, 0x03, 0x04. Package format

In that type of requests CMDs fields have the following meaning:

Type	CMD0	CMD1	CMD2	DATA*
<TYPE>	MPU_ADDR	REGISTER_ADDR	DATA_LEN	<DATA>

The DATA field is optional, depending on TYPE code.

MPU_ADDR values:

MPU_ADDR	Description
0x00	Reserved
0x01-0xFE	Controller address.
0xFF	Reserved

REGISTER_ADDR values:

REGISTER_ADDR	Description
0x0000	Reserved
0x0001 0xFFFE	Register address in controller.
0xFFFF	Reserved

DATA_LEN values:

DATA_LEN	Description
0 - 1018	Length of DATA field in bytes.

Package DATA values:

Register access mode		Data field content description
Read REQR	Request	No data
	Response	Actual register data. Length and format depend on MPU_ADDR and REGISTER.

The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

		Or ERROR code in case T_ERROR_FLAG is set.
Write REQW	Request	New data for the register. Length and format depend on MPU_ADDR and REGISTER.
	Response	Error code

Error codes:

Code	Description
0x00	OK, no errors.
0x01	Incorrect frame format
0x02	Incorrect Type
0x03	Incorrect MPU address
0x04	Incorrect REGISTER address
0x05	Incorrect data length
0x06	Incorrect data values
0x07	Slave device error. Data integrity (CRC).
0x08	Slave device error. Timeout.
0x09	Slave device error. Busy.

MPU_ADDR

Table of addresses of sub-system (board/MPU) in the system

Address	Alias	Board name
0x00		RESERVED
0x01	addr_linux	Linux
0x02	addr_main	Main
0x03	addr_ventilation_1	Ventilation_1
0x04	addr_ventilation_2	Ventilation_2
0x05	addr_ventilation_3	Ventilation_3
0xFE	addr_supervisor	Supervisor

The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.

0xFF		RESERVED

The information contained in this document are strictly confidential and are intended for the Owner's employees, representatives or agents who need to have access to the Confidential Information only.