



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



**FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH
TECHNOLOGIÍ**

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

DESIGN OF THE USER-FRIENDLY TOUCH SCREEN GUI AND A PHYSICAL CONNECTION TO AN EXISTING SIMULATION HARDWARE DEVICE

UŽIVATELSKY PŘÍVĚTIVÉ DOTYKOVÉ GRAFICKÉ ROZHRANÍ PRO EXISTUJÍCÍ SIMULAČNÍ
HARDWARE AS-INTERFACE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. JAN HUSAR

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. RADEK ŠTOHL, Ph.D.

BRNO 2014



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Jan Husar

ID: 115184

Ročník: 2

Akademický rok: 2013/2014

NÁZEV TÉMATU:

Uživatelsky přívětivé dotykové grafické rozhraní pro existující simulační hardware AS-Interface

POKYNY PRO VYPRACOVÁNÍ:

Design and implement a communication interface base on the resistive touch screen GUI STK-480272C in order to control the existing AS-Interface Slave Simulator device. Using the GEMstudio for this purpose, the following tasks should be fulfilled:

- Design the project structure, design of hardware-display communication protocol
- Design of graphical user interface, program system functions
- Test the functionality on the target hardware

Design and implement a connection of this communication interface to the simulation hardware based on the FPGA technology. For this purpose use the development tool Quartus II. The required hardware interface is to design by using the hardware description language VHDL. Following tasks should be fulfilled:

- Design of hardware interface concept, design of hardware-display communication protocol
- VHDL implementation, Simulation of system functionality
- Test the functionality on the target hardware

DOPORUČENÁ LITERATURA:

Team of autors: Automatisierungs und kommunikationssysteme FTZ HTWK Leipzig. AS-i Netzwerk Simulator. Leipzig, 2010. p. 96

Termín zadání: 10.2.2014

Termín odevzdání: 19.5.2014

Vedoucí práce: Ing. Radek Štohl, Ph.D.

Konzultanti diplomové práce: M.Eng. Dipl.-Ing. Tobias Rudloff

doc. Ing. Václav Jirsík, CSc.

Předseda oborové rady

Abstract

This master's thesis deals with fundamental principles of the industrial bus AS-Interface as well as with an extension of the FTZ AS-Interface Slave Simulator by touch panel which makes it incomparably easy to control this device. Progress and solution of the touch communication interface for this Slave Simulator have been sketched out by using Amulet LCD module STK-480272C. The design of this communication interface has been created by GEMstudio, software of Amulet Technologies and graphics programs. Further this study deals with software adjustment of FTZ AS-i Slave Simulator. The modification of controlling FPGA in VHDL programming language has been described which ensures the communication with the touch screen. Last chapter deals with questions concerning a user-friendly design of the application.

Keywords

AS-Interface, topology, communication, profiles, FTZ AS-Interface Slave Simulator, GEMstudio, touch screen, STK 480272C, graphic design, FPGA, VHDL, user-friendly

Abstrakt

Práce uvádí základní informace o průmyslové sběrnici AS-Interface a popisuje její funkce. Dále se zabývá rozšířením stávajícího FTZ AS-Interface Slave Simulátoru o dotykový display, který značně usnadní ovládání tohoto simulačního nástroje. Je zde nastíněn návrh a řešení uživatelského dotykového rozhraní k tomuto simulátoru s použitím Amulet LCD modulu STK 480272C. Vývoj tohoto rozhraní je proveden pomocí GEMstudia, softwaru firmy Amulet Technologies a grafických programů. Dále tato studie pojednává o softwarové úpravě FTZ AS-i Slave Simulátoru. Jedná se o úpravu řídicího FPGA v jazyce VHDL zajišťující komunikaci s dotykovým displejem. Poslední kapitola se týká problematiky spojené s návrhem uživatelsky přívětivé aplikace.

Klíčová slova

AS-Interface, topologie, komunikace, profily, FTZ AS-Interface Slave Simulator, GEMstudio, dotykový display, STK 480272C, grafický design, FPGA, VHDL, uživatelsky-přívětivý

HUSAR, Jan. *Design of the user-friendly touch screen GUI and a physical connection to an existing simulation hardware device*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Specialization of Cybernetics, Control and Measurement, 2014. 80s. Supervised by Ing. Radek Štohl, Ph.D.

Declaration

I declare that I have elaborated my master's thesis on the theme of "*Design of the user-friendly touch screen GUI and a physical connection to an existing simulation hardware device*" independently, under the supervision of the master's thesis supervisor and with the use of technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the master's thesis I furthermore declare that, concerning the creation of this master's thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal copyright and I am fully aware of the consequences in the case of breaking Regulation S 11 and the following of the Copyright Act No 121/2000 Vol., including the possible consequences of criminal law resulted from Regulation S 152 of Criminal Act No 140/1961 Vol.

Brno

.....
(author's signature)

Acknowledgement

Special acknowledgement belongs to the FTZ Leipzig which offered me this topic and gave me a long term support. Nominally M.Eng. Tobias Rudloff was always helpful and patient to answer my question.

Another acknowledgement belongs to Ing. Radek Štohl, Ph.D. who gave me as well long term support from my home institution of FEEC, Brno University of Technology.

Furthermore I would like to thank my home university itself for giving me the possibility to stay abroad while writing this master's thesis which significantly increased my personal and technical skills.

Brno

.....
(author's signature)

1 CONTENTS

1	Contents	6
2	Terms Explanation	8
3	Introduction.....	9
4	AS-Interface	10
4.1	Basic Specification	11
4.2	Network Topology.....	12
4.2.1	Linear Topology (Bus topology).....	12
4.2.2	Tree (Free Topology)	12
4.2.3	Star Topology.....	13
4.2.4	Ring Topology.....	13
4.3	Organisation of Data Flow.....	14
4.4	System of AS-i Communication.....	14
4.4.1	Physical Layer	15
4.4.2	Data Link Layer	17
4.4.3	Application Layer.....	17
4.5	AS-i Device Profiles	20
4.5.1	Slave Profiles.....	21
4.5.2	Master Profiles	23
5	FTZ AS-Interface Slave Simulator	24
5.1	Basic Information	24
5.2	Amulet Resistive 4.3” GEMmodule™	25
5.3	Amulet Capacitive 4.3“ GEMmodule™ comparison.....	26
5.4	Linking Module	27
6	Project Design Using GEMstudio.....	28
6.1	Introduction.....	28
6.2	Project Structure	28
6.3	Application Description	30
6.4	Project Design.....	31
6.4.1	META Refresh Object	32
6.4.2	Amulet InternalRAM	33
6.4.3	Usage of InternalRAM	34
6.4.4	Program Description	35
6.4.5	GEMscript	38
6.4.6	Main Program Description.....	39
6.4.7	Other Page Processes	40

7	Graphic Design	41
7.1	Draft.....	41
7.2	Final Design.....	42
7.2.1	Colour Theory	43
7.2.2	Final Look	44
7.2.3	Photo Shop	47
7.2.4	Fonts	48
7.2.5	Touch Target Size	48
7.3	Problems with Size of the Memory	49
8	Testing of Communication	51
9	FPGA-Amulet Communication Protocol.....	54
9.1	Introduction.....	54
9.2	Connection Arrangement.....	54
9.3	Hardware Description.....	55
9.3.1	Board Altera DE1	55
9.3.2	USB to UART FT232	56
9.4	FPGA of Final FASS	57
9.5	Software Description	58
9.5.1	Quartus II and ModelSim	58
9.6	VHDL	60
9.6.1	Introduction	60
9.6.2	Program Structure	60
10	Implementation of VHDL	62
10.1	Program Structure	62
10.2	Main	62
10.2.1	Process of Page 1	63
10.2.2	Process of Page 2	63
10.2.3	Process of Page 3	64
10.2.4	Simulation of Command Counter	65
10.3	Amulet UART.....	65
11	Test of User Usability	66
11.1	Introduction.....	66
11.2	Test Performance and Design Changes	67
11.3	Conclusion	72
12	Conclusion.....	73
13	List of Figures	74
14	List of Tables.....	76
15	BiblioGraphy.....	77
16	List of Appendices	80

2 TERMS EXPLANATION

AS-i – AS (Actuator Sensor) - Interface, industrial bus

ASCII protocol – ASCII based communication protocol, one byte is sent as two ASCII characters

CBUS – a communication protocol based on the seven layer OSI model, often used in home and building automation

Decentralized control system – there are more PLC stations which are communicating in one of the higher-order industry networks. This way is more reliable thanks to the division of the program into single PLCs.

Deterministic access – to a member of the communication is assigned the permission to transmit/receive data, there are any collisions on a bus.

Embedded system – computed devices working in real time, which contain a microcontroller, used for different control utilization (mobile phones, MP3 players etc.).

FASS or Slave Simulator – FTZ AS-Interface Slave Simulator

Flash – reprogrammable non-volatile memory of Amulet LCD module

FTZ (Forschungs und Transferzentrum) Leipzig – Research centre of the Faculty Electrical Engineering and Information Technologies of HTWK Leipzig

GUI – Graphical User Interface

Master/slave – participants of communication which is hierarchal organised

Modbus RTU – a communication protocol popular in industrial automation, one byte is sent as one character, the data integrity of the communication is secured by CRC protocol and parity bit. RTU (Remote Terminal Unit)

MSByte/LSByte – Most and Least Significant Byte

Multidrop – to one cable is allowed to connect more than one participant, their calling access is deterministic

Page – a page of Amulet GUI design

Respond time – time needed for data transfer from one device to another

SDRAM – Synchronous Dynamic Random Access Memory

Serial communication – process, in which bits of data are transferred one after another

Starter kit – hardware + software used for user-friendly programming for e.g. embedded system using higher-order programming form

Viewing angle – an angle from which is best to observe the GUI for good image quality

Widget – one certain object of Amulet GUI design

3 INTRODUCTION

During last years, there is emphasis put on development of devices which are high user-friendly and easy to control. One of the instruments that allows to achieve this point is to use high-resolution touchscreens. Thanks to them, users can understand fast and intuitively the application which is well arranged, easy and fast to control. What is more the designing devices can be smaller because the touch screen includes the tasks of information output and input. It brings adjustments and completion for already existing devices. The kinds of tasks mentioned above as well as other similar tasks constitute the topic of this master's thesis.

This document treats an extension of the FTZ AS-interface Slave Simulator (FASS) by touch panel (communication interface) and possible changes caused by this adjustment.

The first chapters of this master's thesis describe fundamentals of the fieldbus AS-Interface. The reader gets an overview of principal topics such as network topology, message flow, system of communication etc. Furthermore the architecture of FASS as well as the extension by Amulet LCD module STK-480272C have been depicted.

The following chapters should be understood as the most important part of the project. There the progress and the solution of the touch communication interface for this Slave Simulator have been described by using the Amulet LCD module. The design of this communication interface has been created by GEMstudio, the software of Amulet Technologies. The Microsoft Visio and Adobe Photoshop software has been used for graphical design.

The next part of this thesis deals with the connection of the Amulet LCD module with the Slave Simulator motherboard. The core of the motherboard has been created by FPGA (Field Programmable Gate Array) chip. The chapter begins with essential information of FPGA architecture and VHDL programming language. Furthermore the progress of achieving the communication between Amulet LCD module and FPGA has been described, as well as the design and the structure of the program, information exchange and adjusting of the program to the Amulet communication protocol.

Hereinafter a user- friendly test has been conducted and its data projected in order to prove the user friendliness of the application and other main traits of the design e.g. the importance of colour in graphic design.

The result of this master's thesis is study of the stated problematic which will be continued in the future. The aim is the development of a testing device for new v3.0 generation devices working with AS-i. This thesis faces problems and gives possible solution for the development.

The task was given and the progress supported by FTZ Leipzig in Germany where the main part of this thesis was treated.

4 AS-INTERFACE

AS-Interface (Actuator Sensor Interface), referred further to as AS-i, is a fieldbus conceived for networking binary terminal devices, sensors, actuators. As other fieldbus systems it is used in automation. AS-i is characterized by:

- Digital serial communication
- Suitability for industrial utilization
- Suitability for connection of sensors/actuators
- Device application in field of decentralized control units
- Capability of “multidrop”

It substitutes parallel networking PLC with peripherals and therefore the costs of underground line decrease. Thanks to the penetration technique, the connection and disconnection of new participants of communication is very simple and possible during the running state of the control system. In this way the down-time caused by failure, service and revision is reduced [2].

AS-i is often marked as Sensorbus. Essentially Sensorbus is used in the lowest level of the so-called Communication pyramid of industrial automation. See Fig. 1 Communication pyramid of industrial automation [29] below. This level, known as the “Sensor/Actor level”, is characterized by simplicity, cost-effectiveness, transparency, size of the data transfer and the response time. The size of the net rate has been presented in *bit* unit range. The response time has been presented in the *ms* unit range [2].

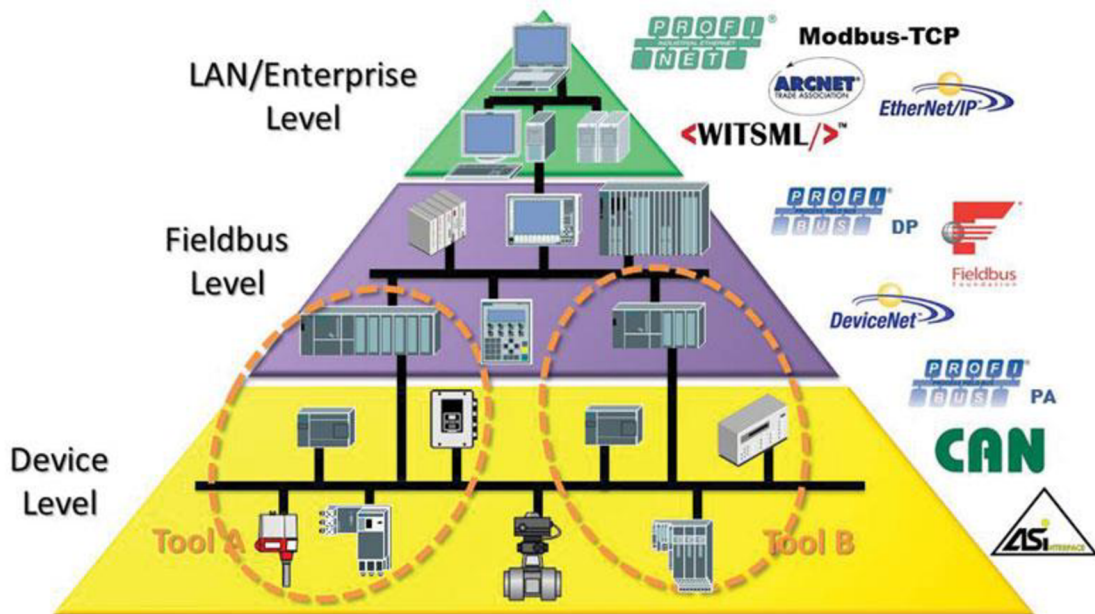


Fig. 1 Communication pyramid of industrial automation [29]

4.1 Basic Specification

Tab. 1 Basic specification of AS-Interface [4]

Data transfer:	Master-multislave
Addressing:	Every slave obtains a permanent address in range 0-31. The address is assigned by the master or by hand addressing tool. New slave has the address 0 as default. The address is stored in non-voltage memory.
Network structure:	Linear, Star, Ring, Mesh, Tree topology is possible
Transmission medium:	Non-shielded, not twisted and not terminated cable with two conductors for data and energy (24V DC), up to 8A per line. Connected by penetration technology. The one device power consumption is max. 200mA for specification v1.0 and 100mA for v2.1 specification.
Number of slaves:	31 for specification v1.0 and 62 for specification v2.1 62, address in range 0-31 share always two devices, so-called device A and device B.
Number of I/O:	A slave has up to 4 binary inputs and outputs. It means up to 124 (128) binary inputs and outputs by specification v1.0 248I, 186O by specification v2.1 248 I/O by specification v3.0 with application of 62 slaves
Communication:	Cyclic messages from master to slave and from slave to master containing 4 data bit. Cyclic messages contain 4 parameterization bits.
Communication speed:	167 kb/s
Device profiles:	15 standards (distinguished by ID code) by v1.0 225 standards by v2.1
Network cycle:	Cycle depends on the number of slaves. In full configuration (32 slave devices) the cycle time is < 5ms. In full configuration for v2.1 (62 slave devices) the cycle time is < 10ms.
Network dimensions:	Total dimension of the network including branches cannot be bigger than 100m and up to 500m with repeaters. A doubled network size can be achieved with a terminator.
Error detection:	The way of modulation and message format provides reliable error detection. In case of message deformation repeating is assured.
Master:	Assurance of network initialization, identification of connected devices, cyclic setting of parameters for all slaves, error detection during the data transfer, cyclic calling to all slaves, search for new-connected slaves which were changed for the reason of failure and giving them a new address

4.2 Network Topology

By this term a physical structure of network is specified, respectively alignment of communication participants. AS-i allows following types of network topology: Linear, Tree, Star and Ring. The basic characteristic of topologies according to [2] has been depicted below.

4.2.1 Linear Topology (Bus topology)

All nodes are connected together by a single bus with short stub lines and with limited longitude. All nodes take part in communication at all times with certain rules. The line generally has terminating resistors at each end so that there are no reflections. A unique identification of each slave is necessary, generally consisting of an assigned address.

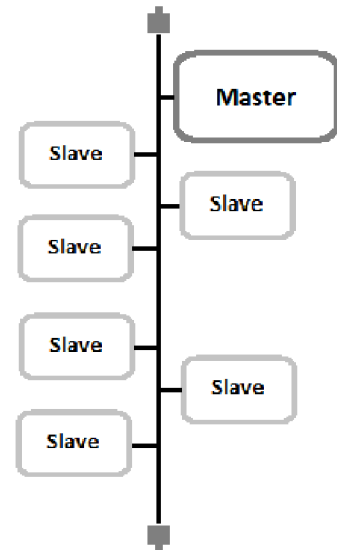


Fig. 2 Linear topology

4.2.2 Tree (Free Topology)

This one is similar to Linear topology apart from the limitation of stub lines. In addition, it is possible to create new stub lines from existing stub lines and the usage of terminating resistors is not mandatory. This topology allows a big flexibility during the network installation. A new node is connected to the closest branch.

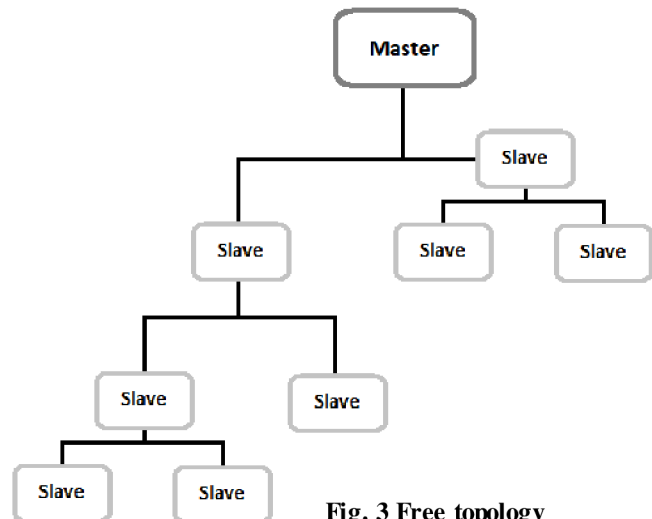


Fig. 3 Free topology

4.2.3 Star Topology

The centre node is connected to each field device via 2-point connection (traditional parallel wiring).

As a result, this topology has no saving cabling. It is not necessary to assign address to the slave. No bus access procedure has to be established.

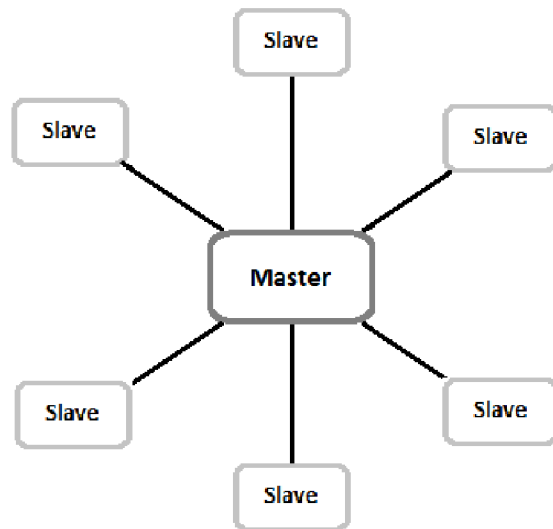


Fig. 4 Star topology

4.2.4 Ring Topology

This topology is created, when multiple slaves are connected with 2-point connections to each other in series and then the last node is connected to the first node.

Communication flows through each slave, and each individual assumes the function of an extender or repeater.

There is an address assigned to each slave in AS-i ring topology. But generally in this type of topology it is not necessary to assign an address to the slave. The address is usually given by the slave's position in the ring.

If a new slave connection is required, it will be needed to cut off the communication, disconnect the ring and add the new slave.

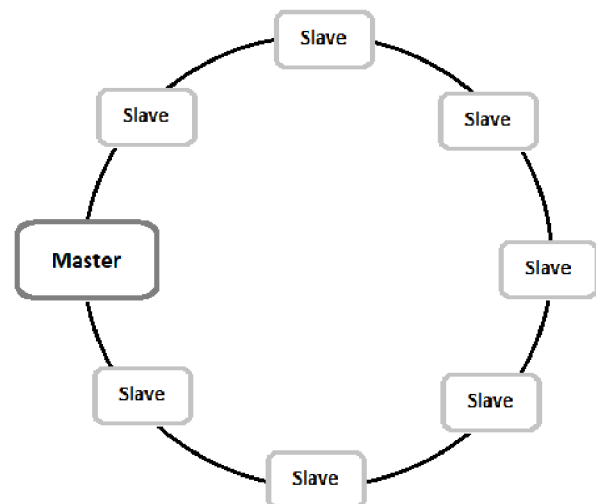


Fig. 5 Ring topology

4.3 Organisation of Data Flow

Master-Slave organization with cyclic polling

This is a hierarchal organisation. There can be only one master in the system and it has sole authority to transmit. It generally sends messages cyclically to the slaves. The data flow goes from the central master to a selected slave and returns. Each message is automatically acknowledged and can, if there is an error, be quickly repeated [2].

4.4 System of AS-i Communication

In fieldbus area the speed of communication is very important. This communication is generally simple, in bits units. Therefore the application of all layers of ISO/OSI model is not needed. Their task substitutes the Application layer [3].

Tab. 2 ISO/OSI model of AS-Interface [3]

Num.	ISO/OSI Layer	Function	AS-Interface
7.	Application	Providing the network service to the user	Message, cycle, profiles, automatic addressing
6.	Presentation	Transformation of network formats to user formats	
5.	Session	Logging on and off of the connection	
4.	Transport	Data preparation for network transport interfaces	
3.	Network	Address preparation, direction of data ways	
2.	Data Link	Data structure, frame, protection, error handling	Data telegram, start and end bit, parity, error handling
1.	Physical	Mechanical and electrical connection for information transmission	Cable, power supply, data decoupling, APM

4.4.1 Physical Layer

Transmission Medium

The bus cable for AS-i is the familiar non-shielded and not twisted pair, 2-conductor flat cable which transports data and slave power. The cable is specially formed and the position of conductors is known and fixed therefore there are no problems regarding the polarity reversal issues. Thanks to the penetration technique the cable does not have to be split or stripped, nor does it need a shield attached (see Fig.6 and 7). This does not cause additional voltage drops. The technology achieves high level of the ingress protection (IP67, IP68, IP69K) [3].

If the way of power supply is insufficient, a second supply voltage circuit is provided which is galvanically isolated from the AS-i network. Black version of the AS-i cable is used for supply voltage up to 30V DC. Red cable is used for supply voltage 230/120V AC [3].

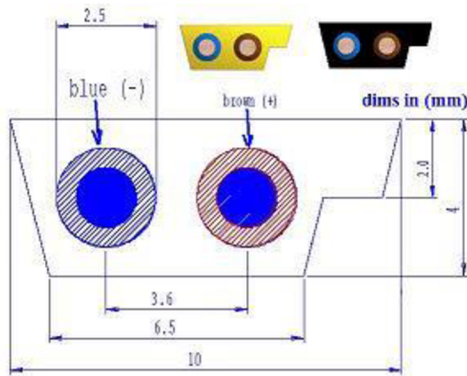


Fig. 6 Transmission medium

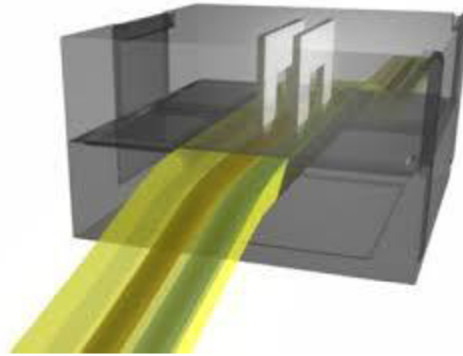


Fig. 7 Penetration technique of AS-i

Voltage Source

The voltage source supplies slaves and part of the master over the 2-conductor cable. It provides a DC voltage from 29,5 to 31,6V at the current up to 8A. It is also designated for balancing the AS-i network to avoid symmetrical noise coupling. It provides special circuits in the voltage source. The other feature of the source is data-decoupling. The circuit for this purpose consists of two inductor (50mH each) and two parallel-wired resistors (39Ω each). Inductors convert current pulses to voltage pulses [3].

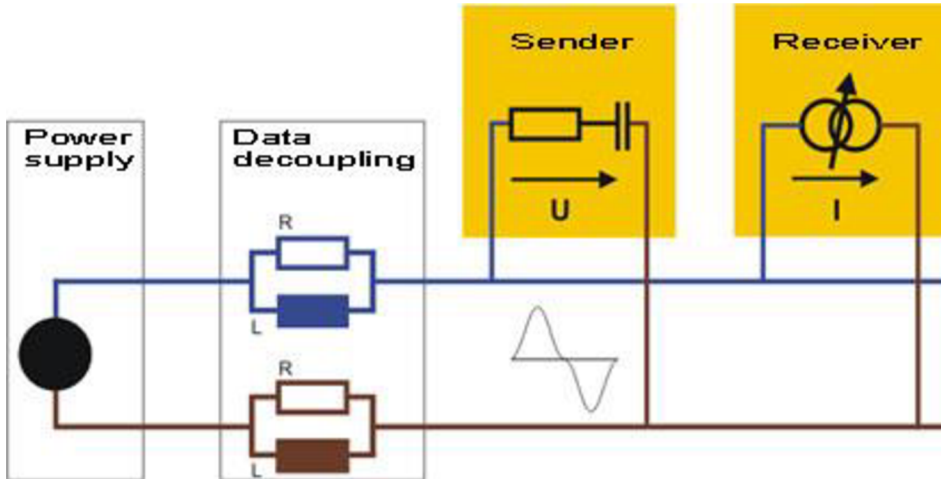


Fig. 8 Physics layer [1]

Way of Signal Modulation

A new way of modulation was specified and developed. It is called Alternating Pulse Modulation (APM). The sent bit sequence is firstly encoded into bit-sequence that performs a phase change whenever the sent signal changes (Manchester coding where 0 and 1 represents fall and rise edge of a signal), see Fig.9. It is resulting in a current signal which uses the inductors in voltage source for generation of voltage signals into the AS-i cable.

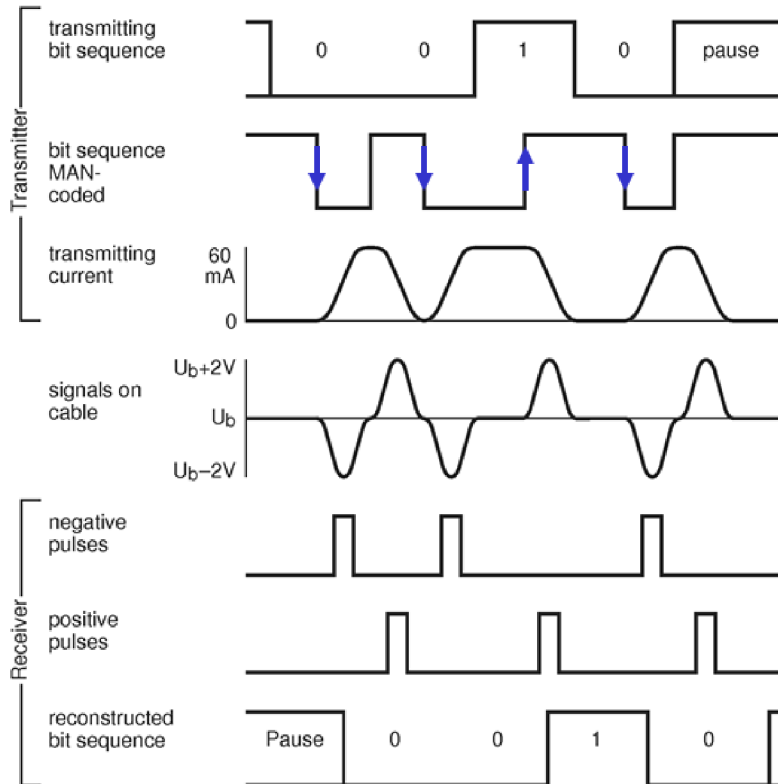


Fig. 9 Principle of communication [1]

4.4.2 Data Link Layer

The master uses a cyclical polling and short telegrams in order to communicate with slaves. Every slave has to answer in exact, predefined time range. If an error occurs during the communication, telegrams obtained by the master, with no or invalid reply are repeated. Following kinds of communication errors can be distinguished: Start-bit error, alternating error, pause error, information error, parity error, end-bit error, telegram length error [1].

An AS-i message consists of a master request, a master pause, a slave response and slave pause. All master requests are exactly 14 bits long and all slave responses have length of 7 bits [1]. See one example on Fig.10 below.

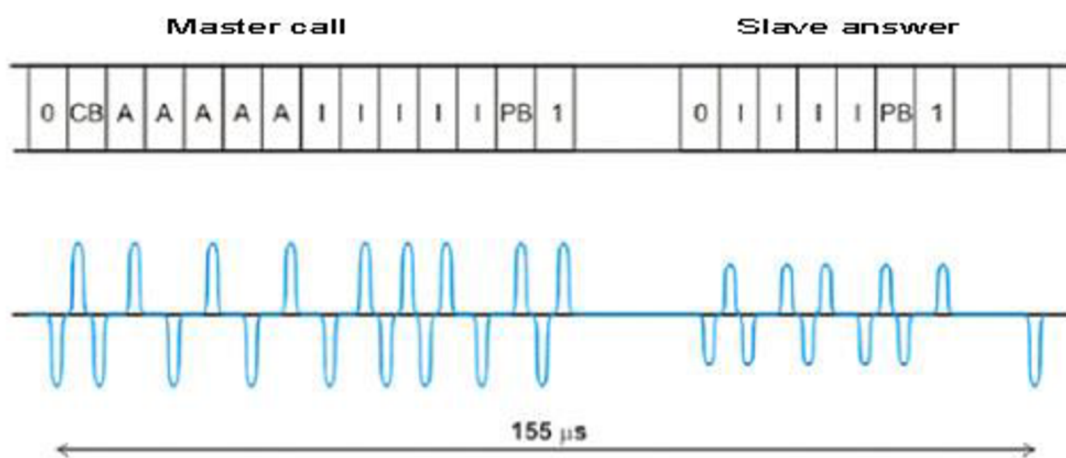


Fig. 10 Structure of AS-i message [1]

Term explanation from Fig.10

- 0 – Start bit
- CB – Control bit
- A – Address (5 bits)
- I – Information (5 bits)
- PB – Parity bit
- 1 – End bit

4.4.3 Application Layer

Master:

- recognizes in detail every slave thanks to the specific codes (IO Code, ID Code etc.)
- can call a particular slave and give him statements.
- communicates cyclically with slaves.

Network Cycle [1]

Standard addressing:

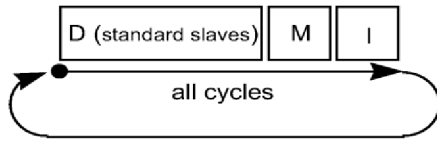


Fig. 11 Network cycle in standard addressing mode [1]

Extended addressing:

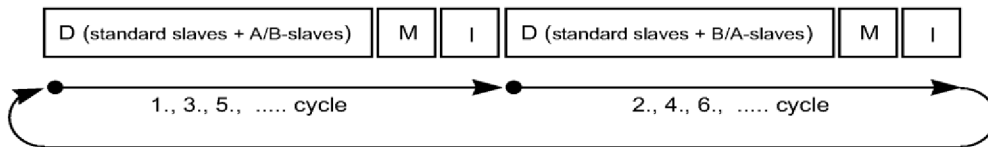


Fig. 12 Network cycle in extended addressing mode [1]

D – Data transfer I/O

M – Management

I – Inclusion (search for new devices)

Network Cycle Time [1]

It depends on number of devices:

$$T = n * 154 \text{ [us]}$$

n...number of slave devices + 2

(valid also for extended 62 slaves version)

Tab. 3 Master calls and related slave responses [19]

Instruction	MNE	Master Request														Slave Response						
		ST	CB	A4	A3	A2	A1	A0	I4	I3	I2	I1	I0	PB	EB	SB	I3	I2	I1	I0	PB	EB
Data Exchange	DEXG	0	0	A4	A3	A2	A1	A0	0	D3 -Set	D2	D1	D0	PB	1	0	D3 E3	D2 E2	D1 E1	D0 E0	PB	1
Write Parameter	WPAR	0	0	A4	A3	A2	A1	A0	1	P3 -Set	P2	P1	P0	PB	1	0	P3 I3	P2 I2	P1 I1	P0 I0	PB	1
Address Assignment	ADRA	0	0	0	0	0	0	0	A4	A3	A2	A1	A0	PB	1	0	0	1	1	0	0	1
Write Extended ID Code_1	WID1	0	1	0	0	0	0	0	0	ID3	ID2	ID1	ID0	PB	1	0	0	0	0	0	0	1
Delete Address	DELA	0	1	A4	A3	A2	A1	A0	0	0 Set	0	0	0	PB	1	0	0	0	0	0	0	1
Reset Slave	RES	0	1	A4	A3	A2	A1	A0	1	1 -Set	1	0	0	PB	1	0	0	1	1	0	0	1
Read ID Configuration	RDIO	0	1	A4	A3	A2	A1	A0	1	0 Set	0	0	0	PB	1	0	ID3	ID2	ID1	ID0	PB	1
Read ID Code	RDID	0	1	A4	A3	A2	A1	A0	1	0 Set	0	0	1	PB	1	0	ID3	ID2	ID1	ID0	PB	1
Read ID Code_1	RID1	0	1	A4	A3	A2	A1	A0	1	0 Set	0	1	0	PB	1	0	ID3	ID2	ID1	ID0	PB	1
Read ID Code_2	RID2	0	1	A4	A3	A2	A1	A0	1	0 Set	0	1	1	PB	1	0	ID3	ID2	ID1	ID0	PB	1
Read Status	RDST	0	1	A4	A3	A2	A1	A0	1	1 -Set	1	1	0	PB	1	0	S3	S2	S1	S0	PB	1
Broadcast (Reset)	BR01	0	1	1	1	1	1	1	1	0	1	0	1	1	1	-- no slave response --						
Enter Program Mode	PRGM	0	1	0	0	0	0	0	1	1	1	0	1	1	1	-- no slave response --						

Note: In Extended Address Mode the "Select Bit" defines whether the A-Slave or B-Slave is being addressed. Depending on the type of master call bit I3 carries the select bit information (Sel = A-Slave) or the inverted select bit information (-Sel = B-Slave).

In Table 3 there are possible master calls (commands) and slave responses depicted. With the aid of these commands the master can characterize a slave by reading identification codes, set slave settings, receive data from the slave etc. A short characterization of commands follows according to [2]:

Data request (Data Exchange) is the most important and most frequently used AS-i request. It is used for receiving data from a slave.

Parameter request (Write parameter) is used for sending data to a slave. It controls remotely certain functions in the slave.

Address assignment is used to assign a new address to a slave with previous address 00_{HEX} or to support the replacement of a defective slave. The slave stores the new address in non-volatile memory. This process takes 500ms.

Reset Slave call can be used to set the slave back to its base state. It has the same effect as the reset after having applied supply voltage.

Delete address is used for the temporary deletion of the operating address in order to change the slave address because the slave address can be changed only when it is 00_{HEX}. Then the "Address assignment" call is performed.

Read status call reads the content of the slave status register. It can contain three different flags.

- S0 - Address volatile - is set when the slave is storing the new address in memory

- S1 - Peripheral error - when the slave has detected an error in its peripherals (e.g. supply overload)
- S3 - Read error of non-volatile memory - when the read error occurs during the reset

Broadcast (Reset) extent command containing 15_{HEX} doesn't need to be replied. At the present time only "Reset" is defined. Additional broadcast commands are reserved for future use.

Read I/O configuration is one of the master's requests. It allows the master to read the set I/O configuration of a slave. This configuration is sent as the slave's response to this request and it is used together with "Read ID-Code" request for unambiguous identification of the slave. It represents the slave profile. The Configuration is 4-bits long and it is given by manufacturer and stored in the slave's memory without the possibility to change [3].

Read ID-Code

- The actual ID Code – it is fixed by manufacturer, permanently stored in the slave's memory and it cannot be changed
- Extended ID-Code 1 – the user can change it. It is used to identify slaves where the manufacturer is identical, however, from the user point of view are different. It is 4-bits long, 3-bits long for extended version (62 A/B slaves)
- Extended ID-Code 2 – the same as The actual ID Code which is used for expanded identification. It is 4-bits long and fixed by the manufacturer [3].

The combination of I/O configuration and the ID Codes make up the slave profile. This profile contains specification with meanings of the data and parameter bits that the slave expects or sends.

4.5 AS-i Device Profiles

AS-i supports communication via four Inputs/Outputs (I/O) data bits and four parameter bits between the master and single slaves. With the aid of parameter bits we can set the device profile respectively slave profile (in contrast to "master profile"). It is a definition of properties (e.g. number of I/O), behaviour, control and the way of communication (e.g. number of cycles for the reading of data). Slave profiles are independent on manufacturer which helps to increase the non-interchangeability and individuality of the slaves.

Profiles define: IO Code + ID Code [+ ID2 Code]. The Free profile for substandard devices: ID Code= 0xF. Code syntax: S-[IO Code].[ID Code].[ID2Code]. For the standard addressing 0-31 slave has up to 4 Inputs or 4 Outputs (possible both together). For extended addressing 0-31 A/B slave has 4 I or 3 O (possible both together). Slave which supports combination connection has up to 4 analogue I or O or other I or O which needs one or more information bits [1] [5].

4.5.1 Slave Profiles

The slaves can be set to standard or extended profiles. It depends on the number of participants on the bus or the amount of data transfer. The standard version is up to 32 participants and extended version allows up to 62 participants. Further Sub-profiles with usage of ID2-Code are designated for exact identification of the slaves.

There are 16 standard slave profiles, i.e. 16 different configurations of I/O data bits. Data bits can be configured as: input, output, bi-directional, three-state. What is more, each slave with the same I/O configuration can have data and parameter bits with different meanings. It encourages high flexibility of the system. [23]

Tab. 4 Table of standard slave profiles [23]

Slave-profiles	ID Code															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	I, I, I, I	0.0	0.1								0.A	0.B				0.F
1	I, I, I, O	1.0	1.1								1.A					1.F
2	I, I, I, B	2.0									R					2.F
3	I, I, O, O	3.0	3.1								3.A					3.F
I/O	I, I, B, B	4.0									4.A					4.F
O	I, O, O, O	5.0									5.A					5.F
	B, B, B, B	6.0									6.A					6.F
C	B, B, B, B	7.0	7.1	7.2	7.3	7.4	7.5				7.A	7.B		7.D	7.E	7.F
o	O, O, O, O	8.0	8.1								8.A					8.F
d	O, O, O, I	R									9.A					9.F
e	O, O, O, B	A.0									R					A.F
	O, O, I, I	R	B.1								B.A					B.F
	O, O, B, B	C.0									C.A					C.F
	O, I, I, I	R	D.1								D.A					D.F
	O, B, B, B	E.0									E.A					E.F
F	T, T, T, T	For Future Use														

Terms explanation of Tab.3

- I - Input
- O - Output
- B - Bi-directional
- T - Three-states
- R - Reserved

Tab. 5 Type of standard slaves [23]

Profile Name	IO	ID	Chapter
Remote I/Os (X = 0 ... E, not 9, B, D)	X	0	4.2.2
Free profiles for Standard Slaves (X = 0 ... E)	X	F	4.2.1
Remote I/Os with dual signals(X = 0, 3, 8)	X	1	4.2.3
Safety Sensors	0	B	4.4.9
Single Sensor with extended control	1	1	4.2.4
High speed Slave Profile for Combined Transaction type 5	6	0	4.4.8
Slave Profile for Combined Transaction type 1 (Analog profile)	7	1	4.4.1
Extended Slave Profile for Combined Transaction type 1 (Extended analog profile)	7	2	4.4.2
Slave Profile for Combined Transaction type 1 (Integrated Analog profile)	7	3	4.4.3
Extended Slave Profile for Combined Transaction type 1 (Extended Integrated Analog profile)	7	4	4.4.4
Combi Slave with support of combined transaction type 2	7	5	4.4.5
Safety Sensors with non-safe outputs	7	B	4.4.9
Motor Control Devices (electromechanical)	7	D	4.2.5
Motor Control Devices (semiconductor)	7	E	4.2.5
Dual Actuator with feedback	B	1	4.2.6
Single Actuator with monitoring	D	1	4.2.7

Tab. 6 Slave profiles with extended addressing [23]

Slave-profiles	ID2 Code															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	I, I, I, I	0A0		0A2											0AE	R
1	I, I, I, ⊕	1A0													1AE	R
2	I, I, I, B	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
3	I, I, O, ⊕	3A0	3A1	3A2											3AE	R
I/O	I, I, B, B	4A0													4AE	R
5	I, O, O, ⊕	5A0													5AE	R
6	B, B, B, B	6A0													6AE	R
C	B, B, B, B	7A0		7A2		7A5		7A7	7A8	7A9	7AA				7AE	R
o	O, O, O, O	8A0		8A2											8AE	R
d	O, O, O, I	9A0													9AE	R
e	O, O, O, B	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
B	O, O, I, I	BA0		BA2		BA5									BAE	R
C	O, O, B, B	CA0													CAE	R
D	O, I, I, I	DA0													DAE	R
E	O, B, B, B	EA0													EAE	R
F	T, T, T, T	For Future Use														

Tab. 7 Type of slaves with extended addressing [23]

Profile Name	IO	ID2	Chapter
Remote I/Os (X = 0 ... E, not 2, A)	X	0	4.3.2
Free profiles for Slaves in extended address mode (X = 0 ... E)	X	E	4.3.1
Remote I/Os with dual signals (X = 0, 3, 7, 8, B)	X	2	4.3.4
Single Sensor with extended control	3	1	4.3.3
Combi Slave with support of combined transaction type 2	7	5	4.4.5
4I/4O in extended addressing mode	7	7	4.4.6
Slave profile for combined transaction type 4 (single channel)	7	8	4.4.7
Slave profile for combined transaction type 4 (dual channel)	7	9	4.4.7
8I/8O in extended addressing mode	7	A	4.4.6
Slave with support of combined transaction type 2	B	5	4.4.5

4.5.2 Master Profiles

The master as well as the slaves can have the standard or the extended profile. The standard profile supports only communication with “A” addressing slaves up to 31. The extended profile supports communication with “A” addressing slave up to 31 or with 62 slaves with extended configuration (“A/B” addressing) [23].

Tab. 8 Type of standard master profiles v1.0 [23]

profile identifier	name	remark
M0	minimum standard master	only for data I/O
M1	full standard master	data I/O and parameter and all other functions
M2	reduced standard master	data I/O and minimum parameter functions

Tab. 9 Type of extended master profiles [23]

profile identifier	name	remark
M3	full extended master	data I/O and parameter and all other functions at host interface and support of Combined transaction type 1
M4	Version 3 extended master	M3 functionality plus support of Combined transaction type 2, 3, 4 and 5

5 FTZ AS-INTERFACE SLAVE SIMULATOR

5.1 Basic Information

The Slave Simulator is used as a testing device on fieldbus AS-Interface. The main task of this device is to simulate (imitate) the operation of optional amount of slaves up to 62 or the whole AS-i network. It includes communication of different slave configurations respectively profiles with master and their mistakes. In this way all the situations that might occur in reality can be simulated. Thanks to these devices proper functionality of the network as well as troubles and failures can be found.

The type of slave profile corresponds with its application. It means that this device has to represent and simulate the slaves as well and as accurate as possible. The Slave Simulator should be capable of slave profile configuration, monitoring of communication and showing of results. For these reasons a touch interface using the 4,3" Amulet LCD module has been created.

The construction of this simulator results from the upgrade of an already existing master-test device (ATTEST AS-Interface Master-tester). The simulator has been created at the FTZ Leipzig at University of Applied Sciences. Its construction relies on FPGA and it will be completed by touch screen and other functions will be added. This simulator should belong to the v3.0 generation devices working with AS-i.

In Fig.13 we can see the master-tester that has already been modified. Units marked with the red colour are the fields this project deals with.

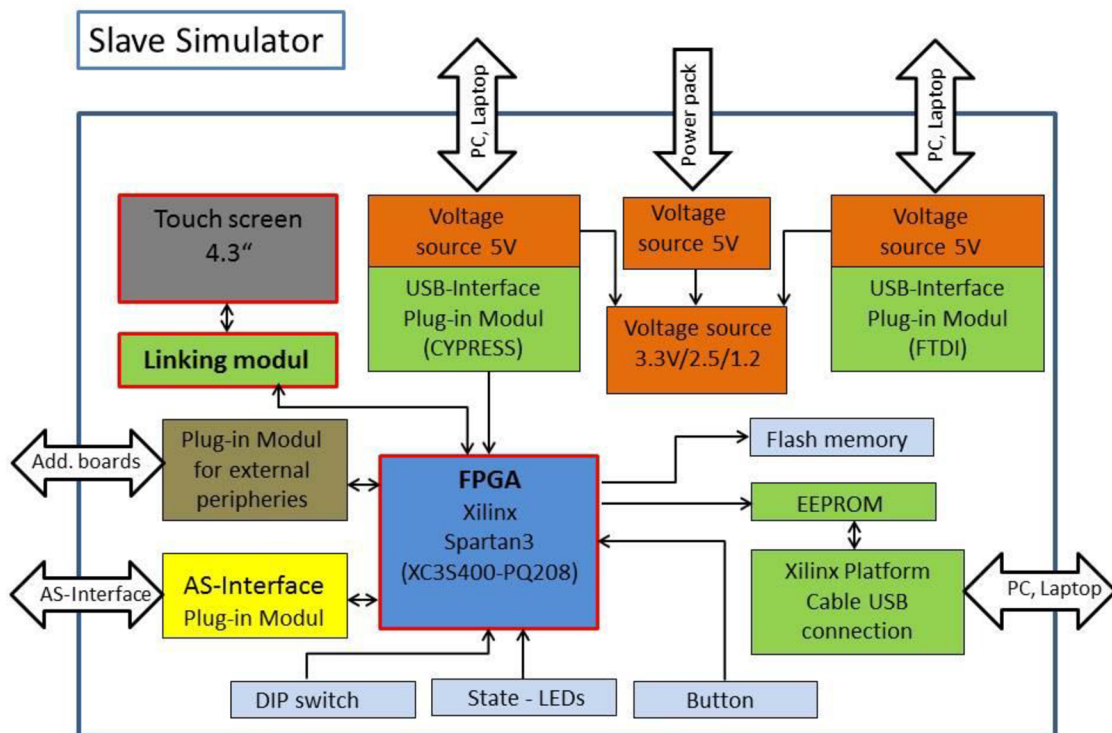


Fig. 13 Block diagram of Slave Simulator HW arrangement [25]

5.2 Amulet Resistive 4.3” GEMmodule™

As a touch display a starter kit Resistive 4.3“ GEMmodule™ Colour Display Module MK-480272 coloured embedded GUI (graphical user interface) by the company Amulet Technologies has been used. This fully integrated embedded product allows to implement great-looking GUI within a very short time period regardless of the size and speed of their microprocessor. This device is programmed by development tool GEMstudio via USB or UART. Programmed pages can be stored in internal flash memory [6].

This starter kit was already purchased by FTZ and therefore is used in this application.

The basic characteristics of the touch display [31]:

- Resolution 480X272 TFT LCD with a 16:9 aspect ratio
- Dot Pitch 0.198 x 0.198mm
- Amulet GEM Graphical OS Chip™
- Voltage/current supply: 5V/300mA
- Operation temperature -20 to 70°C
- White LED backlight
- Supported graphic formats JPG, PNG, GIF in 24-bit colour
- Integrated resistive touch panel
- 32MB Serial Flash for storing GUI pages
- 24 Pin interconnector
- Communication Interface: USB, RS232, UART
- Viewing Angle 12 o'clock
- Colour Supported – Up to 24bit + 8bit Alpha
- Price: 300\$



Fig. 14 Amulet Resistive 4.3“ GEM starter-kit [31]

5.3 Amulet Capacitive 4.3“ GEMmodule™ comparison

In the following chapter the capacitive STK-CY-043 has been described in order to compare it with the mentioned resistive display. This device has the same resolutions and similar features as the previous one but it is based on capacitive technology which brings following advantages. Usage of stylus is not required and the user can comfortably control this device just by finger touch. Moreover, the multi-touch feature empowers more functions, for instance scrolling or sizing pages. One of the noticeable disadvantages of this touch display is its price which is still approximately by 60% higher comparing with the resistive display [26].

The basic characteristics of the touch display [27]:

- Resolution 480X272 TFT LCD with a 16:9 aspect ratio
- Dot Pitch 0.198 x 0.198mm
- Amulet GEM Graphical OS Chip™
- Voltage/current supply (recommended): 5V/250mA
- Operation temperature -20 to 70°C
- White LED backlight
- Supported graphic formats JPG, PNG, GIF in 24-bit colour, BMP
- Integrated projected capacitive touch panel
- Multi-touch
- 32MB Serial Flash for storing GUI pages
- 24 Pin interconnector
- Communication Interface: USB, RS232, UART
- Viewing Angle 6 o'clock
- Colour Supported – Up to 24bit + 8bit Alpha
- Price: 487\$



Fig. 15 Amulet Capacitive 4.3 GEMmodule™ [27]

5.4 Linking Module

When it comes to the supply voltage range of the FPGA and the levels of voltage logic used for RS232 communication with Amulet LCD module, the usage of this linking module is not necessary. It is sufficient to connect wires Rx and Tx of LCD module and FPGA to cross to RS232 asynchronous communication.

Specification of voltage ranges:

- Supply voltage of FPGA:
2,5V and 3,3V
- RS232 logic levels of Amulet LCD module [6]:
Log 0 from -0,3V to 0,8V
Log 1 from 2V to $V_{cc} + 0,3V$ ($V_{cc} = 5V$)

The sense of this linking module can be found in the task of communication monitoring between Amulet and FPGA or, if we desire to test the communication, with the aid of a PC. In this case a module performing the conversion RS232/USB can be placed on this board. For this purpose the integrated circuit called FT232H Single channel USB to serial port can be used.

6 PROJECT DESIGN USING GEMSTUDIO

6.1 Introduction

GEMstudio is a development tool made for design of embedded GUI systems. It allows creation of a smartphone-like GUI design in short time without requirement of knowledge of a programming language as C++. It enables easily design, customize, test and implement a high-end user interface into all types of embedded products [11]. In my case this design tool is used for programming resistive touch display 4.3'' referred to as STK-480272C from the company Amulet Technologies as well as software GEMstudio.

The simplicity of programming embedded device by using this tool consists in graphical programming environment. It means that the bulk of programming work is to create the graphical design consisting of pages which can be further divided into objects. The design of the project in GEMstudio can be classified into three parts. Each of this part is described lower in detail.

The first task is to design the structure of the project. It means to create a network of pages, which are basic units of this graphical design. This page can be connected to each other without any rules.

The second part of the project is to put together every page consisted of listed object. As the object can be used buttons, pictures, lists, graphs etc. Every of these parts have specified functions and on top of that as a strong instrument can be used "Page function". This task is done in a simple predesign. The main point is to test the proper functionality of the project.

The last section is the final design. This should be done as much as user-friendly as possible. The layout of page should be well-arranged, intuitive and must consider simplicity and reliability to control the touch-panel.

6.2 Project Structure

In the beginning of the project it is necessary to ask what the embedded system application is going to do and what it will be used for. This touch display application is used as communication interface between user and the AS-Interface Slave Simulator. This communication interface should allow to the user these following functions:

- To add slave with its parameters
 - Name, address, online/offline status
 - Profile configuration: IO, ID, ID2, data, parameter, status
 - Selection of commands for command counter statistic
- Delete slave
- To see the topology (network) of slaves and to change slave settings
- Additional functions: information about application itself, time and date

After summarization of these points I designed the first block diagram of the project. Every box in Fig.15 represents one page of the interface graphic design. Arrows represent the direction of the movement between each page. In the Final block diagram (see Fig.16) there are not shown arrows from every page to the “Home” level for better lucidity. The possibility to go to Home level is from every page with exception of “Confirmation” pages.

The schema was changed over the time with findings of needs of each function. In the picture below it can be seen the first (grey) and the final (coloured) block diagram. These schemes were drawn in Microsoft Visio.

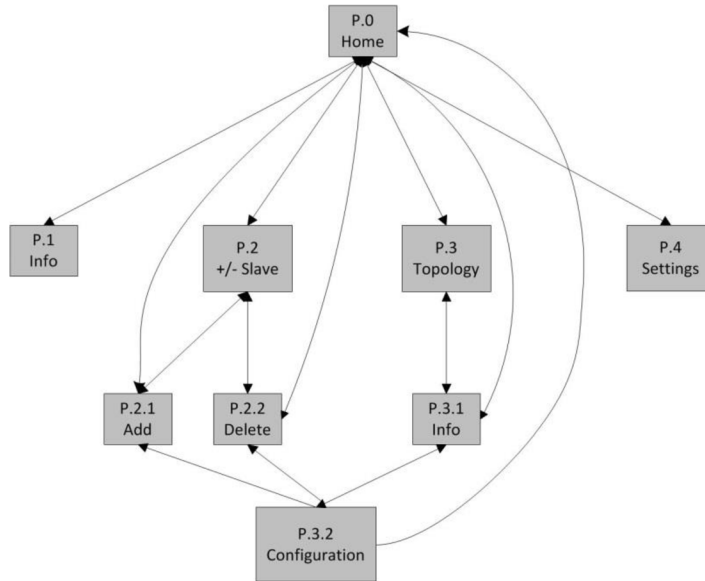


Fig. 16 Sketch of the project block diagram

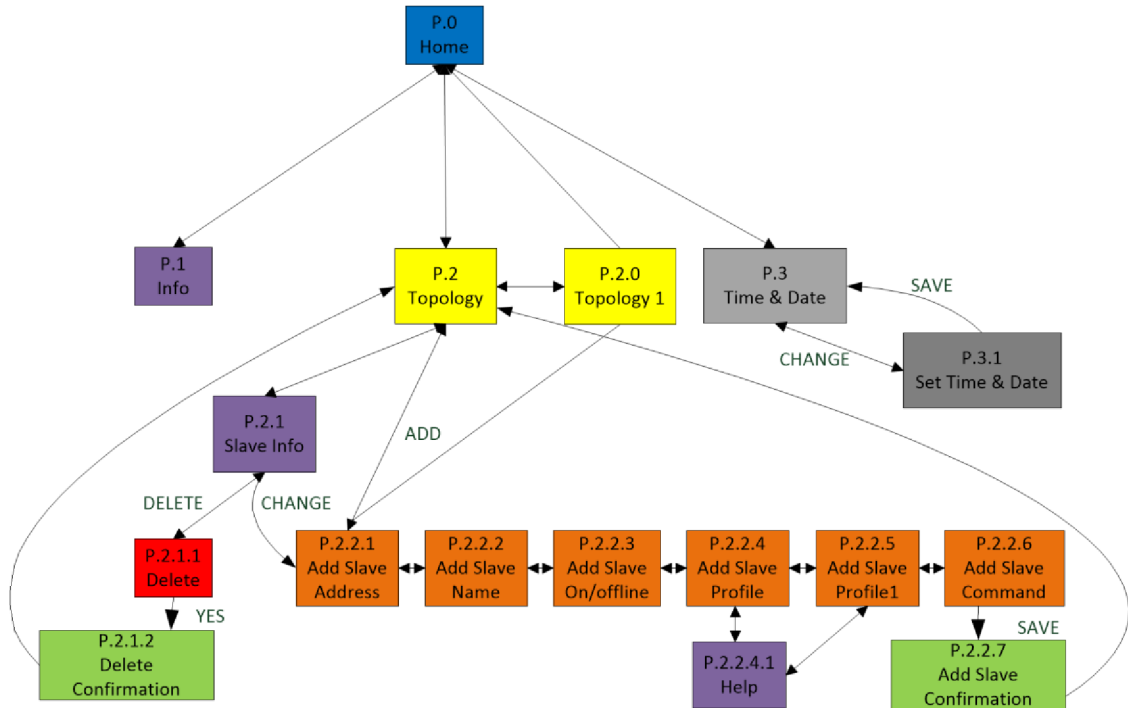


Fig. 17 Final block diagram of the project

6.3 Application Description

In this chapter individual pages and their facilities and capabilities from Fig. 17 will be described. Images of individual pages can be found on the page 44-46.

In the first page in the application three icons pop up of Home menu. First of them from left is called "Information" (P.1). Here are written information about this application as a version, date of last update, service contact, name of the software producer and designer.

First from right is icon which leads to page "Time & date" (P.3). Here appear actual system time and date. It can be adjusted by clicking to the set button. This time and date is programmed in simple way which is detailed described in chapter 6.4.4 as an example of programming way. The reason why it is placed in this application is to demonstrate features of Amulet LCD module and it can be used as well as screensaver.

In the middle of Home page is placed icon of "Topology" page (P.2). This icon leads to the centre of the application. In page Topology we can see already existing slave which are represented by grey button with slave address. If a slave is in "Online mode" a green indicator appears next to the button. Slaves are arranged along yellow line representing AS-Interface. In the first page of topology P2 we can observe slaves with address from 0 up to 31. By clicking on the button to right direction the page "Topology 1" (P.2.0) appears where are displayed slave with address from 32 up to 62.

The plus button placed in the Topology page leads to "Add slave menu" or "Creation of a new slave" which consists of six steps. First step it to add slave address in range 0-62 (P.2.2.1). Next step is to name the new slave. Limitation of it is 15 characters. Online/Offline mode setting follows in page P.2.2.3. Next step is to set configuration of the profile described in chapter 4.5 "AS-i device profiles". Profile configuration by following parameters can be performed: I/O, ID, ID1, ID2, Data, Parameter and Status. Every of these parameters have 16 different possibilities to be set. Not every of it has to be used and the meaning of this configuration is explained in "Help" page (P.2.2.4.1).

As the last step of the new slave creation is the setting of command sensitivity in page (P.2.2.6). Here the user can choose commands which are later on showed in the command counter statistic of page "Slave information" (P.2.1). Then, if the "Save" button is pressed, the setting of parameters is saved into the memory. Not earlier. It means, when user jump out from Add slave menu without pressing Save button in the last page, the change of configuration will not be performed.

By pressing certain "Slave" button in page Topology, the page "Slave info" (P.2.1) appears. As a main feature of this page is to show the command counter statistic of certain commands (see Fig. 42) and possibility to change or delete actual slave. By pressing the "Change/tooth-wheel" button appears the Add new slave menu which already contains pre-set configuration. Change of the configuration has to be confirmed by Save button in last step of Add new slave menu.

By pressing "Delete" button in Slave info page the "Delete" page appears with slave address and name. To confirm the delete of a certain slave the "Confirmation/check" button has to be pressed. All of slave configuration settings are deleted from the memory.

6.4 Project Design

In this part will be described in detail the design of the project in GEMstudio. We can start with creating a new project. In the first page will GEMstudio demand LCD size, LCD manufacturer, LCD part number and board name. In my case the answer is in the same order: 480x272, Amulet, STK-480272C, 32M_flash_64M_sdram.

After confirmation we can start to create first page by pressing *plus button* and choosing a *new page*. A design in GEMstudio is very intuitive. In the middle of the programming environment is placed the screen of the touch display which we are programming. In the page we can place different objects in the same way as a new page. The most often object in my project are: Picture, Static Text, Background and Widgets CustomButton, FunctionButton, List, NumericField, String Field. It is possible to place the object anywhere we want and change its size. Each of the objects has predefined list of functions. We can choose which ones we want to use and which not by *add/remove parameter*.

On the left side of the programming environment is placed a list of the designed pages which contains its objects. See Fig.18 below. When we click on the object, we can see the list of parameters which can be defined. As a name of the object, colour, size etc. One of the most important parameters is the “Href” function which connects these objects among each other. In the Href function we can use just predefined function specific for each object. To see all of allowed commands see Appendix C.

For example when we set the Href function of CustomButton as: *Amulet:internalRAM.byte(247).setValue(0)* => byte with address 247 of the “InternalRAM” of the Amulet system set to value 0. When we will set the Href function of “NumericField” as: *Amulet:internalRAM.byte(247).value()* => show value of byte 247 of amulet internalRAM. The connection is set and every time when we will press the CustomButton the “0” will appear in NumericField.

Another very strong instrument and often used is “Page function” (Right click on the page). As the name says itself, this function is called in the same time when the page is opened. To describe more exact parameters of the Page function and to use e.g. conditions thereby so-called “META refresh object” is used which has the exact structure. Because this object is often used in this project therefore it is in detail described below.

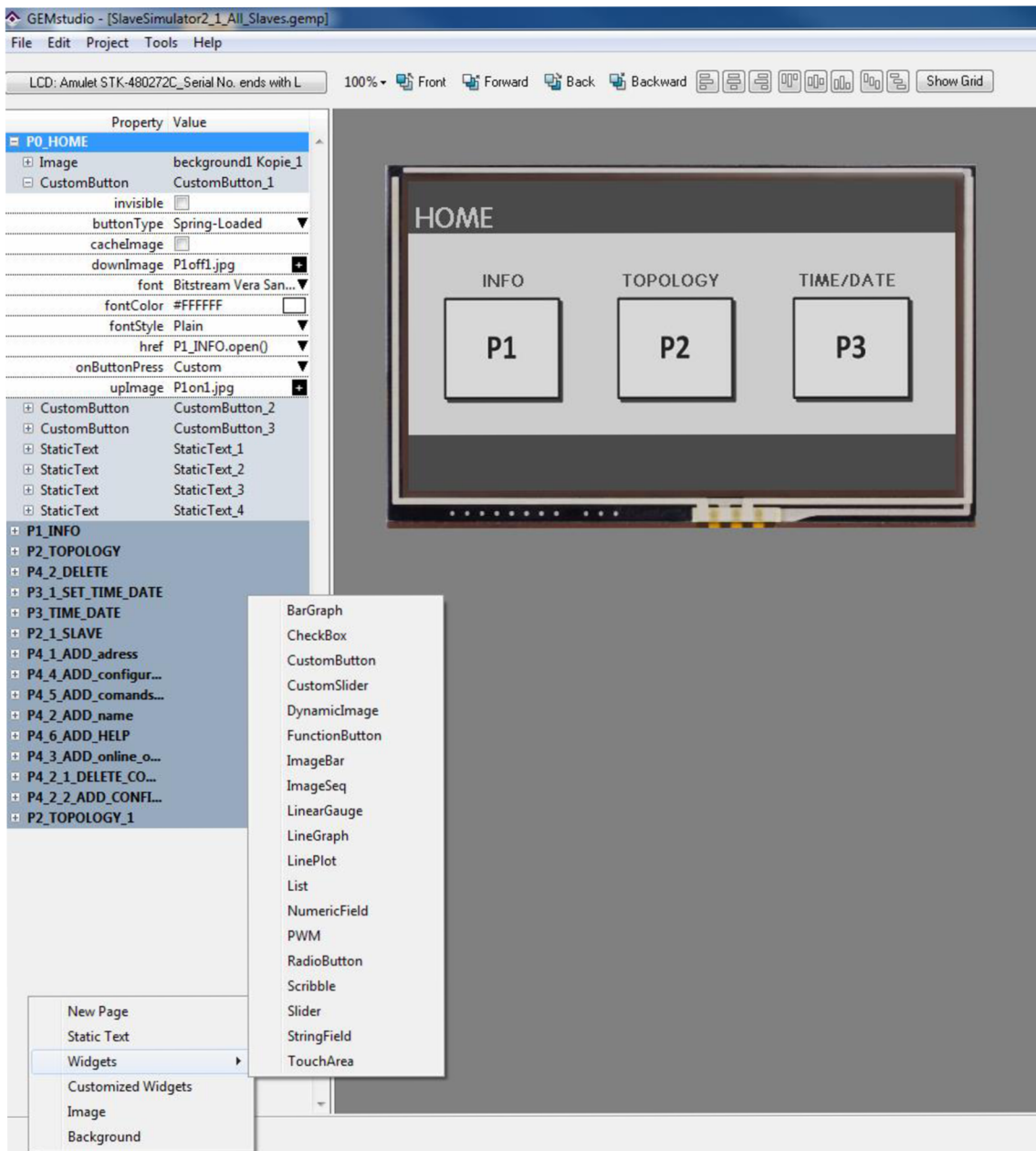


Fig. 18 GEMstudio programming environment

6.4.1 META Refresh Object

There are four different ways how to use this tool. Two of them are listed according to [32], the ones which were applied on my project.

- 1) Call function/s based upon a timer event.
- 2) Call function/s when a timer-based function returns a specific value (if...then...else function).

Ad 1) Exact structure is given in form:

```
<META HTTP-EQUIV="REFRESH" CONTENT="1st number, 2nd  
number;URL=function(s);VALUE=number;NAME=string">
```

- *1st number*: it specifies the frequency of called URL function (in seconds) in the range of 0.00-655.35. In case of using 0.00, it updates just once (2nd number must be specified).
- *2nd number*: it specifies the time delay between the loading of the page and the calling of URL function (in seconds) the range is 0.01-655.35. If the 2nd number is not specified, then the delay time defaults to the 1st number (frequency) value.
- *URL*: only listed commands can be used here. See Appendix B.
- *VALUE*: it is intrinsic value of this meta refresh object (not used in case of this design).
- *NAME*: it specifies the internal name of this object (not used in case of this design).

Ad 2) Exact structure is given in form:

```
<META HTTP-EQUIV="REFRESH"  
CONTENT="updateRate, delayRate;  
IF=function;  
{EQ | GT | LT | NEQ}=value;  
THEN=function(s);  
ELSE=function(s);  
NAME=string">
```

Description of the Meta tag:

1. *If function* is called with defined frequency and time.
2. Returned value of *if function* is compared with defined *value* by EQ - equal, GT - greater than, LT - lower than, NEQ - not equal.
The *value* can be byte, word, InternalRAM byte variable or word variable.
3. If the comparing is in order, then is called function defined behind *Then*, if not, is called *Else function*.

6.4.2 Amulet InternalRAM

To understand codes and commands it is necessary to know more details about InternalRAM. Amulet has over 1 kB of onboard RAM. There are 256 different byte variables, 256 different word (16-bit) variables, 256 different colour (32-bit) variables, and 256 different 25-character null terminated string variables (25+1=26 bytes allocated per string variable). The amulet Widgets can read or write to these InternalRAM variables. The InternalRAM variables can also be saved in flash, thus giving the variables permanence. In addition, an external processor can read and write to these InternalRAM variables as well. The external processor can send an unsolicited serial message to the Amulet to read or write to the InternalRAM variables. This means that the external processor is not required to be the Amulet's slave [12].

Some of the main features of InternalRAM [12]:

- 1) InternalRAM keeps variables from page to page of the project.
- 2) InternalRAM can be saved back to the flash, so the variable can persist even after powering down.
- 3) InternalRAM variables can be used as arguments within Amulet methods. i.e. (Amulet:UART.byte(0).setValue(InternalRAM.byte(2)))
- 4) InternalRAM variables can be used as variable indices. i.e. (Amulet:UART.byte(InternalRAM.byte(0)).setValue(2))

6.4.3 Usage of InternalRAM

Tab. 10 Variable usage of InternalRAM

Type of memory unit	Address	Usage
byte	255	Actual slave address, used for determination of storing address
byte	248	Help variable of counting RAM address of slave profile configuration
byte	247	Actual value if slave is set to online(1) or offline(0)
byte	246	Help variable, if slave is set to online then it is equal to actual address of slave, if is offline, then it is added 64 to actual address
byte	244	Actual mode: change slave(1), add slave(2), change and slave addition is finished(0)
byte	206	Variable of Image Sequence in the Help slave profile configuration
byte	205	Edit flag of time and date, (1) time and date was changed
byte	200-204	Save actual and then modified value of hour, minute, seconds, day, month
byte	0-191	Save slave x profile config. of IO,ID,ID2,Data,Parameter, Status to 3 bytes
string	0-62	Save slave x name
string	64	Actual slave name
string	100	Actual slave address
string	101-164	Initialization of slave x address in change mode
string	165,166	Date of last update, number of version
word	0-62	Save slave x command setting
word	64	Actual slave command
word	68-79	Command counter showed values in page Slave info
word	200	Save actual and then modified value of year
word	238-244	Help variable in order to show actual value of profile configuration
word	245,237	Actual setting of slave profile config. IO,ID,ID2,Data,Parameter, Status
word	246	Help flag for showing actual status of On/offline button
word	247	Save actual and then modified the value of the year
word	248-251	Set bit to (1) if slave was created to the LSByte of the word 248 for slave 0, MSByte of the word 251 for slave 62
word	252-255	Set bit to (1) if slave is online, the same rule as previous line

6.4.4 Program Description

For simplicity few examples are given where it is shown how the previously mentioned tools of GEMstudio are used. This Slave Simulator interface is designed for 62 slaves and the code is often the same for each slave.

Program can be divided into two main parts. Page function is referring to certain page, usually there are used Meta refresh objects and Href function is referring to certain object respectively the Widget in designed page.

Because in this way of programming a programmer can use just given commands, no loops or cycles (see Appendix B), therefore solution of some codes seems to be very complicated in order to achieve simple goals. This is the biggest disadvantages of programming in GEMstudio. It is already improved in new version of GEMstudio Pro version 2.0.0.0 which was published during this project. Here is implemented a new tool called **GEMscript** which markedly enhances the processing capabilities as using of macros, functions, mathematic operators, cycles, switch-case, nested if and if-else statement etc.

Let's focus on Page function. For example in the "Time and date" page (see Fig.20). The program of this Page function is described below. There we can easily see how this tool works. With a loading of the page the initialization of variables is performed. It represents time and date. Immediately we can see the actual values of time and date displayed on the screen. If this page is loaded after changing the time and date in "Set time and date" page, this initialization will not be performed via the starting "If condition" and we will see modified values. Blinking of colon between hours and minutes is done via CustomButton where we can allow appearing and disappearing "Up" and "Down" image on the screen. Up image is just grey, and Down image is black colon on grey background. This change of pictures is made every second.

The clock program is programmed very simple without leap years and it makes no differences between months with 30 and 31 days. All of the months are programed for 30 days. Clock program is programmed by using "If condition" with overflow limit. For example if variable representing seconds achieves value 60 then the variable of minutes will be increased by 1 and the second will be equal to 0 and so on for another values of time and date.

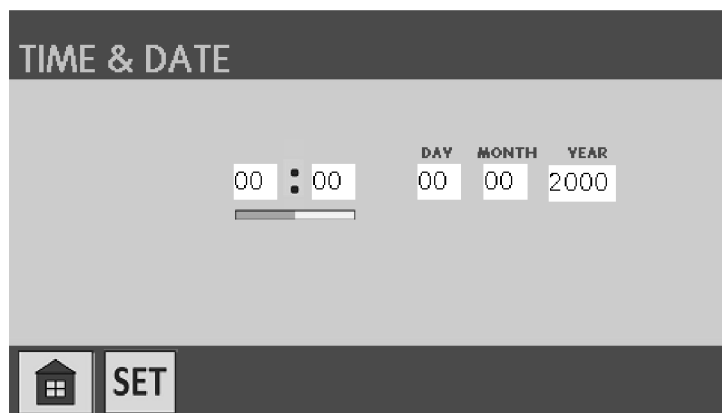


Fig. 19 Time and date page

```

/****TIME AND DATE*****/
/******/
/*INICIALIZATION*/ save actual value of time and date, if is not changed
<META HTTP-EQUIV="REFRESH" CONTENT="0.01,0.01;
    IF=Amulet:internalRAM.byte(205).maskedValue(0x01);EQ=0; no change of condition flag
    THEN= load the value of actual time and date running in processor
    Amulet:internalRAM.byte(200).setValue(Amulet:calendar.militaryHour.value()),
    Amulet:internalRAM.byte(201).setValue(Amulet:calendar.minute.value()),
    Amulet:internalRAM.byte(202).setValue(Amulet:calendar.second.value()),
    Amulet:internalRAM.byte(203).setValue(Amulet:calendar.dayOfMonth.value()),
    Amulet:internalRAM.byte(204).setValue(Amulet:calendar.month.value()),
    Amulet:internalRAM.word(200).setValue(Amulet:calendar.year.value());">

/*blinking of colon */every second the up und down custombutton picture is changing
<META http-equiv="Refresh" content="1,1;
    URL=Amulet:document.CustomButton_8.buttonDown();">
<META http-equiv="Refresh" content="2,1;
    URL=Amulet:document.CustomButton_8.buttonUp();">

/*CLOCK PROGRAM
/*-----*/
/*seconds*/
/*frequency*/ every second add 1 to seconds
<META http-equiv="Refresh" content="1,1;
    URL=Amulet:internalRAM.byte(202).add(1);">

<META HTTP-EQUIV="REFRESH"
CONTENT="0.01, 0.01;
IF=Amulet:internalRAM.byte(202).value();
EQ =60; if value of second is equal to 60, set it to 0 and add 1 to minutes
THEN=Amulet:internalRAM.byte(202).setValue(0),
Amulet:internalRAM.byte(201).add(1);">

/*minute*/
<META HTTP-EQUIV="REFRESH"
CONTENT="0.01, 0.01;
IF=Amulet:internalRAM.byte(201).value();
EQ =60; if value of minutes is equal to 60, set it to 0 and add 1 to hours
THEN=Amulet:internalRAM.byte(201).setValue(0),
Amulet:internalRAM.byte(200).add(1);">

/*hours*/
<META HTTP-EQUIV="REFRESH"
CONTENT="0.01, 0.01;
IF=Amulet:internalRAM.byte(200).value();
EQ =24; if value of hours is equal to 24, set it to 0 and add 1 to days
THEN=Amulet:internalRAM.byte(200).setValue(0),
Amulet:internalRAM.byte(203).add(1);">

/*days*/
<META HTTP-EQUIV="REFRESH"
CONTENT="0.01, 0.01;
IF=Amulet:internalRAM.byte(203).value();
EQ =30; /*simplified!*/ if value of days is equal to 30, set it to 0 and add 1 to months
THEN=Amulet:internalRAM.byte(203).setValue(1),
Amulet:internalRAM.byte(204).add(1);">

```

```

/*month*/
<META HTTP-EQUIV="REFRESH"
CONTENT="0.01, 0.01;
IF=Amulet:internalRAM.byte(204).value();
EQ =13; if value of months is equal to 13, set it to 0 and add 1 to years
THEN=Amulet:internalRAM.byte(204).setValue(1),
Amulet:internalRAM.word(200).add(1);">

```

When we focus on the Href function, the program in this case is not usually very complicated as it was in the case of Page function. Let's focus on page "Set time and date" (Fig 19), we can see below the Href functions. The commands contain the Href functions of object "Home", "Check", "Previous page", "+/-" button and NumericField widget which projects values of time and date. In this case simple functions are called as: open page, set value, add or decrease value. Functionality of this tool is simple. When we press the button, the Href function is called. All program description can be found in Appendix P1.

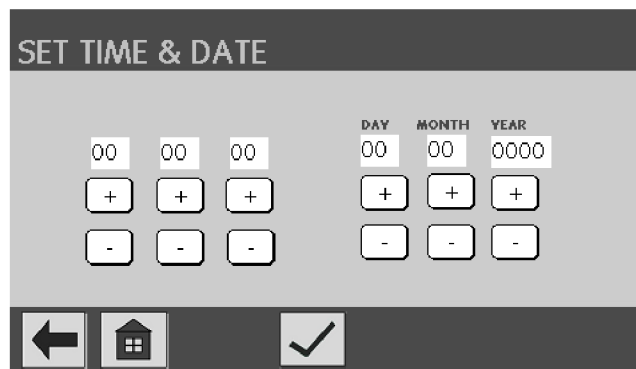


Fig. 20 Set time and date page

*****SET TIME/DATE*****

-----home button-----

P0_HOME.open() open Home page

-----check button-----

P3_TIME_DATE.open(), open page Time and date

Amulet:internalRAM.byte(205).setValue(0x01)

Set flag of time change to 1

-----back button-----

P3_TIME_DATE.open(),

----- + - minutes button -----

Amulet:internalRAM.byte(201).add(1) add 1 to byte of minutes

Amulet:internalRAM.byte(201).dec(1) decrease by 1 the byte of minutes

----- numeric field -----

Amulet:internalRAM.byte(201).value() show actual value of byte 201 with set rate of update

6.4.5 GEMscript

As it was stated in the previous chapter, the GEMscript of the new programming environment GEMstudio Pro v2.0.0.0 enables prominent simplification of the way of programming and enhances capability of the previous GEMstudio version. To compare differences between programming using META refresh object and the GEMscript, the code of Clock program, which was mentioned in the previous chapter, is displayed.

Example:

Initialization of variables is omitted in the code.

```
/*CLOCK PROGRAM
/*-----*/
<META HTTP-EQUIV="REFRESH" CONTENT="1;
URL=GEMscript.time_1();> refresh/call GEMscript "time_1" every second

<script> start of public GEMscript "time_1"
public time_1 ()
{
    second = second+1;

    if (second == 60)
    { second = 0;
      minute = minute + 1;}
    if (minute == 60)
    { minute = 0;
      hour = hour + 1; }
    if (hour == 24)
    { hour = 0;
      day = day + 1; }
    if (day == 30)
    { day = 0;
      month = month + 1; }
    if (month == 13)
    { month = 0;
      year = year + 1; }

    internalRAM.byte(0)=hour; assignment of variables to the memory
    internalRAM.byte(1)=minute;
    internalRAM.byte(2)=second;
    internalRAM.byte(3)=day;
    internalRAM.byte(4)=month;
    internalRAM.word(0)=year;
}
</script> end of the GEMscript
```

6.4.6 Main Program Description

As the main part of the program it can be consider the configuration of a new slave or change of the slave settings.

During the configuration of a new slave, settings are stored in actual variables in every page. Only when the Save button is pressed in the last page (P.2.2.7), then the actual variables are stored into the memory. It avoids the change of settings before finishing the process of slave configuration.

If we want to change already existing slave, stored parameters have to be loaded from memory to show the already existing settings of parameters. This loading of data is often performed one page before. Backward movement in the Add/change slave menu is considered as well. That's why e.g. for page P.2.2.4 the initialization is performed one page before and one page after in order to show the right values for forward and backward movement in the menu (see

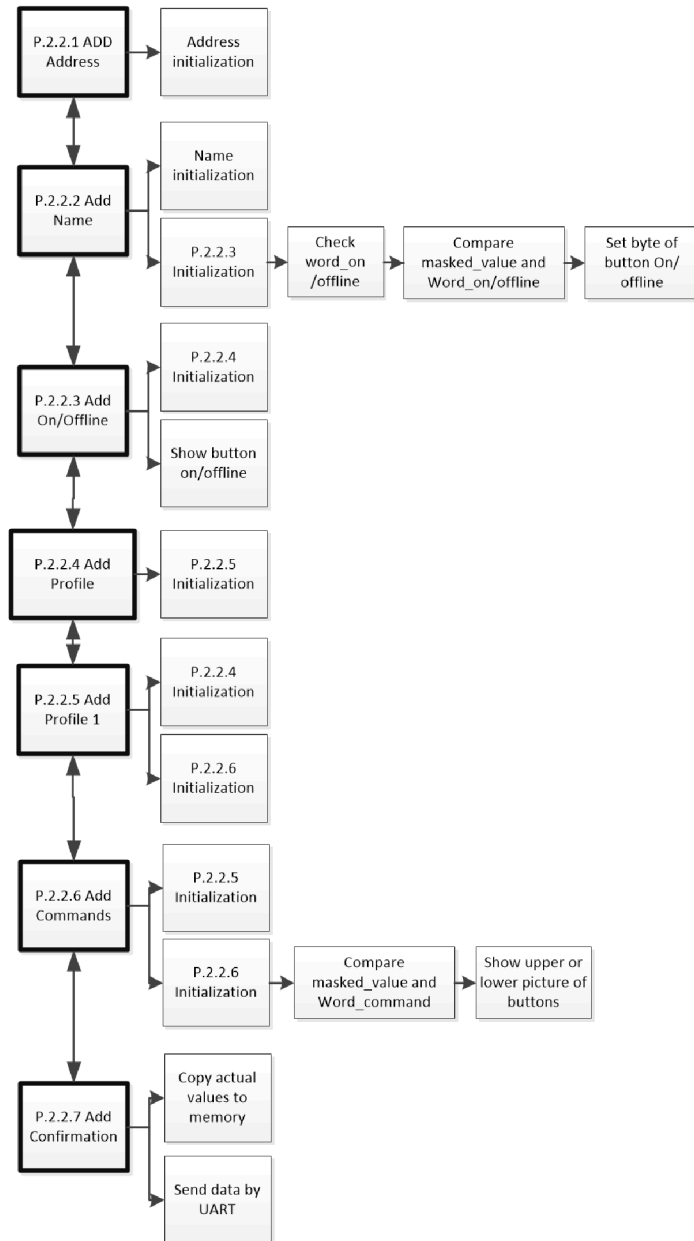


Fig.21). In this figure you can see which process is performed in which page of the menu. These processes are performed with help of the Page function.

Fig. 21 Block diagram of Add new slave menu

If the creation of the new slave is performed, the default values are set as address, name and offline status. The other parameters as the slave profile configuration and the commands settings keep values of last created slave. It can simplify setting of similar slaves after each other.

6.4.7 Other Page Processes

Page Topology

By every start of the page Topology the test of “Slave_is_created” flag and “Slave_is_online” flag is performed (see Fig. 22). Created flag and online flag are represented by one bit in an appropriate word. This word is tested for a masked value and then compared with a pre-set value.

E.g. the creation of Slave 0 represents least significant bit of a word with the address 248. This word is tested by masked value ‘0001’ and compared with pre-set value ‘1’. If the condition is true, then the icon of Slave 0 appears in the page Topology. The same principle is used for signaling where the slave is in the online status by appearing green small square next to the slave icon (see in Fig. 31). This is done separately for all 62 slaves.

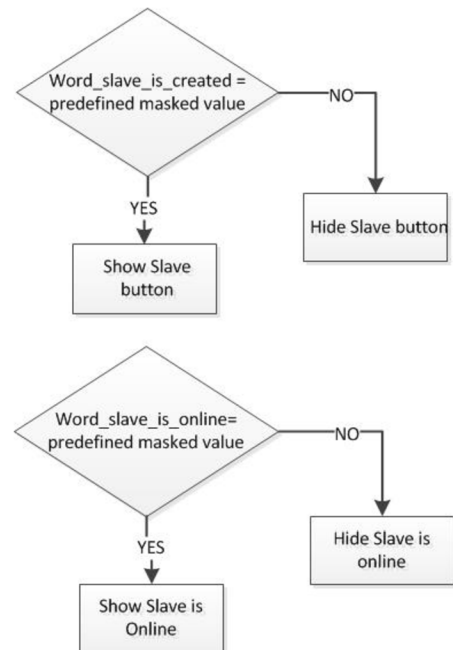


Fig. 22 Block schematic of process in Topology page

Program description:

Show Slave 0 button if the slave 0 is created

```

<METAhttp-equiv="Refresh" content="0,0.01;
IF=Amulet:InternalRAM.word(248).maskedValue(0x0001);EQ=1;
THEN=Amulet:document.b_0.reappear();
ELSE=Amulet:document.b_0.disappear()
">
  
```

Page Delete Slave Confirmation

Block diagram showed in Fig. 23 represents task performed in Page function of page Delete slave confirmation. Deletion of actual and saved data is necessary to ensure the validation of a new slave configuration which would have the same address as already deleted slave. In simple terms, it is the reset of allocated memory space.

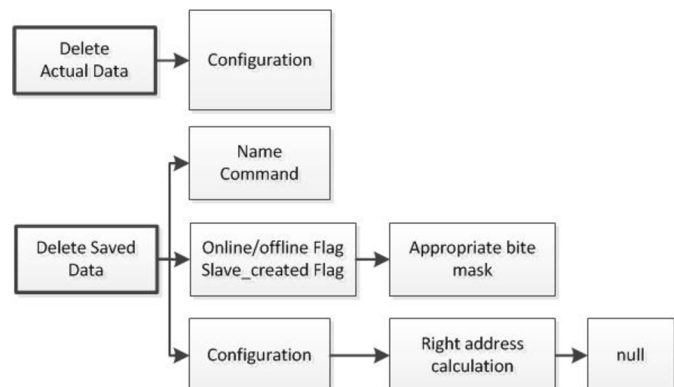


Fig. 23 Block schematic of process in Delete page

E.g. deleting of profile configuration data precedes the address calculation. Data of profile configuration are saved in memory in three bytes after each other.

7 GRAPHIC DESIGN

As it was already mentioned above, this part can be split into two sections. Draft was very simply designed in Microsoft Visio in order to test the application's functionality. In this point it is not reasonable to spend time with an accurate design which can be modified very soon by reason of application's adjustments. After successful testing the final design was performed by Adobe Photoshop. This software is very powerful and it allows making a well-looking design by few steps.

7.1 Draft

The layout of the sketch was designed with an idea to keep the lucidity of the touch application as good as possible. From this reason, the layout of the draft was divided into the three horizontal bands. In the upper part of the touchscreen there is located the "Label band". Here appears a page name of the application where the user is located. In the middle of the screen there is placed band which we can call the "Body" of the application or let's say content. Here appears the main information. In the lower part of the screen there is placed "Control band". There are all the basic buttons used for a movement in the application itself such as Next/Previous page, Home and some additional functions such as Save slave settings etc. See below in Fig. 24 the page division.

The colours were selected in very neutral tones with small exceptions. I used different hues of grey colour. Their contrast was used for better perspicuity.

The buttons are designed in contrast with the background. Their content is simple and sometimes it is complemented by label to clarify the meaning. In the final design the additional labels are replaced by better graphical design.

For signalization of pressing the button is used effect of reducing the size of the button and it changes the hue of the button picture. This allows the object CustomButton in the program GEMstudio where we can use "Up" image which appears normally and "Down" image which appears due pressing the button. See in Fig. 25.

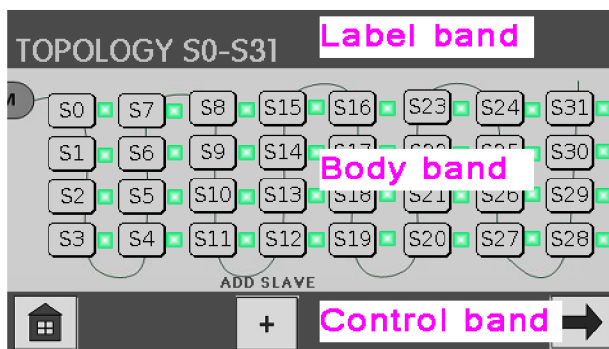


Fig. 24 Page division



Fig. 25 Home button Up and Down-image

As it was noted before Microsoft Visio was used for the sketch of the graphic design. This software is suitable for work with block diagrams or to design a simple shaped object without using many graphical effects. Working in this program is very intuitive and fast. See picture below of Microsoft Visio working environment.

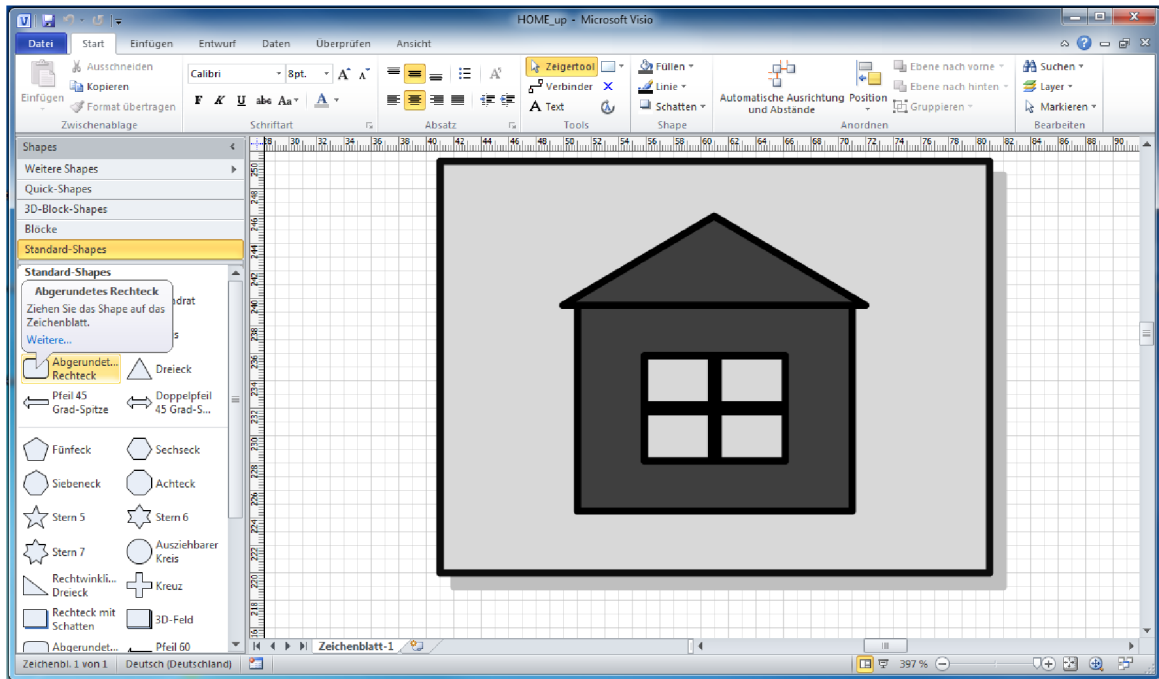


Fig. 26 Microsoft Visio work environment

7.2 Final Design

The tree band concept of the draft was kept. Just the size of the bands was adjusted more for their purpose. As the primary colour, the light hue of grey colour was chosen which appears in neutral zones. The colours of headlines were mostly chosen in the sense of colour theory (described in detail below), where the colour is expressing or approaching to the content of the page. For example the page “Delete slave” is designed in red colour to express the warning. The same I used for two following bands of the page. This meaning of colours should help to understand the application, the orientation and its usability which is based on the intuition. The user should quickly comprehend the concept of the application during its using. Hence in the final design there are missing additional labels describing the meaning of the buttons. According to it the icon itself should be so well designed that the user understands the meaning of it without any additional text [15].

7.2.1 Colour Theory

As well as the colour is important in our everyday life, the same importance has colour in a graphic design. The biggest contribution of colour is that it evokes feelings, emotion, association, reaction. What kind of meaning certain colour has, it depends on different aspect.

The colour theory results from the natural meaning of colours. The meaning and the understanding of colours depends on the cultural background [24] and it can be very personal. The basic colours used in my application according to [14] [15] and from my point of view of a Central European citizen are listed below. See the final graphic layout in three following pages.

Red colour is associated with fire, violence, warfare and as well with love and passion. In our society this colour is considered as colour of warning or stop state. Therefore I designed the page Delete slave in the red colour. In this case it should evoke more attention to the user.

Orange colour can be associated with earth, health and vitality thanks to the fruit of the same name. It can also represent autumn, change and movement. Due to this reason in my graphic orange colour is used for the “Next page” and “Previous page” button and for pages “Add slave” where there are assigned functions and parameters to the new slave.

Yellow colour is selected for label and wire in page Topology because it is representing the colour of AS-Interface.

Green is associated with nature, beginnings and growth. In our society green is connected with successfully working system, confirmation or permission. Due to this theory I selected this colour for “Confirmation” pages, for online status of the slave and “Add slave/plus” button in Topology page.

Blue usually symbolizes the colour of calmness, peace and responsibility. Hence it is colour of the main menu respectively label and button of “Home”.

Purple is in our society connected with wealth, royalty and wisdom. In the applications it is a colour of “Help” which is a source of information respectively source of wisdom.

Grey in normal life is depressing and moody colour but in graphic design it is generally connected with conservative, formal and modern style. I choose light tone of grey colour with one slow median angle to give a little bit of movement, elegance and modern looking style to the background.

Black is mostly considered as the strongest colour of neutral colours, commonly associated in positive side with power, elegant and formal design, in negative side with evil, death, sadness and mystery. Black colour is in the design used generally for common text and labelling mainly for the purpose of good contrast with light background.

White colour is used very sporadically in the design, especially for the good contrast with black and other darker tones of colours. This colour can be also connected with purity, health care industry and goodness.

7.2.2 Final Look



Fig. 28 Home menu (P.0)

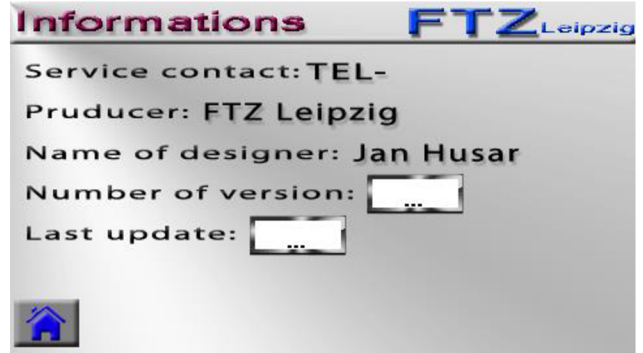


Fig. 27 Information (P.1)



Fig. 29 Time and date (P.3)

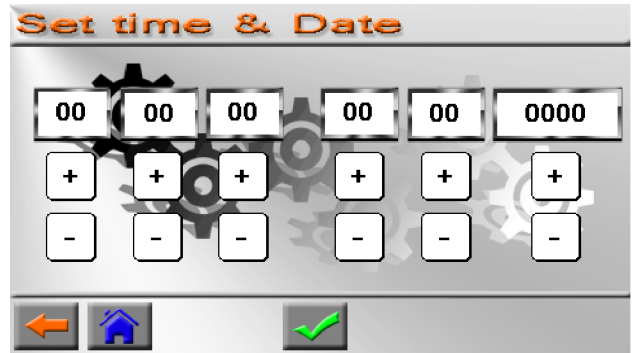


Fig. 30 Set time and date (P.3.1)

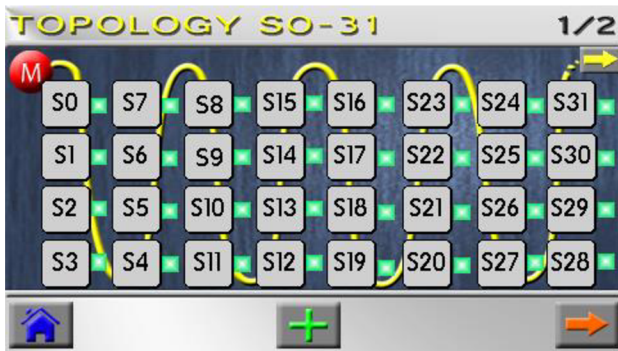


Fig. 31 Topology (P.2)



Fig. 32 Topology 1 (P.2.0)

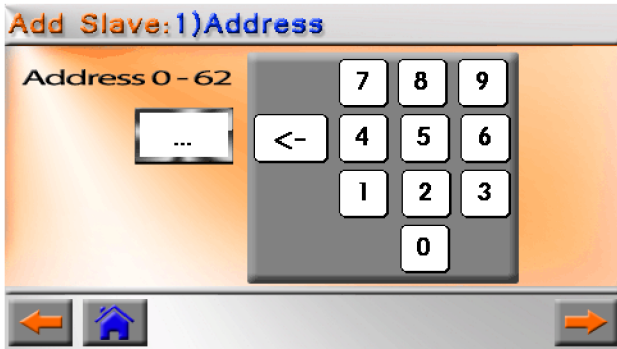


Fig. 33 Add slave address (P.2.2.1)



Fig. 34 Add slave name (P.2.2.2)

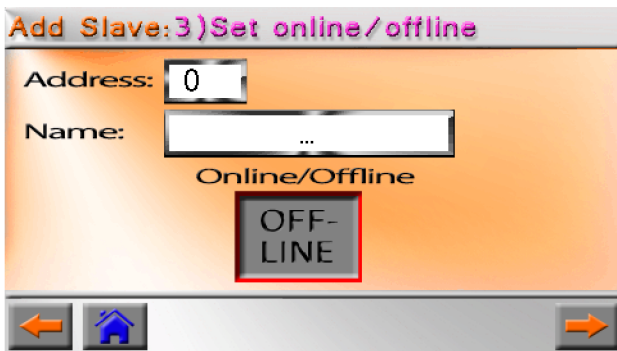


Fig. 38 Set slave online/offline (P.2.2.3)

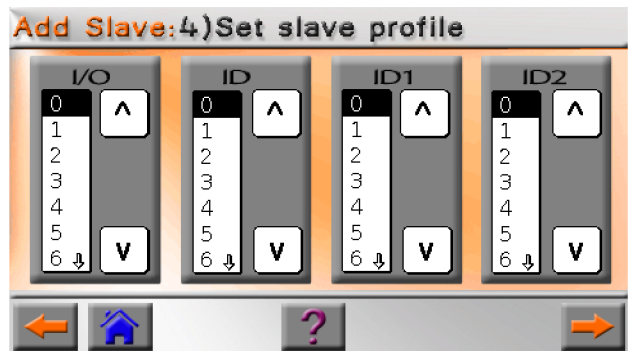


Fig. 37 Set slave profile (P.2.2.4)

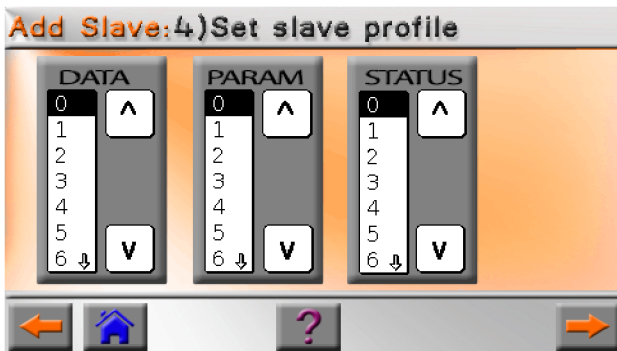


Fig. 36 Set slave profile (P.2.2.5)

Help: Slave configuration settings

I/O-Konfiguration	Slave-Profiles										ID-Code															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	A	B	C	D	E	F				
0	1111	00	01														0A	0B				0FF				
1	1110	10	11														1A					1FF				
2	1118	20															R					2FF				
3	1100	30	31														3A					3FF				
4	1188	40															4A					4FF				
5	1000	50															5A					5FF				
6	8888	60															6A					6FF				
7	8888	70	71	72	73	74	75										7A	7B		7D	7E	7FF				
8	0000	80	81														8A					8FF				
9	0001	R															9A					9FF				
A	000B	A0															R					AFF				
B	0011	R	B.1														BA					BFF				
C	00BB	C0															CA					CBFF				
D	0111	R	D.1														DA					DFF				
E	0BBB	E0	1.1														EA					EFF				
F	TTTT																					FFF				

← Home ? →

Fig. 35 Help slave configuration (P.2.2.4.1)



Fig. 40 Choose command sensitive (P.2.2.6)



Fig. 39 Confirmation slave added (P.2.2.7)

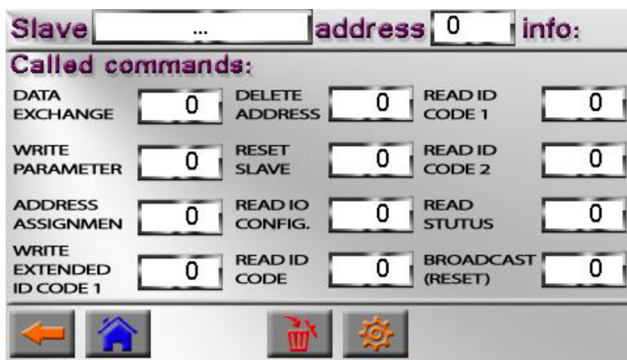


Fig. 42 Slave info (P.2.1)



Fig. 41 Delete slave (P.2.1.1)



Fig. 43 Confirmation slave deleted (P.2.1.2)

7.2.3 Photo Shop

The final design was made in Adobe Photo Shop. The reasons of my decision for this software were big amount of graphical tools and effects of this program. In the same time regarding the popularity of this program I consider it as the best choice. Therefore there is a very good online support [13]. The Photoshop is very complex program and contains countless numbers of tools as transformations, filters, masking, colour adjustment etc. For my purpose it is not necessary to know all the functions.

The work with layers is one of the basic tools, which is necessary to know. The designed picture is based on layers, which are covered by each other. The opacity can be set via different effects to this layer. The opacity represents the level transparency of the layer. I used this tool in all backgrounds for smoother and elegant look. Then I adjusted colours by increasing or decreasing certain colour. In order to obtain warmer tone of the orange colour, we can decrease the saturation of red and yellow in the certain area with help of a mask. Masks are very helpful tools if we want to work just with certain area or to cut an object from one picture and move it to another one. I used this tool for logo of clocks, settings, or picture of radio-clock etc. After masking, the chosen object can be covered by colour or gradient, shadows, satin etc. I worked mostly with this features for a 3D look of the buttons with inner shadow. By pressing the button the shadow is hanged by 180 degree which makes an effect of pushing the button. In order to make this effect more visible, I used deformation or reduction of the contained object in the button (e.g. icon Info Fig.45) and in the same time, size change of the inner shadow which makes effect of pressing bowed button.



Fig. 45 Information button - deformation effect



Fig. 44 Topology button – size effect



Fig. 46 Home button – shadow effect

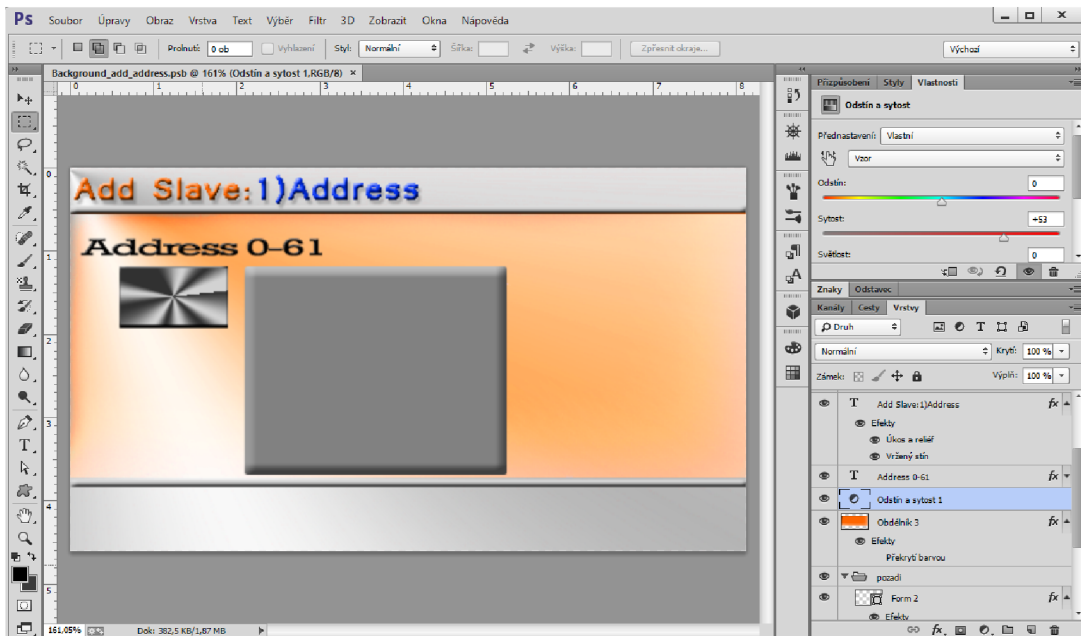


Fig. 47 Photoshop work environment

7.2.4 Fonts

For this touch application the font selection is important as well. The each size of the text has to be clear and distinct for a user. We have to consider the big range of users' age which can be related to this application. For this purpose font "Raavi" is chosen with a light shadow, which is well readable, in font size 11points. To increase the comprehension of small text which is placed in the Body band, there text with capital letters was chosen (e.g. page "Choose command sensitivity" Fig. 40). In other cases, for instance the capital letters used in "Label band" are increasing the importance of this page (e.g. page Topology Fig.31). According to [15] in professional design should not be more than 3 different kinds of fonts or a font family. The font has to be chosen with regard to the purpose of the application and whom it is aimed. The appearance and the "feel" of design are largely given by the font style.

7.2.5 Touch Target Size

To avoid problems with hitting the touch target and to make user-friendly application, buttons has to be designed in proper way. Two of the most important parameters are the size of the touch target and the size of the gap between them. According to [18] where the Nokia's developer resources suggest the minimum target size of the finger usable UI element 7x7mm with 1 mm gaps for index finger usage. For this reason there are buttons designed to this dimension or bigger. For buttons mainly placed in the Control band it is used size 9,6 x 7,5mm.

Even that the size of the buttons is big enough, the Amulet display is developed on a resistive technology which requires the usage of stylus. A user has to press down the surface of the display much harder comparing to the capacitive technology. The utilization of stylus or fingernail is in this case more effective and it causes required action with much higher probability than using the all size of finger top. For this reason I highly recommend to use stylus to keep this communication interface in a user-friendly area.

7.3 Problems with Size of the Memory

The graphical design is closely connected with size problems of the memory. The picture is as big as the amount of information contained within. It means if we have a lot of text, colours, transparency, edges in the picture, all of these units have certain influence on the file size. From another point of view, much more important for the size of the file, it is the technique of image compression. There exist a lot of different image formats but I will mention just those I dealt with during the progress. These are formats supported by the touch display and also commonly used.

These problems are important to mention from the reason of limited touch display SDRAM (64Mb) and Flash (32Mb) memory. Colour depth of the display is 24bit + 8 bit alpha channel which is representing the value of pixel transparency by number from 0 (transparent) to 1 (opaque) [16].

JPEG (Joint Photographic Expert Group) is one of the most common image formats. It supports 8-bit grey scale image and 24-bit colour image (8 bit each for R, G, B). Its advantage is relative small file size but disadvantage is losing information thanks to compression or recompression by another image edition. This raster image format is not suitable for further edition unlike with other formats which are lossless. JPEG format is best for final design thanks to the small file size [16].

PNG (Portable Network Graphic) supports 8-bit palette images (with optional transparency for all colours palette) and 24-bit true colour. Lossless PNG format is most suitable for pictures under edition and excels when the image has large, uniformly coloured areas [16].

GIF (Graphic Interchange Format) 8-bit colour image supports only 256 colours and single transparent colour. It is suitable for storing simple graphics, animation, shape, logos or cartoon style images. It uses the lossless compression too. It is more effective when large areas are single coloured. This format achieves the smallest file size due to the reliable compress algorithm [16].

Tab. 11 Comparison of the same picture

Format	File size [kB]
GIF	48
JPEG	56
PNG	133

During the project approach appeared problems with JPEG format. Display projected different colours mainly in parts of shadow. Therefore I consequently used PNG format.

On the top of it size problem of Amulet SDRAM appeared during the project progress due to the exhaustion of the memory capacity. That's why I chose GIF format

as a final solution, although the quality of design decreased. However, the change of the graphic quality is not considerably noticeable and thanks to this format change it reduces the file size of pictures by two-thirds comparing to PNG format.

In the GEMstudio exists another way how to reduce the utilisation of the SDRAM and Flash memory. This can be achieved by feature of object Background, "CustomButton" called "noSDRAM" where we can choose whether there will be allocated place in SDRAM for the object size. It is needed for the Href function of this object in case that it is used "appear" or "disappear" function. If this function is not used, we can "check" parameter "noSDRAM" and we will save more memory space. The default set of this parameter is "uncheck".

The exact usage of Flash and SDRAM memory can be seen in a file named "project_name.map", which is automatically created during the project compilation in Map folder. This file can be opened as a text file.

Tab. 12 Example of reducing page size via "noSdram" parameter for "Home" page

noSdram parameter	Usage of SDRAM [bit]	Usage of Flash[bit]
"unchecked"	8878152	378
"checked"	708972	370

8 TESTING OF COMMUNICATION

Amulet Resistive 4.3” LCD module allows the communication with an external processor via UART or USB. UART communication will be used for sending information regarding the setting which was set by a user in the touch interface. Settings configuration are address of created slave, command sensitivity, online/offline state, profile configuration and other information which will be further important for the Slave Simulator’s functionality. The core of the Slave Simulator represents a FPGA processor which sends back information to the touch screen e.g. amount of slave calls and responses. This communication has special protocol given by the producer of Amulet LCD module. We can choose from the CRC (Cyclic Redundancy Check) or ASCII protocol.

For this project the CRC protocol was chosen for its better data integrity. Also this protocol is set as default in the GEMstudio project and nowadays is encouraged. This protocol is similar in structure to the Modbus RTU. The external processor has to support the RS232 serial communication.

Tab. 13 Communication Format

Baud Rate [bps]	115200
Parity	None
Data Bits	8
Stop Bits	1

Amulet LCD module can behave as a master or a slave; it depends on the command which is sent or received. When Amulet is the master, the external processor is the slave and vice versa.

If host (external processor) is sending a master message to Amulet, the message will start with the Amulet ID. If Amulet is sending a master message to the host, the message will start with the Host ID. Responses start with the same ID as the original message. The default slave ID for Amulet is 1 and for the Host is 2 [17].

To set the Amulet as the master, the Href command of Amulet page needs to contain command which starts with “Amulet:UART”. The Amulet will send out the Href command at the interval based upon the “updateRate”, specified by particular object. The Amulet expects a response from the external processor within 200ms, by default.

The first byte of all master messages is the ID of the slave processor. The second byte is always the opcode of the function. See Appendix A for the full list of available opcodes. The remaining payload bytes are opcode dependent. The final two bytes are the LSByte and the MSByte of the CRC (16 bit) [17].

For example, if the GEMstudio file is compiled from view widget with the Href function of `Amulet:UART.byte(0x1A).value()`, which will send out the “Get byte variable” request “0x1A”, the transmitted message would consist of five bytes. The first byte is the Host ID (0x02), the second byte is the "Get byte variable" opcode (0x20), third byte is the byte variable number (0x1A), fourth byte is the LSByte of the 16-bit CRC of the first three bytes (0x48), and the fifth and final byte is the MSbyte of the 16-bit CRC (0x0B). So the five byte message looks like: 0x02 0x20 0xA1 0x48 0x0B [17].

Another example of “Set byte” can be seen in Fig.48. In this case the Host processor is master and Amulet is the slave. The sent message looks like: 0x01 0x30 0xC7 0x11 0x93 0xEB. The first byte is the Amulet ID (0x10), the second byte is the Set Byte opcode (0x30), the third byte is the byte address (0xC7), the fourth byte is the byte value which we want to set (0x11), the fifth byte is the LSByte of the 16-bit CRC of the first four bytes (0x93), and the final byte is the MSbyte of the 16-bit CRC (0xEB).

The Amulet response is 0x01 0x30 0x00 0x34. The first byte is the Amulet ID, the second byte is the "Set byte" opcode, the third byte is the LSByte of the 16-bit CRC of the first two bytes and the final byte is the MSbyte of the 16-bit CRC.

In the following figure 48 you can see the realisation of the example. To send information via RS232 to the Amulet, the HTerm program for serial interface was used. This program was configured to 115200 data rate, 8 bit data size, no parity and 1 stop bit. The connection of this program and Amulet was conducted utilizing the port COM6 via USB with integrated circuit FT232H Single channel USB to serial port. Amulet LCD module was connected to this chip via UART (Rx, Tx and ground signal). See Fig. 49 below.

As it is seen in the picture below in the window “Event method list”, the Href command “Amulet:internalRAM.byte(199).value()” was assigned to the widget “NumericField”, which shows the content of the byte 199. For the page, where this NumericField is placed, the Page function was created. It sends back a message ‘AAAA0000’ when the byte 199 is changed from 0 to another value. As it can be seen, this message was received by HTerm showed in the window “Received data” after the slave respectively Amulet response. The CRC code for this commands were calculated with an online CRC calculator with 0x8005 polynomial function.

The CRC code is responsible for reliability respectively data integrity of the communication. If the CRC code, which contains sent data, is not valid (decoded CRC code has to be the same as received data) then the received packet is ignored because the integrity of the data cannot be assured. The CRCs are popular because they are simple to implement in binary hardware, easy to analyse mathematically, and particularly good at detecting common errors caused by noise in transmission channels [17].

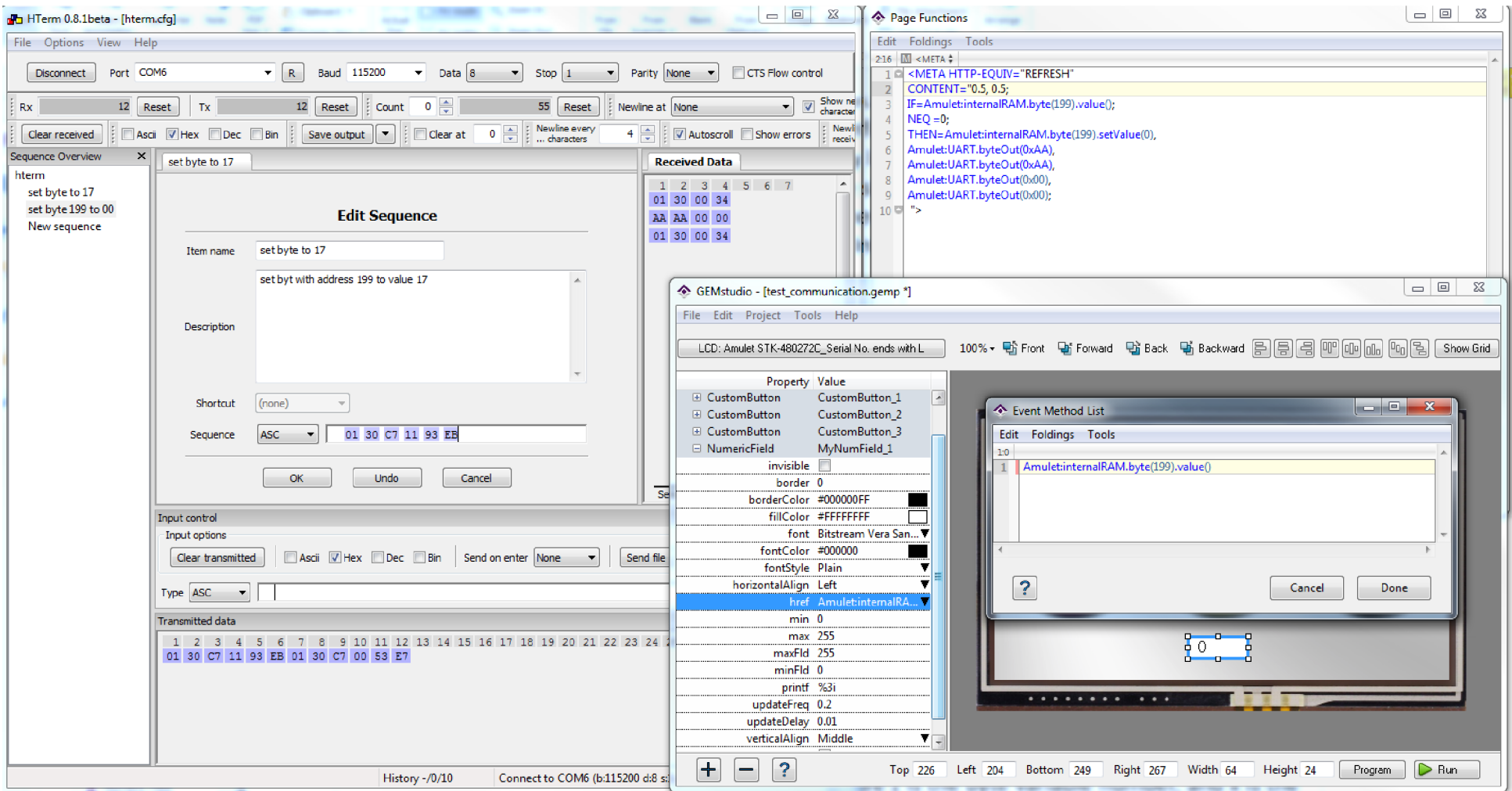


Fig. 48 Example of command Set byte

9 FPGA-AMULET COMMUNICATION PROTOCOL

9.1 Introduction

This part describes the aim of this protocol and providing features. In the following chapter the connection arrangement of the application during the design, testing and final connection of the application is described. The description of the used hardware and software so far not stated is in the next chapters. Basic information about FPGA architecture and VHDL programming language are listed.

The purpose of this FPGA-Amulet communication protocol is to provide and ensure communication between the Amulet LCD module and FASS's FPGA core. It means data transfer of user inputs from communication interface and Slave Simulator outputs which are important for the user. This communication has specific rules especially based on Amulet communication protocol already presented in the chapter 8 "Testing of Communication".

This task concerns the understanding of existing FASS communication interface written in VHDL and adjustment to the Amulet communication protocol in order to transfer needed data between Amulet and Slave Simulator.

9.2 Connection Arrangement

In Fig. 49 can be seen a type of hardware and software applied in design of the application and via which communication bus was connected.

In our case FASS represents board Altera DE1. To control and program this board it was conducted by utilizing software from company Altera called Quartus II. This board was connected with PC via USB.

The Amulet LCD module was programmed with the help of the GEMstudio and connected via USB to PC. The final communication between the Amulet and the FPGA is performed via RS232. IC FT232 provides monitoring and simulating communication with help of the HTerm software between FPGA and Amulet connected via USB to PC.

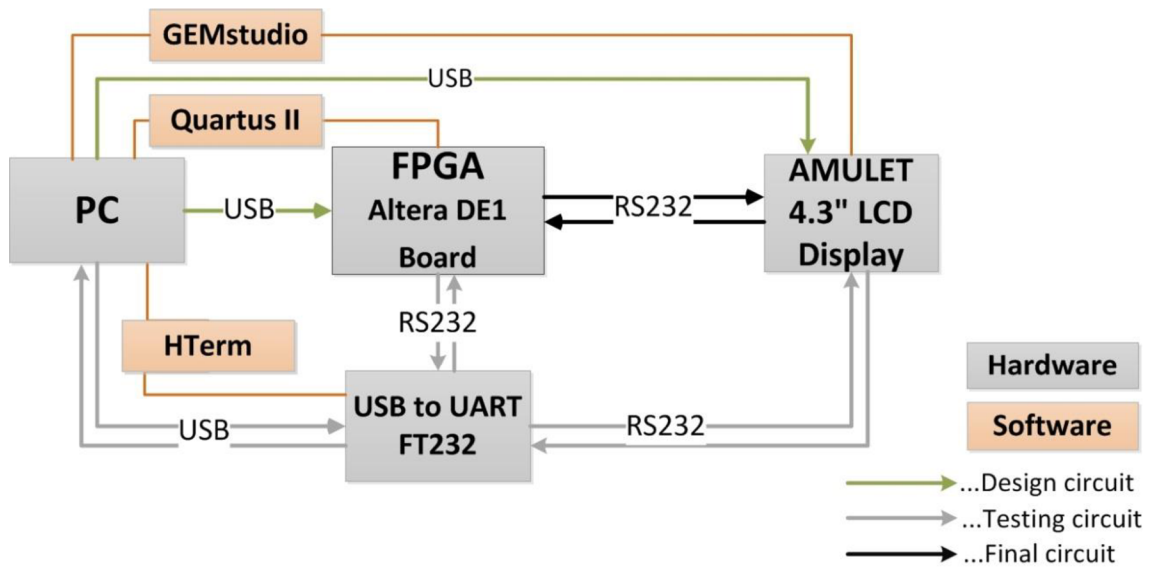


Fig. 49 Connection arrangement of the application

9.3 Hardware Description

9.3.1 Board Altera DE1

Board Altera DE1 was used for simulation of target hardware FASS. This board is equipped by FPGA Cyclone II and it is designed for university and college laboratory use. It is suitable for wide range of exercises with digital logic and computer organisation. [20]

Table 1 DE1 Board specification[20]:

FPGA	Cyclone II EP2C20F484C7 with EPCS4 4-Mbit serial configuration device
I/O Interface	Built – in USB Blaster for FPGA configuration Line In/Out, Microphone in (24-bit Audio) Video Out (VGA) Serial port (RS232) PS/2 mouse or keyboard port Expansion headers (two 40-pin headers)
Memory	8 MB SDRAM, 512 KB SRAM, 4 KB Flash SD memory card slot
Display	Four 7 - segment displays
Switches and LEDs	10 toggle switches 10 red LEDs 8 green LEDs Four debounced pushbutton switches
Clocks	50 MHz clock 27 MHz clock External SMA clock input

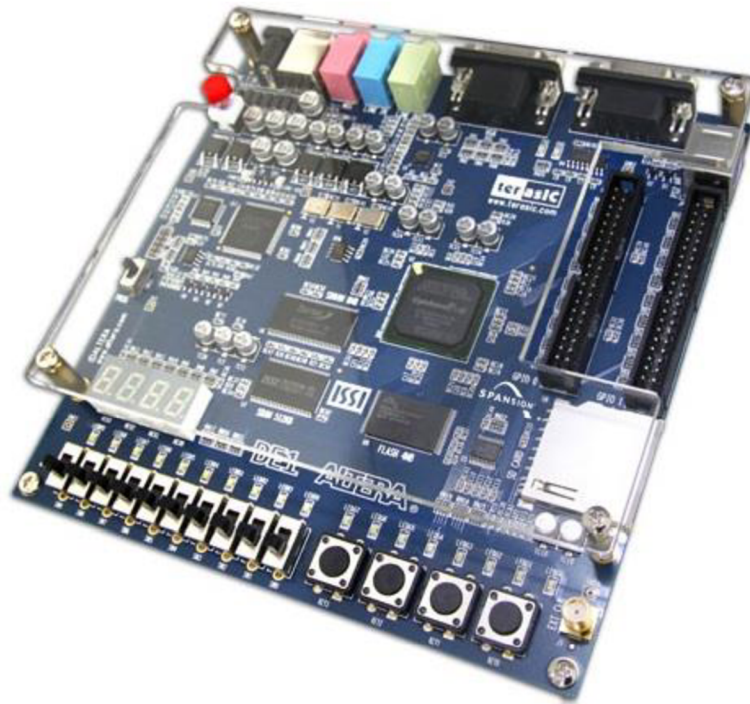


Fig. 50 DE1 Development and education board [20]

9.3.2 USB to UART FT232

This device was used to monitor and simulate communication between Slave Simulator and Amulet LCD module with the help of HTerm program. The communication between these two devices and findings of possible failures was performed. Follow the basic features of this device according to [21]:

- Single chip USB to asynchronous serial data transfer interface
- Integrated 1024 Bit EEPROM
- Integrated 3.3 level converter for USB I/O
- Integrated level converter on UART and CBUS for interfacing to 5V–1.8V Logic
- Fully integrated clock – no external crystal, oscillator or resonator required
- Fully integrated AVCC supply filtering – no separate AVCC pin and no external R-C filtering required

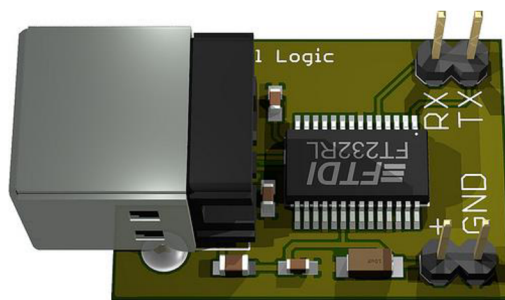


Fig. 51 USB to UART FT232 module [21]

9.4 FPGA of Final FASS

FPGA is a central control unit of the Slave Simulator. The FPGA Spartan 3 will be used from the company Xilinx type XC3S400-PQ208. The internal logic of this FPGA is based on a matrix structure of free configurable logic blocks (CLB). It is disposing by 400 000 gates altogether and over 8000 logic units which are working as Look-Up Tables [7].

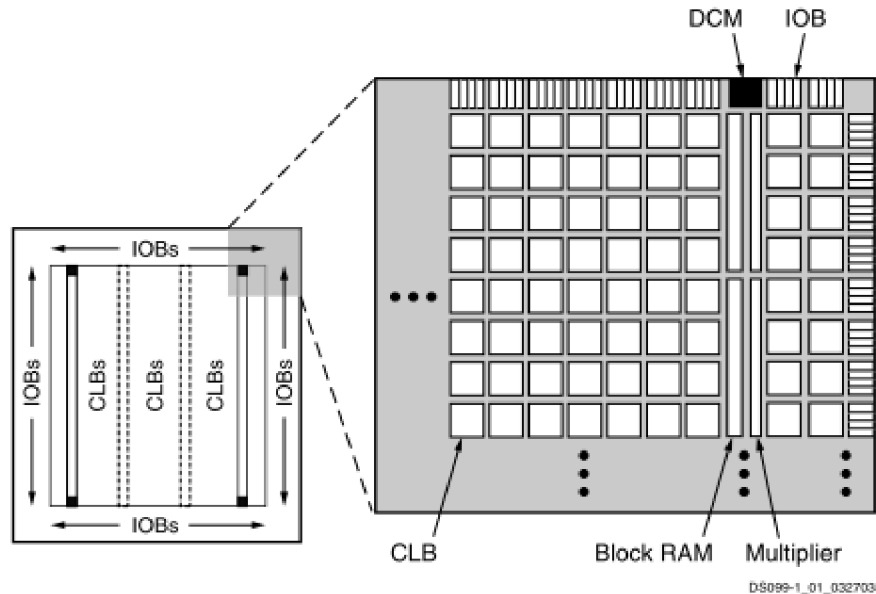


Fig. 52 Spartan-3 family FPGA block diagram [22]

On the top of that, it contains 56Kb distributed RAM, 16 pre-set multiplier unit, 4 DCM (Digital Clock Manager) and 246 free configurable inputs/outputs. There are predefined inputs/outputs as voltage supply or input for a system clock. This FPGA has 8 clock inputs which are connected to 280MHz crystal. This FPGA also allows own time generation via the internal DCM [7].

Three different voltages are used for voltage supply of programmable gate array. 1.2V for supply of intern logic (V_{CCINT}), 2,5V is used as a help voltage for function blocks (V_{CCAUX}) and 3,3V is used to set voltage level in in/output pins. The maximum current consumption is 180mA [7].

To program this FPGA the software Xilinx ISE will be used with the help of a programming language VHDL, which is described in detail in the next chapter. For reliable configuration of FPGA an USB convertor was developed, which allows to program FPGA directly from development tool ISE [7]. The program of this FPGA respectively Slave Simulator will be adjusted for Amulet LCD module and further function of a new FASS version 3.0 of the AS-i system.

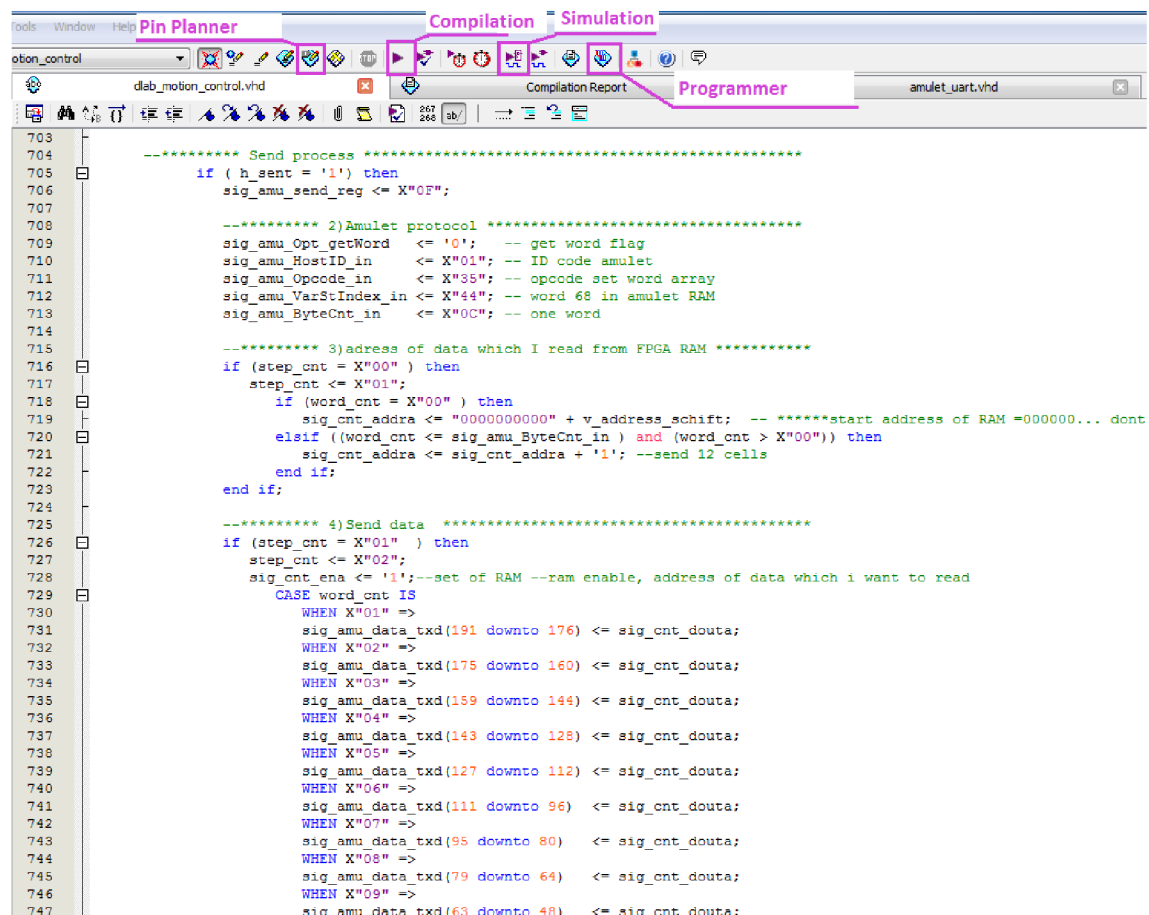
9.5 Software Description

9.5.1 Quartus II and ModelSim

Quartus II (see Fig.53) is a development tool produced by the company Altera. This software allows designing programs in VHDL or Verilog language and also via block diagrams for designs of FPGAs, CPLDs (Complex Programmable Logic Device) and SoCs (System on Chip). [30]

With help of this software the communication protocol between Amulet LCD Module and board DE1 (in future FASS) was designed in programming language VHDL. The next chapter introduce basic information about this programming language.

Quartus II with help of ModelSim tool allows simulation of the designed program. User can pick certain variables and show the change of the values in a predefined simulation time (see Fig.54). This instrument dramatically speeds the designed time up of the application and helps to understand behaviour of variables in complicated algorithms.



```
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
--***** Send process *****
if ( h_sent = '1') then
  sig_amu_send_reg <= X"0F";

--***** 2)Amulet protocol *****
sig_amu_Opt_getWord <= '0'; -- get word flag
sig_amu_HostID_in <= X"01"; -- ID code amulet
sig_amu_Opocode_in <= X"35"; -- opocode set word array
sig_amu_VarStIndex_in <= X"44"; -- word 68 in amulet RAM
sig_amu_ByteCnt_in <= X"0C"; -- one word

--***** 3)address of data which I read from FPGA RAM *****
if (step_cnt = X"00") then
  step_cnt <= X"01";
  if (word_cnt = X"00" ) then
    sig_cnt_addr <= "000000000" + v_address_schift; -- *****start address of RAM =000000... dont
  elsif (word_cnt <= sig_amu_ByteCnt_in ) and (word_cnt > X"00") then
    sig_cnt_addr <= sig_cnt_addr + '1'; --send 12 cells
  end if;
end if;

--***** 4)Send data *****
if (step_cnt = X"01" ) then
  step_cnt <= X"02";
  sig_cnt_ena <= '1';--set of RAM --ram enable, address of data which i want to read
  CASE word_cnt IS
    WHEN X"01" =>
      sig_amu_data_txd(191 downto 176) <= sig_cnt_douta;
    WHEN X"02" =>
      sig_amu_data_txd(175 downto 160) <= sig_cnt_douta;
    WHEN X"03" =>
      sig_amu_data_txd(159 downto 144) <= sig_cnt_douta;
    WHEN X"04" =>
      sig_amu_data_txd(143 downto 128) <= sig_cnt_douta;
    WHEN X"05" =>
      sig_amu_data_txd(127 downto 112) <= sig_cnt_douta;
    WHEN X"06" =>
      sig_amu_data_txd(111 downto 96) <= sig_cnt_douta;
    WHEN X"07" =>
      sig_amu_data_txd(95 downto 80) <= sig_cnt_douta;
    WHEN X"08" =>
      sig_amu_data_txd(79 downto 64) <= sig_cnt_douta;
    WHEN X"09" =>
      sig_amu_data_txd(63 downto 48) <= sig_cnt_douta;
```

Fig. 53 Quartus II development environment

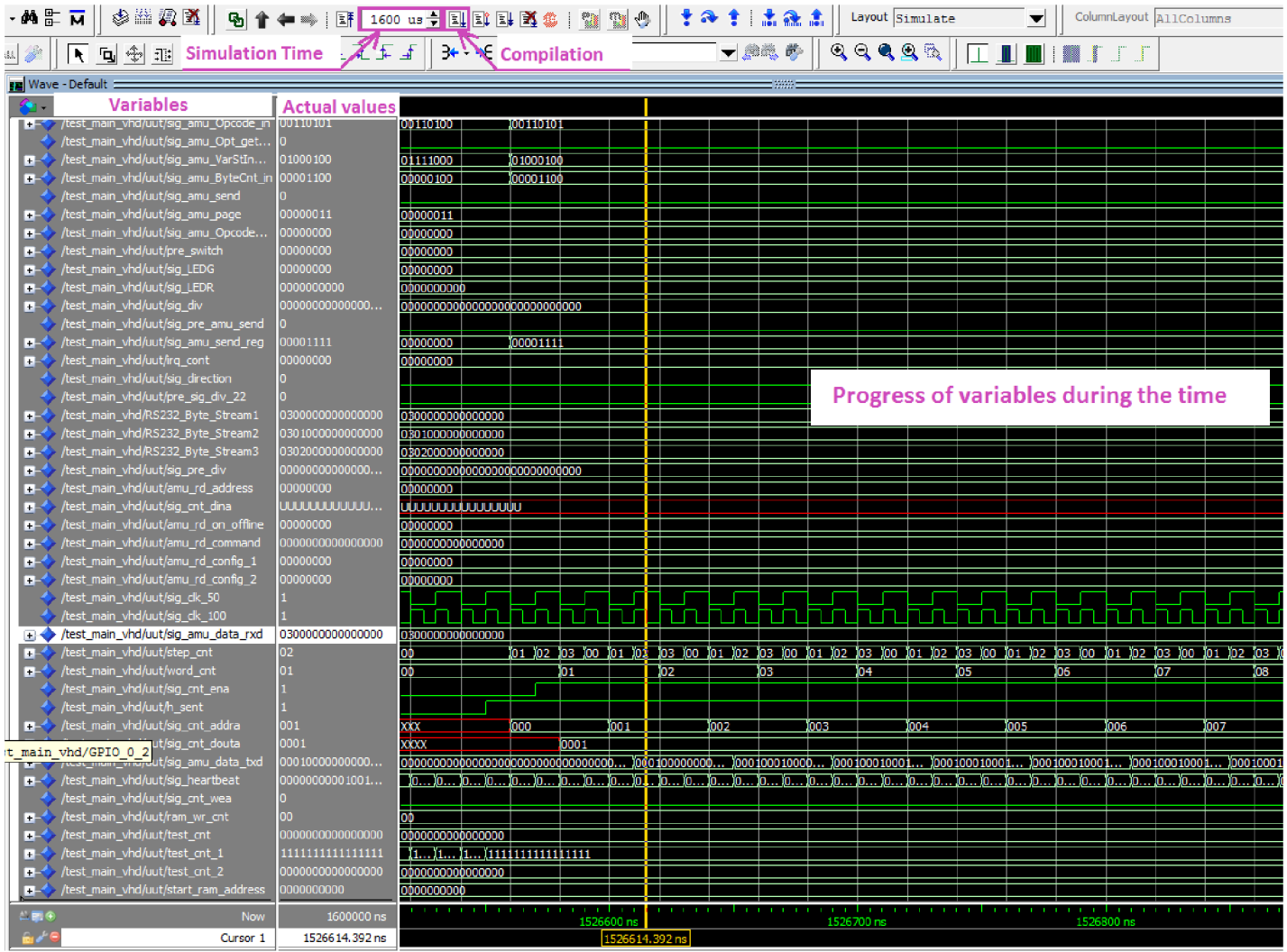


Fig. 54 ModelSim simulation tool

9.6 VHDL

9.6.1 Introduction

The programming language VHDL (VHSI Hardware Description Language) is a hardware description language used for description and simulation of extended numerical systems such as FPGA and integrated circuits. It is used mainly in electronic design automation. This programming language allows a wide range of expression and the described numerical system is independent on aimed technology. It means that this language together with programming language Verilog belongs to the most worldwide used programming languages working with numerical systems [8].

VHDL was originally developed at the behest of the U.S. Department of Defence under the name VHSIC (Very High Speed Integrated Circuits), which gave the name to this language (VHSIC HDL). In 1987 this language was standardized by the IEEE organisation (Institute of Electrical and Electronics Engineers). In 1993 this organisation published second standard of VHDL, which became commonly used. The name of this standard is IEEE Std 1076-1993 or VHDL-93. A new VHDL standard is published every five years [8].

9.6.2 Program Structure

The bases of the program structure will be explained according to [9][10] in the following example. This example is not complex and it contains only basics of VHDL.

Program realising AND and OR logic wiring (see Fig. 55)

```
library ieee;  
use ieee.std_logic_1164.all;  
  
entity gates is  
  port( a,b: in std_logic;  
        q,r: out std_logic  
        );  
end;  
  
architecture implement of gates is  
  begin  
    q <= a and b;  
    r <= a or b;  
end;
```

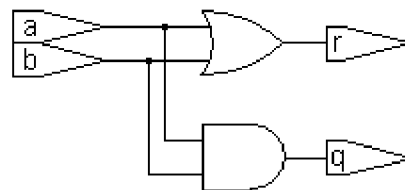


Fig. 55 Wiring of AND a OR logic [9]

It is evident, that the base of the program consists of three parts; key words are marked by bold.

The first part starts with a key word **library**, by this command the library *ieee* is attached. => Libraries make accessible entities and architectures analysed previously.

Second line with **use...all** adds all from this library which is defined in the library package under the name *std_logic_1164*. This package is used for the definition of *std_logic*, which is used as the **architecture** part and **and, or** logic operation.=> Packages contain the declaration and definition of the object, which can be used in projects.

The key word **entity** defines the (input/output) interface of an object with the surrounding. With the object it can be intended a logic gate, a circuit or a whole system. Behind the **entity there** is placed a so-called Entity ID respectively identification name. Inside of the entity part behind a keyword **port**, there is situated a free definition of inputs and outputs. It says, that signals *a* and *b* are inputs (marked by **in**), *q* and *r* are outputs (marked by **out**) of *std_logic*. Further keywords **inout** (bi-directional mode) and **buffer** (same as out mode but possible read it inside of entity) can appear here. => The entity names inputs and outputs; it defines their type and direction of transfer.

The keyword **architecture** is followed *Arch ID* of *Entity ID*. As a next step, the definition and declaration is placed. In the example this part is empty and the keyword **begin** follows.

The next parallel commands define the behaviour between ports of entity. This notation is behavioural but it can as well be in RTL form (Register Transfer Logic). It is described in process form. There can appear in a structural form, too - schema in a text form. => The architecture defines the content of entity (behaviour between ports). In every entity at least one architecture must be defined. Each architecture character of each entity must have a specific name.

Another basic object in VHDL

Signal - it is after an implementation realised by a wire. As a part of entity, signals are declared as gates (**ports**), as a part of architecture as an inner signal (**signal**). They are accessible only inside of the given entity. An assignment or a matching is performed by a command “<=”.

Variable - is used as an additional variable for realisation of iterative cycles (for, while). It is possible to define it for the usage of architecture (as shared) or in a single process (non-shared). To assign a value to the variable, a command “:=” is performed. It is allowed to perform this command just as a declaration or in **process**. **Variables** in contrast with **Signals**, get assigned value immediately. Signal gets assigned value after one clock cycle.

Constant - it is used for the storage of an unchangeable value, which is possible to assign to **signal** or **variable**. It can be defined in the declaration part of the architecture or the process. The assignment is performed by command “:=”.

10 IMPLEMENTATION OF VHDL

10.1 Program Structure

The VHDL program of FASS communication interface consists of 4 parts. The first two mentioned units are the most important of the program and are described in detail by block schematic.

“**dlab_motion_control**” is the “Main” program of the communication interface. Here are the main tasks programmed and performed which the communication interface should fulfil. The program is receiving, saving and sending data from/to Amulet. In this part a simulation of command counter is performed which requires a data change in RAM memory. This part will be described in detail the next chapter.

“**Amulet_uart**” is designed to save incoming data from Amulet and to ensure the Amulet communication protocol for outgoing data. It consists of right number of bytes in right order starting with the ID code, opcode, variable address, data and ending with the CRC code. The CRC code is calculated in this part of the program. For the calculation the length of the sending data is very important which changes by different Amulet commands.

“**Cnt_ram**” perform the access to RAM memory. In this part of the program RAM memory is structured to cells of appropriate size and predefined.

“**PLL1**” this block of program is called “Phase Locked Loop” and the task of this program is to ensure the system clock period stability and in addition the system clock division.

10.2 Main

The first process (Fig. 56) check an incoming message and decide for which process (respectively page) this message is destined to. The message from Amulet sent by command “ByteOut” always contains eight bytes. The first byte brings the value of “Page code” changing from 1 to 3. This code is read after checking the number of incoming bytes. Then a code of the appropriate page is performed.

If the incoming message contains seven bytes, then this message is considered as the answer of the “Get word” requests of “Page 1” process. The second byte (Opcode) which this message identifies is read.

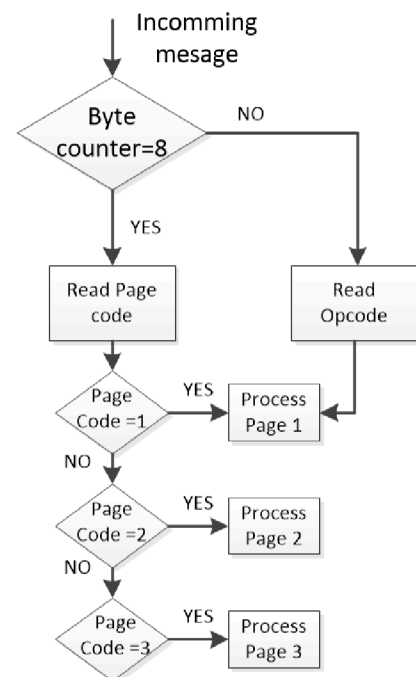


Fig. 56 Block diagram of Main

10.2.1 Process of Page 1

The main goal of this process is to read and write the needed information about created slave from/to the memory of the Slave Simulator.

A message with Page code 01 is sent by the page “Confirmation slave added” (P.2.2.7) from Amulet LCD module. This page appears after successfully saving newly created slave in communication interface. It brings the following information of slave settings in presented order and format.

- Address (1 byte)
- Online/offline settings (1 byte)
- Profile configuration (2 byte)

Profile configuration represents settings of all ID codes, Data, Status and Parameter. After laps of the pre-set timer the request “Get word variable” (Opcode 21) is sent to Amulet. Answer of Amulet is 7 bit-long and second byte contains Opcode 21, which is checked and the received word is saved. This word contains the setting of the command counter (P.2.2.6).

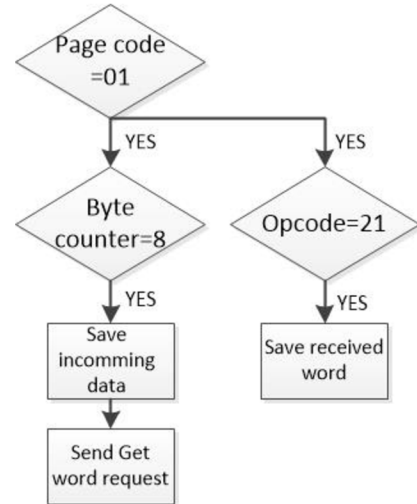


Fig. 57 Block diagram of process Page 1

10.2.2 Process of Page 2

A message with Page code 02 is sent by the “Delete slave confirmation” page (P.2.1.2) of Amulet. This message only contains the address of the slave. This process fulfils basic tasks of deleting settings of an appropriated slave in order to clear the allocated memory space. Following settings of slave are deleted:

- Online/offline status
- Profile configuration
- Settings of command counter

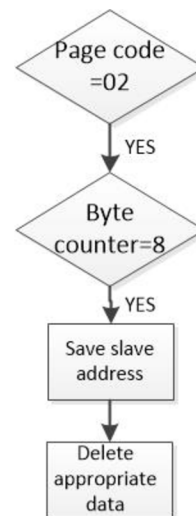


Fig. 58 Block diagram of process Page 2

10.2.3 Process of Page 3

The main purpose of this process is to send data of command counter from the RAM memory of the Slave Simulator to the Amulet module.

A message with Page code 03 is sent by Amulet page "Slave info" (P.2.1) where the command counter statistic of an appropriated slave is projected. The amulet page sends this message periodically with 200ms delay in order to have actual information of the

command counter. This message only contains the address of the slave from which the data shift in RAM memory is calculated to send the appropriate data of the command counter.

As a first step, the check of Byte counter of the incoming message is done. The address of a slave is saved and the "Time counter 1" of "Send process" is set to zero.

The Time counter 1 starts and after lapse of certain time the Send process runs. The Send process consists of 5 steps as you can see in Fig. 60.

1. The Amulet protocol is set ("Send word array" opcode, address of Amulet memory, number of sent words)
2. The address shift of RAM is counted
3. RAM is enable and appropriated data are sent
4. RAM is disabled and Send process flag is set to null
5. The Step counter is set to null

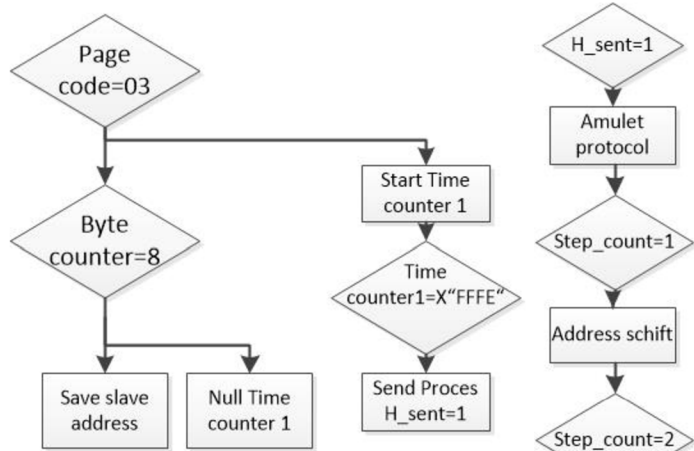


Fig. 59 Block diagram of process Page 3

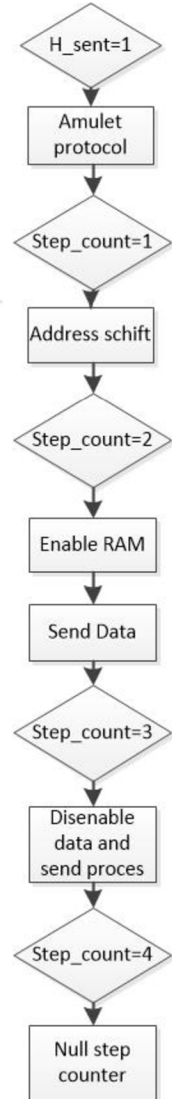


Fig. 60 Block diagram of Send process

10.2.4 Simulation of Command Counter

This part of the program is designed to simulate the command counter. Its task is to count command calls between an appropriate slave and the master. In the simulation it means that twelve RAM cells (12 different commands) are changing its value in different speed for each slave. Some of the commands are called very often, some of them not. From this reason twelve 16 bit-long counters incremented by 6 different timers were designed.

You can see the process of data writing into RAM in Fig.61. Every 100ms the Time counter is nulled and starts the writing process. Then enable RAM, enable writing, writing data, RAM disable and write disable is performed in 14 steps with 5ms timer. For writing it into RAM the channel B of the memory is used. For reading from RAM the data process of page 03 channel A is used. The Dual Port RAM (A and B channel) allows writing and reading from RAM in different speeds. To use one channel for reading and writing is not allowed.

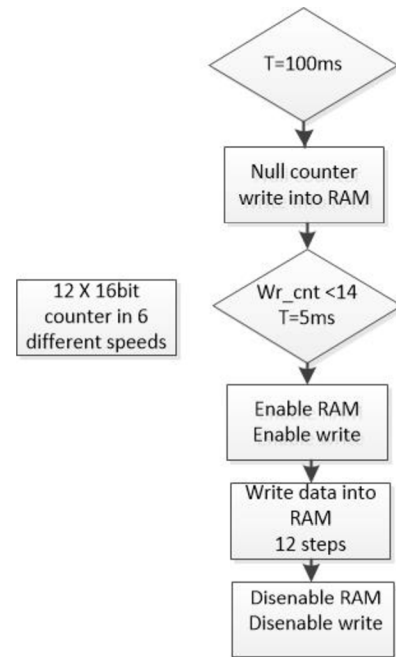


Fig. 61 Simulation of command counter

10.3 Amulet UART

In this part of the program **Amulet_uart** the performed tasks are needed for receiving an incoming message from Amulet and sending a message from FPGA to Amulet in the right format.

If the incoming message request occurs, a saving cycle of 8 steps runs. In every step an incoming byte is saved into a store register. There is not bigger size than 8 byte of the incoming message expected.

An outgoing message process is more complicated. In this case the Amulet communication protocol has to be maintained. It means to maintain the right order of bytes and their right number. Three different messages have to be distinguished. You can find the exact description of Amulet communication protocol in Appendix A.

- “Get word variable” request, 5 byte long message
- “Send word variable array”, changeable size from 8 bytes up to 28
- “Send byte variable array”, changeable size from 7 bytes up to 28

In this part there is also calculated size of outgoing message which is needed for CRC code determination. CRC code appears in every last two bytes of the outgoing message to ensure the communication reliability and security respectively the data integrity.

11 TEST OF USER USABILITY

11.1 Introduction

The task of this thesis is to achieve a user-friendly design of the application. Due to this reason a literature research was done with key phrases: intuitive GUI design, interactive design, user-friendly design, colours in GUI design.

The result of this literature research didn't bring direct answers. Most often there is an overview of guidelines or manuals "how to make a user-friendly design" available but the full version is only for sale. The design of a user-friendly application usually belongs to the know-how of the company.

Another possibility, according to [28] which says, that the effective user interface design is only possible through testing and studying, considering the many user interface guidelines and the different usage of the application. Hence the test was performed in order to proof intuitiveness, simplicity, logical arrangement and the importance of colours of the application for the user.

The test was performed with 10 interviewees. All of the interviewees had technical background and basic or advanced knowledge of AS-i or FASS, which corresponded to the target group. Responders were given an introduction of the application. Every asked person had to complete four different tasks with the consideration of all the application features. An operating time and a number of inputs were measured. After task performing, seven questions were asked concerning the usability, logical arrangement, colour diversity of the application etc.

The test was divided into two parts. The first five respondents were interviewed about using application with "Layout 1" and then the design was adjusted according to occurred problems and users' suggestions. Then the test was performed with the next five interviewees using "Layout 2". The notes of the responders were also considered in the last application adjustment ("Layout 3"). The exact form of the User-friendly test can be seen in Appendix T.

11.2 Test Performance and Design Changes

As it was already stated, the interviewees were given four tasks. The first three concerned the setting of a new slave with predefined parameters:

- Address
- Name
- Online/offline status
- Slave profile
- Settings of command counter statistic

The second and the third task were completed by deleting the already created slave. In case of the second task, the slave with the address 60 is created and placed in the page “Topology 1” (Fig. 32, P.2.0), which doesn’t appear directly after slave creation, but the user have to use the button “Next page” to see the topology in address range 32-62. This task appeared to be the most difficult part of the test.

The fourth task was about the additional function of the application. The user should find the name of the producer and the actual time and date. As mentioned above, the operating time was stopped and the number of inputs was counted for every task.

On the end of the test, 7 questions were given to the interviewer. The first six questions were evaluated by points from 0 up to 10, 0 for worst and 10 for best correspondence with a question.

Asked questions:

1. Is the application logically arranged?
2. Have you been satisfied with the colour diversity?
3. How much do you think the colours help the orientation in the application?
4. Do icons represent its meaning enough?
5. Is the text good readable?
6. Is it easy to use this application?
7. What would you change? What do you not like?

In Table 14 you can see the collected data of the test. We can split the data into two main parts. The data of Layout 1, is prior the design adjustment and the data of Layout 2, after the design adjustment. From this data and from diagram 1 and 2 we can see the resulting contribution of the adjustment. Reference data were needed because every task is slightly different and requires a different time and number of inputs. As the referential data of the test, I used my own data collection, because I have the best knowledge of the application.

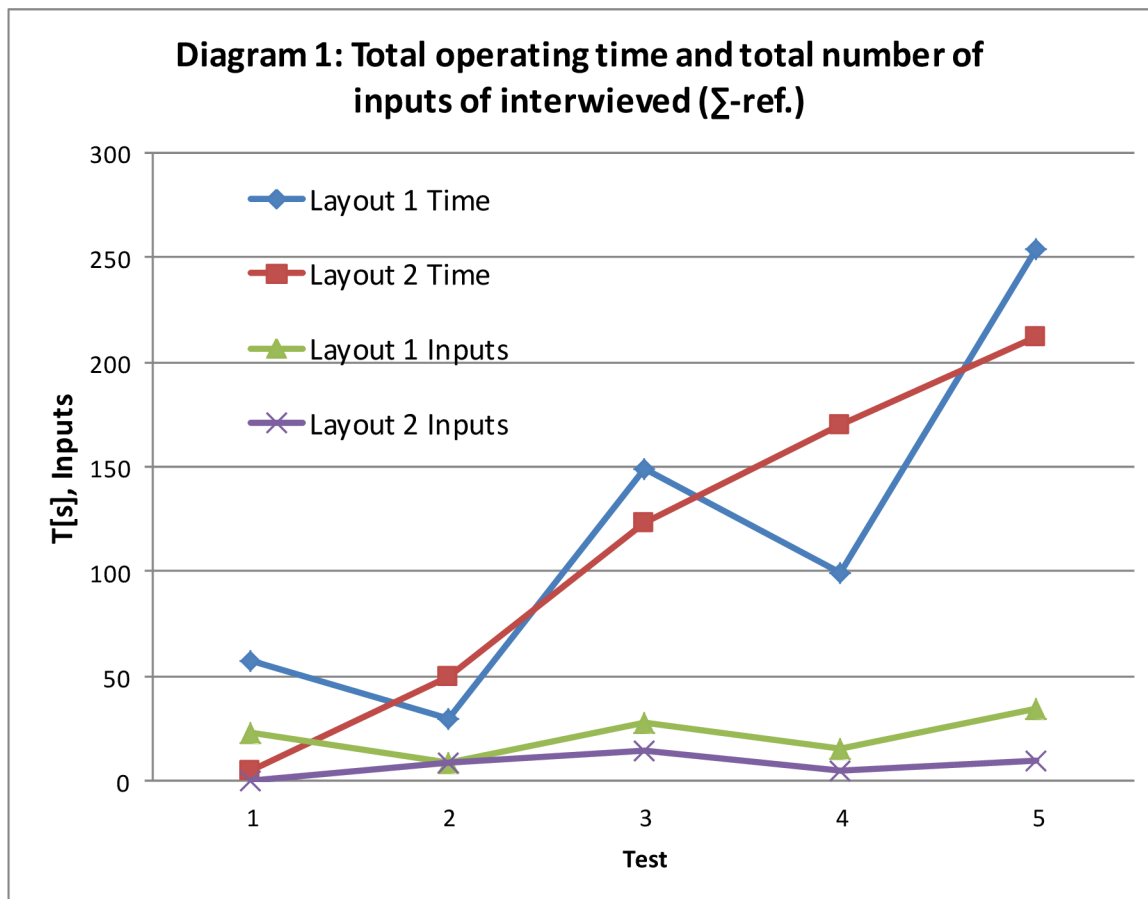
Tab. 14 User-friendly test collected data

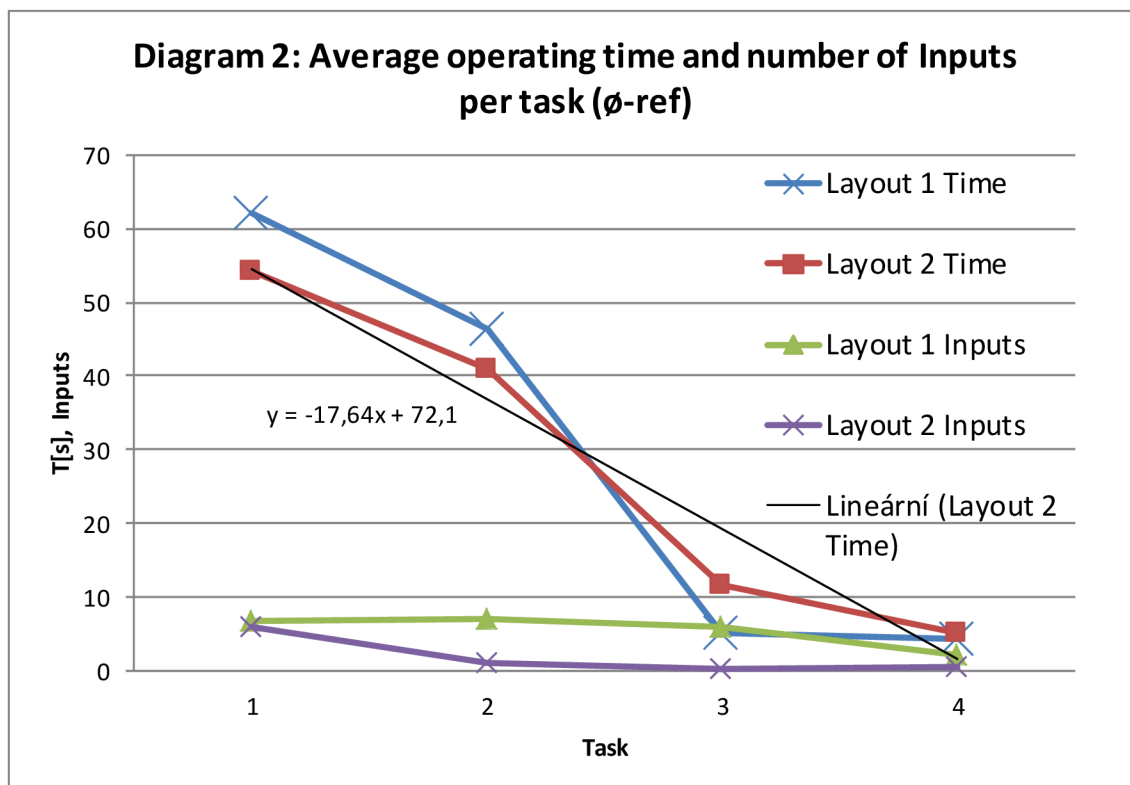
Layout	1							2							
Test	1	2	3	4	5	∅	∅-ref.	6	7	8	9	10	∅	∅-ref.	Ref.
Time[s]															
Task 1	108	81	176	89	136	118,0	62,0	61	84	110	117	179	110,2	54,2	56
Task 2	73	73	84	118	179	105,4	46,4	70	81	113	121	115	100,0	41,0	59
Task 3	61	64	70	69	71	67,0	5,0	50	60	83	103	72	73,6	11,6	62
Task 4	7	4	11	15	60	19,4	4,4	16	17	9	21	38	20,2	5,2	15
∑-ref.	57	30	149	99	254		117,8	5	50	123	170	212		112,0	192
Number of inputs [times]															
Task 1	38	30	41	31	34	34,8	6,8	28	30	40	32	40	34,0	6,0	28
Task 2	*35	*35	37	48	47	44,0	7,0	37	40	*31	38	37	38,0	1,0	37
Task 3	39	33	48	34	40	38,8	5,8	33	35	35	33	30	33,2	0,2	33
Task 4	4	4	4	4	15	6,2	2,2	4	6	4	4	5	4,6	0,6	4
∑- ref.	23	9	28	15	34		21,8	0	9	14	5	10		7,8	102
Question [points from 0 to 10]															
1	4	9	10	8	10	8,2		9	9	8	8	10	8,8		
2	10	9	8	8	10	9,0		10	10	8	9	10	9,4		
3	10	10	10	0	10	8,0		10	7	5	10	8	8,0		
4	8	10	8	9	9	8,8		8	10	7	7	10	8,4		
5	10	10	10	10	9	9,8		10	10	9	10	10	9,8		
6	10	10	10	8	10	9,6		10	7	9	8	10	8,8		
Age															
Age	31	28	26	36	62	36,6		26	25	25	32	58	33,2		
*...Task was not correctly completed, average substitution															
∅...average															
∑...sum															
Ref...reference															

The referential data were subtracted and the results are projected in diagram 1 and 2. As we can see in Table 14, the contribution of the design adjustment is not so significant regarding the operating time but much more significant in case of the inputs number. The sum of the average performed invalid inputs per task of Layout 2 achieved the value of 7,8 compared with Layout 1 where the value was 21,8. It means that user performing tasks with Layout 2 made significantly less needless inputs. This dependence is also described in diagram 1 and diagram 2, where the curve of invalid inputs made in Layout 2 is in all cases lower than in diagram1, except of one where it is equal.

Diagram 1 also describes considerable time dependence per task of the user age. The fifth answerer was in both cases the oldest one of the group and achieved the longest operating time. According to [28] this dependence is also strongly influenced by user's experiences with similar application, which was not tested in this study.

In diagram 2, we can notice that the operating time average per task decreases with every finished task. It shows the user's capability of understanding and orientation in the application.





It is important to mention the form of data collection and the influence of the result. The time was measured by stopwatch and the number of inputs was counted by observing the respondent. During the operating there occurred false inputs not performing any device action, however, there were counted. As it was tested, error of measurement is approximately 10%. Noticeable influence on the operating time has also the comprehension of the task submission. The task submission was sometimes needed to be explained again during the task performance.

For a more accurate study, more interviewees would be needed, more tasks and accurate measurement would be done. For example with help of testing application which can evaluate exactly the number of correct inputs and the exact operating time per task.

The average value of the question evaluation is resulting in very positive numbers, but it didn't bring many reasonable comparisons between first and second layout, because this task is very subjective and relative. As we can notice in Table 14, in case of the Layout 2, better results were achieved by the first and the second question. The best results in both cases of layouts are the question concerning readability of text and colour diversity of the application. In case of the third question, 80% of responders find a colour very important for the orientation in the application. All of the interviewees found the application easy to use, intuitive and fast to understand.

In my opinion, the seventh question had the biggest contribution, which opens the discussion of user's not in favour parts of the application and necessary adjustment related to it. Following ideas were noted and the application adjustment is displayed.

- The icon of the Topology page at the Home page should be bigger than the others in order to increase its importance. The meaning of the icon should be designed better.



Fig. 62 Layout 1 of Home page



Fig. 63 Layout 3 of Home page

Problem: After the creation of the slave with an address bigger than 31 the page Topology appears. There the user can not directly see the created slave and the “Next page” button is needed to press to get at page Topology 1.

Suggestions:

- The Next page button should blink, should be bigger or better placed
- The slaves should be arranged in the same order as they are created
- Orientation axis in which part of topology the user is located should be implemented
- The AS-i cable should be dash at the end and a symbol should be 1/2, 2/2 added

Result: The small Next page button was added on the dash end of the AS-i cable in page Topology 1. This arrow is blinking. The size of the button cannot be bigger, because every Slave button has an exact placement at the page (See Fig. 31).

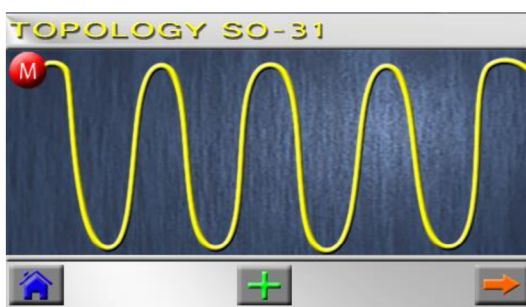


Fig. 65 Page Topology Layout 1

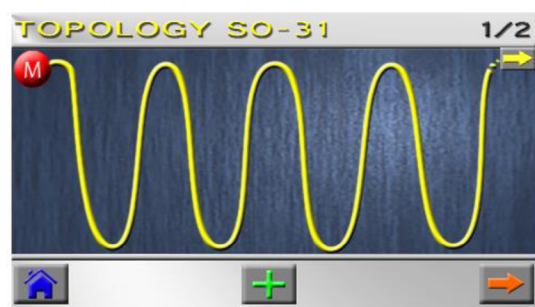


Fig. 64 Page Topology Layout 3

- Online/offline button in page “Set slave online/offline” (P.2.2.3) should be coloured

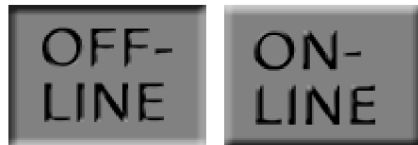


Fig. 67 Online/offline button Layout 1

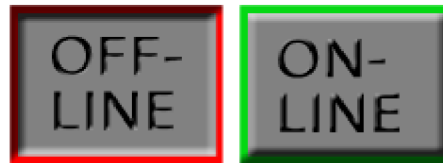


Fig. 66 Online/offline button Layout 2

- Delete button in page “Slave info” (P.2.1) should better represent its meaning

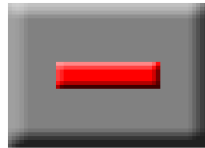


Fig. 68 Delete button Layout 1



Fig. 69 Delete button Layout 3

11.3 Conclusion

As the biggest contribution of the User-friendly test, we can consider a growth of the user-friendliness of the application. The design was adjusted according to the notes of interviewees and its positive result is showed in the measured data. The number of needless inputs decreased approximately by 60% (tested with Layout 2). The application itself is intuitive, understandable and the user learns by doing. It is ensuing from measured operating time, which decreases with angular coefficient -17,6 what can be seen in Diagram 2 projected for Layout 2. It shows the speed of the user’s orientation in the application. Also the questions concerning logical arrangement of the application and easy usability were well evaluated by 8,8 of 10 for Layout 2. 80% of responders found colours very important for the graphic design of an application. The colour diversity of the application was evaluated by 9,4 of 10 for Layout 2.

From the study ensues that the text is even well readable for elderly users. After the last adjustment, all of the icons used in the application are well understandable and represent its meaning even with omitted description.

From the positive evaluation of all questions, we can consider the application as user-friendly.

12 CONCLUSION

In the beginning of this thesis there was presented the basic knowledge needed for the understanding of this project. The reader gets an overview of the AS-Interface, its topology and way of function. Furthermore there is mentioned fundamental information about the FASS, details of used LCD module and its possible capacitive substitution.

Subsequently, in the second part there are explained particular solutions for the touch communication interface using the GEMstudio. The design of the communication protocol, the concept of control, the programming and the graphic design of Amulet STK-480272C communication interface are successfully designed. The development environment, the way of progress and the achieved solutions are sketched out. The communication test of Amulet LCD module and extern processor was successfully realized. A future simplification of the program via the GEMscript is pointed out. The complete program design with comments can be founded in Appendix P1.

There appeared one fundamental problem during the progress of the project. The memory size of Amulet LCD module was exhausted. This memory was not big enough for the graphical design. The method of minimizing the utilisation of the memory was successfully founded and described.

The next part of this project deals with an integration of the communication interface and the motherboard of FASS with help of the Altera DE1 board. The basic understanding of this hardware, FPGA and VHDL is sketched out. The design of communication protocol between FPGA and Amulet LCD module is successfully created and described via block schematic. Usage under the working condition was successfully tested. See a video demonstration in Appendix V.

Another big contribution of this thesis can be seen in creation of user-friendly communication interface and a method describing achievement of this point. As the User-friendly test demonstrated, a user can fast understand the application. After few first performances the user orients intuitively and well within the application. As it is presumed, a final user of this application will work with this device in the long term basis. Nevertheless, the simplicity is very important for the user understanding. As the test proves, to reach high user friendliness, the colour diversity of the application helps as well. It was taken into the consideration during the design. According to the test, users find the application very easy to use.

The result of this thesis is a study of the stated problematic which faces problems and describes possible solution. The proceeding of this project will continue in the future. The developed product can be practically used as a testing device for the final FTZ AS-i Slave Simulator of new v3.0 generation devices working with the AS-i.

Further improvement of this communication interface can be seen in the implementation of condition to avoid the user mistakes. Specifically, the condition avoiding the assignment of already existing address. Realization of this and further condition will be significantly easy with help of the GEMscript included in the new version of the GEMstudio. It was published during the elaboration of the project.

Furthermore a usage of a bigger capacitive display would increase the user friendliness. Finger-touch, scrolling and sizing is much more intuitive and comfortable. Bigger size of the display would be more readable and it would allow showing more information in one time without scrolling or changing the screen.

13 LIST OF FIGURES

Fig. 1 Communication pyramid of industrial automation [29]	10
Fig. 2 Linear topology	12
Fig. 3 Free topology	12
Fig. 4 Star topology	13
Fig. 5 Ring topology	13
Fig. 6 Transmission medium	15
Fig. 7 Penetration technique of AS-i	15
Fig. 8 Physics layer [1]	16
Fig. 9 Principle of communication [1]	16
Fig. 10 Structure of AS-i message [1]	17
Fig. 11 Network cycle in standard addressing mode [1]	18
Fig. 12 Network cycle in extended addressing mode [1]	18
Fig. 13 Block diagram of Slave Simulator HW arrangement [25]	24
Fig. 14 Amulet Resistive 4.3“ GEM starter-kit [31]	25
Fig. 15 Amulet Capacitive 4.3 GEMmodule™ [27]	26
Fig. 16 Sketch of the project block diagram	29
Fig. 17 Final block diagram of the project	29
Fig. 18 GEMstudion programming environment	32
Fig. 20 Time and date page	35
Fig. 19 Set time and date page	37
Fig. 21 Block diagram of Add new slave menu	39
Fig. 22 Block schematic of process	40
Fig. 23 Block schematic of process in Delete page	40
Fig. 24 Page division	41
Fig. 25 Home button Up and Down-image	41
Fig. 26 Microsoft Visio work environment	42
Fig. 27 Information (P.1)	44
Fig. 28 Home menu (P.0)	44
Fig. 29 Time and date (P.3)	44
Fig. 30 Set time and date (P.3.1)	44
Fig. 31 Topology (P.2)	44
Fig. 32 Topology 1 (P.2.0)	44
Fig. 33 Add slave address (P.2.2.1)	45
Fig. 34 Add slave name (P.2.2.2)	45
Fig. 35 Help slave configuration (P.2.2.4.1)	45
Fig. 36 Set slave profile (P.2.2.5)	45
Fig. 37 Set slave profile (P.2.2.4)	45
Fig. 38 Set slave online/offline (P.2.2.3)	45
Fig. 39 Confirmation slave added (P.2.2.7)	46
Fig. 40 Choose command sensitive (P.2.2.6)	46
Fig. 41 Delete slave (P.2.1.1)	46
Fig. 42 Slave info (P.2.1)	46
Fig. 43 Confirmation slave deleted (P.2.1.2)	46
Fig. 44 Topology button - size effect	47

Fig. 45 Information button - deformation effect	47
Fig. 46 Home button - shadow effect	47
Fig. 47 Photoshop work environment	48
Fig. 48 Example of command Set byte	53
Fig. 49 Connection arrangement of the application	55
Fig. 50 DE1 Development and education board [20]	56
Fig. 51 USB to UART FT232 module [21]	56
Fig. 52 Spartan-3 family FPGA block diagram [22]	57
Fig. 53 Quartus II development environment	58
Fig. 54 ModelSim simulation tool	59
Fig. 55 Wiring of AND a OR logic [9]	60
Fig. 56 Block diagram of Main	62
Fig. 57 Block diagram of process Page 1	63
Fig. 58 Block diagram of process Page 2	63
Fig. 59 Block diagram of process Page 3	64
Fig. 60 Block diagram of Send process	64
Fig. 61 Simulation of Command counter	65
Fig. 62 Layout 1 of Home page	71
Fig. 63 Layout 3 of Home page	71
Fig. 64 Page Topology Layout 3	71
Fig. 65 Page Topology Layout 1	71
Fig. 66 Online/offline button Layout 2	72
Fig. 67 Online/offline button Layout 1	72
Fig. 68 Delete button Layout 1	72
Fig. 69 Delete button Layout 3	72

14 LIST OF TABLES

Tab. 1 Basic specification of AS-Interface [4]	11
Tab. 2 ISO/OSI model of AS-Interface [3]	14
Tab. 3 Master calls and related slave responses [19]	19
Tab. 4 Table of standard slave profiles [23]	21
Tab. 5 Type of standard slaves [23]	22
Tab. 6 Slave profiles with extended addressing [23]	22
Tab. 7 Type of slaves with extended addressing [23]	23
Tab. 8 Type of standard master profiles v1.0 [23]	23
Tab. 9 Type of extended master profiles [23]	23
Tab. 10 Variable usage of InternalRAM	34
Tab. 11 Comparison of the same picture	49
Tab. 12 Example of reducing page size via “noSdram” parameter for “Home” page	50
Tab. 13 Communication Format	51
Tab. 14 User-friendly test collected data	68

15 BIBLIOGRAPHY

- [1] ŠTOHL, Radek. Presentation of subject MDSS: Sběrnice AS-Interface možnosti jediného kabelu. FEEC, Brno University of Technology. [cited 2013-4-4]. Link: dzin2.feec.vutbr.cz
- [2] BECKER, R., AS-International Accosiation. Automatizace je jednoduchá s AS-Interface. Frankfurt: AS- International Accosiation, 2008, p.96. [cited 2013-4-4].
- [3] BECKER Rolf a kol. AS-International Association. AS-Interface : Řešení pro automatizaci. [s.l.], : [s.n.], 2002, p.184.
- [4] Zakladni informace o sběrnici AS-i. In: AS-interface Česká republika [online]. [cited 2013-4-10]. Link: www.as-interface.cz.
- [5] AS- International Accosiation. Profiles. Version 3.0, Rev.3 [2010-06]. [cited 2013-4-4], p.110.
- [6] MK-480272 Data Sheet [online]. Rev. 1.2 [2011-03]. [cited 2013-4-4]. Link: http://www.mouser.com/ds/2/20/MK-480272C_Rev1-182220.pdf
- [7] Team of autors: Automatisierungs und Kommunikationssysteme FTZ HTWK Leipzig. AS-i Netzwerk Simulator. Leipzig, 2010. p. 96 [cited 2013-15-4].
- [8] PINKER J.,POUPA M.: Číslicové systémy a jazyk VHDL. BEN, Praha, 2006. ISBN 80-7300-198-5. [cited 2013-15-4].
- [9] JEAN P. Fpga4fun [online]. [2013-1-13]. [cited 2013-15-4]. Link: <http://www.fpga4fun.com/FPGAsoftware3.html>
- [10] KUBÍČEK M. Impulzní a číslicová technika, Přednáška 5 [online]. Ústav radioelektroniky VUT Brno. [cited 2013-15-4]. Link: http://www.urel.feec.vutbr.cz/~fryza/downloads/BICT_Prednaska_5.pdf
- [11] Gui design software. In: Amulettechnologies [online]. ©2012 [cited 2013-11-9]. Link: <http://www.amulettechnologies.com/products/gui-design-software>
- [12] InternalRAM, GEMstudio user guide. In: Amulettechnologies [online]. 2013, p.33 [cited 2013-11-9]. Link: <http://www.amulettechnologies.com/images/stories/Downloads/usersguide.pdf>
- [13] How to get startet with Photoshop PS6- 10 things for begginers. In: Youtube [online]. 7.10.2012 [cited 2013-11-30]. Link: <http://www.youtube.com/watch?v=OjRqZiAgoHo>. Canal of Terry White.

- [14] CHAPMAN, Cameron. Colour theory for designers, Part 1: The meaning of colour. In: Smashingmagazine [online]. Jan. 28. 2010 cited 2013-11-21]. Link: <http://www.smashingmagazine.com/2010/01/28/colour-theory-for-designers-part-1-the-meaning-of-colour/>
- [15] SIMMONS, Jason. The designer's desktop manual. Mies: RotoVision SA, 2010. ISBN:978-2-88893-094-5.
- [16] Image File Formats. In: Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 11.12.2006, last modified on 10.12.2013[cited 2013-12-20]. Link: http://en.wikipedia.org/wiki/Image_file_formats
- [17] Amulet communication protocol, GEMstudio user guide. In: Amulettechnologies [online]. 2013, p.110 [cited 2013-12-27]. Link: <http://www.amulettechnologies.com/images/stories/Downloads/usersguide.pdf>
- [18] WROBLEWSKI, Luke. Touch target size. In: Lukew [online]. May 4. 2010 [cited 2014-1-24]. Link: <http://www.lukew.com/ff/entry.asp?1085>
- [19] ZMDI. AS-I4U/AS-I4U-E/AS-I4U-F Data Sheet Rev.2.2 [online]. April 2012 [cited 2014-3-12]. Link: https://www.zmdi.com/sites/default/files/AS-I4U_Data_Sheet_Rev_2_2.pdf
- [20] DE1 Development and Education Board. In: Altera [online]. ©2014 [cited 2014-4-12]. Link: <http://www.altera.com/education/univ/materials/boards/de1/unv-de1-board.html>
- [21] FT232R USB UART IC Datasheet [online]. Version 2.10 [2012-4]. [cited 2014-3-12]. Link: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
- [22] Spartan-3 FPGA Family Data Sheet [online]. June 2013 [cited 2014-3-12]. Link: http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf
- [23] AS-International Association. Profiles. Version 3.0, Revision 3. 9. June 2010 [cited 2013-5-4].
- [24] KITAO, K., KITAO, S. K. A study of colour association differences Between Americans and Japanese. Human Communications Studies 13, 59–75. 1986 [cited 2014-4-8]. Link: <http://eric.ed.gov/PDFS/ED273134.pdf>
- [25] RUDLOFF, Tobias. Entwicklung eines FPGA basierten Test- und Diagnose-systems für den industriellen Kommunikationsstandard AS-Interface. HTWK Leipzig, Institut für Nachrichtentechnik und Informationstechnik. Leipzig, 2006, p.129. [cited 2013-4-15].

- [26] Amulet technologies STK-CY-043. In: sg.mouser [online]. April 15. 2014 [cited 2014-4-15]. Link: <http://sg.mouser.com/ProductDetail/Amulet-Technologies/STK-CY-043/?qs=sGAEpiMZZMtMbLhITnKwt5YyvwAxmpZT>
- [27] Capacitive 4.3 “GEMmodule™Duo” [online]. Version 1.0, [2012-8]. [cited 2014-4-4]. Link: <http://www.amulettechnologies.com/images/stories/Downloads/capacitive43flyer.pdf>
- [28] C.-F. LEE, W.-C. TSAI. Mapping of user interfaces on electronic appliances. *Applied Ergonomics* 38, 667-674. 2007 [cited 2014-4-8]. Link: <http://www.sciencedirect.com/science/article/pii/S0003687006001256>
- [29] From island to clouds: the data evolution. In: drillingcontractor [online]. Huston, Texas [2011-8]. [cited 2014-4-16]. Link: <http://www.drillingcontractor.org/from-islands-to-clouds-the-data-evolution-10675>
- [30] Quartus II, Design Entry and Synthesis. In: Altera [online]. ©2014 [cited 2014-4-12]. Link: <http://www.altera.com/products/software/quartus-ii/subscription-edition/design-entry-synthesis/qts-des-ent-syn.html>
- [31] GEMstarter-kit™ - STK-480272C [online]. [cited 2014-4-4]. Link: <http://www.mouser.com/ds/2/20/Amulet%20Color%20Starter%20Kit%20v%203-187887.pdf>
- [32] META Refresh Object, GEMstudio user guide. In: Amulettechnologies [online]. 2013, p.105 [cited 2013-10-27]. Link: <http://www.amulettechnologies.com/images/stories/Downloads/usersguide.pdf>

16 LIST OF APPENDICES

CD contains:

Document:

Master Thesis

Appendix A: GEMstudio Opcodes

Appendix B: GEMstudio All possible commands

Appendix C: GEMstudio Inter widget commands

Appendix P1: GEMstudio Program

Appendix P2: VHDL Program

Appendix T: User-Friendly Test

Appendix V: Video of Functionality

Software:

GEMstudio program: FASS

VHDL program: FPGA-Amulet Communication Protocol