Rapier project
A hardware interface system for Linux

Michael B. Sørensen
Århus Tekniske Skole, in Århus, Denmark.

4th of January 2005

Abstract

Language This project documentation is written in English even though the authors native language is Danish. English was chosen because of the license chosen for the hardware and software. For the project to be read and understood by most people around the world, English was the best choice.

Project subject This project is about the design, assembly and test of an interface system between a PC running Linux and external devices. The interface system are capable of handling analog and digital signals from external devices. The interface system is operated from a graphical user interface on the PC.

The author This project is about a hardware interface system designed by me and the software to control it. The project is a part of my final exam which hopefully will occur in January 2005. I'm in the final semester of a 2 year training at Århus Tekniske Skole, in Århus, Denmark.

Result The aim of the project was reached.

Development description I designed the hardware from a modular point of view. The software proved to be the biggest challenge. Especially the GUI and serial communication. I got most of the help from a Linux programming mailing list and several searches on the Internet through Google.

Contents

List of Tables

List of Figures

Chapter 1

Problem description

1.1 The main idea

This project is about designing an interface system for a PC, capable of electronically measure and control the real world. The system will be controlled through a GUI[1].

The system will be build with modules and highly scaleable. The hardware will be based on PCB's[2] on a custom bus system.

As aid in the management of project files, upload releases to the Internet (http://rapier.sourceforge.net/) and backup purpose, a project file management system is written.

Hardware:

- 1 analog ADC[3] input
- 8 digital inputs
- 8 digital outputs
- 1 analog DAC[4] output
- Monitor board
- Easy to maintain and trouble shoot
- External power supply(not included in this project)

[1]Graphical User Interface
[2]Printed Circuit Board
[3]Analog to Digital Converter
[4]Digital to Analog Converter

Software:

- Easy to use graphical interface
- Support for all hardware functionality

Project file management:

- Backup files.
- Distribute released files to sourceforge.net through ssh[5].
- Update drawings, pictures and schematics in documentation and homepage.
- Gathering source code from IDE[6] projects.
- Version control through "rsync"[7].

Practical applications

- Measure temperature.
- Detect daylight.
- Turn on relays.
- Light shows.
- DDS functionality.[8]
- Test equipment.
- Process control.

[5]In computing, Secure shell, or SSH, is both a computer program and an associated network protocol designed for logging into and executing commands on a remote computer. It is intended to replace the earlier rlogin, telnet and rsh protocols, and provides secure encrypted communications between two untrusted hosts over an insecure network.

[6]An integrated development environment (IDE) (also known as an integrated design environment and integrated debugging environment) is computer software to help computer programmers develop software.

[7]rsync is a utility which synchronizes files and directories between two locations while minimizing data transfer.

[8]Direct Digital Synthesis is a method to digitally create arbitrary waveforms and frequencies from a single source fixed frequency.

1.2 Interface connection type

There are 4 types of connection from the PC to the external interface/controller:

- PCI[9] I/O[10] card
- USB[11] I/O card
- Communication through the parallel port
- Communication through the serial port
- Communication through the IR port

1.2.1 PCI I/O card

Pro:

- Available driver means ready to go
- Relative high speed

Con:

- Expensive
- Proprietary driver

1.2.2 USB I/O card

Pro:

- Cheaper than PCI I/O card
- Connect to any modern PC

Con:

- More difficult to write a driver

[9]The Peripheral Component Interconnect standard (in practice almost always shortened to PCI) specifies a computer bus for attaching peripheral devices to a computer motherboard.

[10]Input/output, or I/O, is the collection of interfaces that different functional units (subsystems) of an information processing system use to communicate with each other, or to the signals (information) sent through those interfaces.

[11]Universal Serial Bus (USB) provides a serial bus standard for connecting devices, usually to a computer, but it also is in use on other devices such as set-top boxes, game consoles and PDAs.

1.2.3 Parallel port

Pro:

- Connect to any PC
- Cheap technology
- Driver available

Con:

- Relatively slow
- Traditionally a one way communication

1.2.4 Serial port

Pro:

- Connect to any PC
- Cheap technology
- Driver available
- Easy to communicate with

Con:

- Slowest speed

1.2.5 IR port

[12]

Pro:

- Connect to a laptop PC
- Cheap technology
- Driver available
- Easy to communicate with

[12]IR is an abbreviation of Infra Red

Con:

- Slowest speed
- Must have visual contact

Conclusion After much research and some interviews, the decision fell upon serial connection because of the above mentioned benefits.

1.2.6 External controller

The Z8 microcontroller was chosen because of available knowledge. But there are several benefits:

- Cheap
- Easy to use the serial line to communicate
- A great deal of source code

Future versions will be based on either Atmel AVR[13] microcontroller or the Hitachi H8[14] due to the fact that compilers exists for Linux operating system.

1.3 Choosing a programming language

There are several programming languages to choose between. Each language has its advantages. The project can be split into the following parts:

- Z8 microcontroller
- GUI
- Project file management

The relevant languages available:

- BASH Shell scripting (project file management)[15].
- C[16] (Z8).

[13]The Atmel AVR is a family of RISC microcontrollers from Atmel.

[14]H8 is the name of a large family of 8-bit and 16-bit microcontrollers made by Renesas Technology Corp., originating in the early 1990s within Hitachi Semiconductor and still actively evolving as of 2004.

[15]Bash is a UNIX command shell written for the GNU project. Its name is an acronym for Bourne-again shell - a pun on the Bourne shell (sh), which was an early, important UNIX shell.

[16]The C programming language is a low-level standardized programming language developed in the early 1970s by Ken Thompson and Dennis Ritchie for use on the UNIX operating system.

- C++[17]/QT[18] (GUI).

1.4 Developing the software

It is important to choose a good IDE for the task at hand. For this project I will use QT designer, Kdevelop and Z8 IDE. QT designer and Kdevelop is for Linux, and Z8 IDE is for Windows.

Very usefull tools:

- Doxygen
- rsync

1.5 Board Communication

Control signals		
Signal	Name	Description
Deselect Board	DB	Deselects the board.
Address Decode	AD	Decodes the address by comparing address and ID set with DIP switch.
Function Enable Output	FEO	Enables the output function of a board.
Function Enable Input	FEI	Enables the input function of a board.
Address Enable	AE	Latches the address for later AD.
Deselect Board	DB	Deselects all boards. Not used in current version but kept for future use.

Table 1.1: Control signals

Control signals	
Port	Description
E	Control Bus
G	Address/Data Bus
H	Analog inputs

Table 1.2: Ports and busses

1.6 Fan IN and fan OUT

19

[17] C++ (pronounced "sea plus plus") is a general-purpose computer programming language.

[18] In computer programming, the Qt toolkit is a cross-platform graphical widget toolkit for the development of GUI programs.

[19] Fan-out is a term that defines the maximum number of digital inputs that the output of a single logic gate can feed. Most transistor-transistor logic (TTL) gates can feed up to 10 other digital gates or devices. Thus, a typical TTL gate has a fan-out of 10.

1.6.1 Boards

Fan IN and fan OUT in each board is not a consideration due to the simplicity of the design. There's not enough inputs and outputs to even come near any problem.

1.6.2 System Bus

There is a limited number of boards that the Z8 can communicate with since all the boards has their address decoding enabled at all time.

For the project only 3 boards uses board selection hardware. There are not enough boards to raise any concern about Fan IN and Fan OUT.

It is possible to enable "Hi Drive" on the Z8 to handle the load. But it is not used in this project.

1.7 Ground

To reduce noise, special consideration has been given to the design of cabled connections. Even numbered wires in the cable are grounded and odd numbered wires are used for signals.

1.8 Distributed CPU power

A design consideration: Where to leave the task of computing the instructions?

If the largest part of instruction handling lies within the Z8, the Z8 will only get small instructions to carry out. The Z8 will need to have the full driver of every board. The Z8 will only send the result of an instruction back to the PC.

Pro:

- Less communication.
- Less PC code.

Con:

- More Z8 code.
- For every new kind of board, the Z8 code needs update.

If the largest part of instruction handling lies within the PC, the Z8 will be used as a "serial to parallel converter". The PC will need to have the full driver of every board. The Z8 will only receive and send simple port instructions back and forth to the PC.

Pro:

- Less Z8 code.
- Little or no update of Z8 code for a new board.

Con:

- A lot more communication.
- More PC code.

Conclusion Since the boards use the same addressing routine, and a board can be either input or output, it has been decided to adopt a design in the middle of the road.

The Z8 receives only 4 different instructions:

- Single ADC operation
- Addressing a board
- Byte output
- Byte input

An Addressing instruction is normally followed by a Byte In or a Byte Out instruction.

More of this is described in the protocol section on page 72.

1.9 Decoupling capacitors

For this project all decoupling capacitors has the value of 100 nF of the ceramic type.

Chapter 2

Specifications

2.1 Hardware

2.1.1 General specifications

For all the hardware in the system certain criteria has to be met.

- All Boards using 5V, -5V, 12V or 12V regulated power supply.
- Operating temperature: indoor temperature.

2.1.2 Z8

For this project a Z8 Evaluation Board is used. The Z8 must have the following specifications:

- 2 full ports available for digital signal use.
- 2 inputs available for analog signal use.
- Onboard connectors for the Adaptor Board.

The Z8 evaluation board is not a part of the project and not described in detail. Hardware not a part of the project.

2.1.3 Adaptor Board

The purpose of the Adaptor Board is to make a physical stable connector to the Bus System.

- Power supplied from the Z8 board.

2.1.4 Bus system

The purpose of the Bus System is to connect the Adaptor Board with several Boards.

- Based on a flat cable.
- By turns using signal and ground in the cable wires.
- 16 digital signal wires.
- 1 analog signal wire.
- 25 ground wires.
- 4 voltage supply wires.
- 4 unused wires.

2.1.5 Monitoring and Power Board

The purpose of the Monitoring and Power Board is to monitor signals on the Address/Data bus and Control bus on the Bus System, and to supply the external power into the Bus System.

- Input for external Power Supply.
- Monitor the digital signals using LED's.[1]

2.1.6 Digital Input Board

The purpose of the Digital Input Board is to transfer 8 bit from the input to the Z8.

- 8 bit Input.
- Galvanic separation using opto couplers.[2]

2.1.7 Digital Output Board

The purpose of the Digital Output Board is to transfer 8 bit from the Z8 to the output.

- 8 bit latched outputs.

[1] A light-emitting diode (LED) is a semiconductor device that emits incoherent monochromatic light when electrically biased in the forward direction.

[2] An opto-isolator/opto-coupler is a device that uses optical techniques to electrically isolate two related circuits, typically a transmitter and a receiver.

2.1.8 Analog Input Board

The purpose of the Analog Input Board is to provide surge protection in the analog signal and transfer it to the Z8.

- 1 analog input.
- Signal levels: 0 - 2 V
- Surge protection levels: lower than -0.6 V and higher than +5.6 V

2.1.9 Analog Output Board

The purpose of the Analog Output Board is to make an analog signal.

- 1 analog output.
- Signal levels: 0 - 2 V

2.1.10 Power Supply

The Power Supply is donated to the project, and therefor not described in details.

- 5V regulated.
- -5V regulated.
- 12V regulated.
- -12V regulated.
- Not a part of the project.

2.2 Software

2.2.1 General specifications

- GPL license.[3]
- Inline documentation for Doxygen.[4]

2.2.2 Z8

- Communication protocol.

[3]The GNU General Public License is a free software license, created by the Free Software Foundation, version 2 was released in 1991. It is usually abbreviated to GNU GPL, or, simply, GPL.

[4]Doxygen is a documentation system for C++, C, Java, IDL (Corba and Microsoft flavors) and to some extent Objective-C, PHP, C# and D.

2.2.3 GUI

- Normal use.

2.2.4 Serial Communication

- Resistant to noise.

2.2.5 3rd. party software

- Bash version 2.x
- KDE version 3.x[5]

[5]KDE (K Desktop Environment) is a free desktop environment and development platform built with Trolltech's Qt toolkit.

Chapter 3

Hardware Block/Module description

3.1 System overview

The whole system overview.

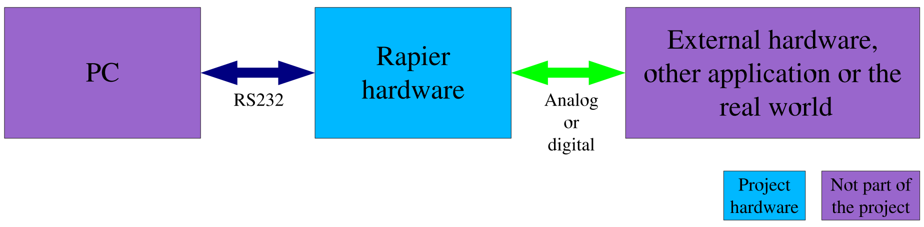


Figure 3.1: System overview

This project is about the hardware that interfaces a PC together to external hardware(the real physical world) through RS232[1], and the software to operate the project hardware.

[1]RS-232 (also referred to as EIA RS-232C or V.24) is a standard for serial binary data interchange between a DTE (Data terminal equipment) and a DCE (Data communication equipment).

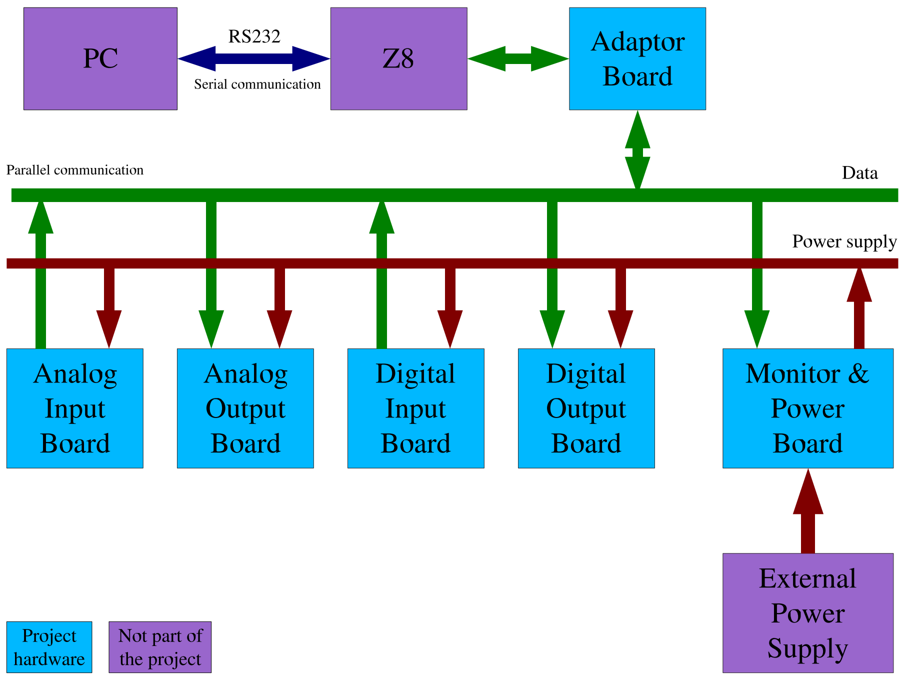


Figure 3.2: Detailed system overview

3.1.1 Description

PC The system is operated from a PC. The PC communicate with the project hardware through a serial connection over RS232 standard. This communication is bi-directional.

External power supply This external power supply provides power for the input/output boards and the monitor board. It is not a part of the project.

Z8 This microcontroller receives commands from the PC and executes them by sending signals to the boards and return the results to the PC if required. Some commands is followed by address or data, some are not. Not all commands requires data to be returned.

Adaptor Board This board connects the Z8 with the Bus System/flat cable. It provides a very stable connection and the possibility to switch off the bus for maintenance purpose.

Monitor and Power Board To supply the other boards with power, this board is connected to external power supplies. The board has LED's showing the activity on the Address/Data bus and the Control Bus.

Analog Input Board This board connects an external analog voltage through to the Z8 microcontroller with its ADC. The board provide some protection for surges in the input voltage.

Analog Output Board This board generates a voltage ranging from 0V - 2V. The board is based on a 8-bit DAC.

Digital Input Board This board latches 8 TTL-level inputs through opto couplers and sends those 8 bit to the Z8 microcontroller, which sends them to the PC.

Digital Output Board This board sends 8 bit from the Z8 microcontroller to the the 8 output terminals through a latch.

3.1.2 Calibration of hardware

The calibrated is located in section 3.7.8 on page 62.

3.2 The Bus System

The purpose of the Bus System is to carry the signals back and forth between the boards and the Z8.

The cable used is a SCSI[2] cable with 50 wires. This cable is chosen because of the number of wires and availability of tools to add extra connectors on the cable. The even numbered wires are connected to ground to suppress noise. The odd numbered wires are connected to signals. All signals are TTL level.

Here's a description of the wires in the flat cable:

Bus System Description	
Signal	Name
1	AD0
3	AD1
5	AD2
7	AD3
9	AD4
11	AD5
13	AD6
15	AD7
17	Address Enable
19	Function Enable Output
21	Address Decode
23	Deselect Board
25	Function Enable Input
27 - 39	No connection
41	Analog Input
43	+12V
45	-12V
47	+5V
49	-5V
Even pin numbers	Ground

Table 3.1: Bus System Description

The signals are split up in groups:

- Address/Data in 8 wires.
- Control bus in 5 wires.
- 4 power supply voltages in 4 wires.

[2]SCSI stands for "Small Computer System Interface", and is a standard interface for transferring data between devices on a computer bus.

- 1 analog input in 1 wire.
- Several unused wires.

Each group of connections are also grouped on the cable. The power supply voltages and analog input are kept in one side of the cable, and the rest in the other side of the cable to reduce the noise on the cable.

3.3 Adaptor board

3.3.1 Purpose

The purpose of this Adaptor Board is to connect the bus to the Z8 board. The need for good and mechanical stable connections is paramount for electrical stability.

The board had 2 switches for maintenance and testing purpose. It is possible to switch off the whole bus to separately test the signals from the Z8 board and the Rapier hardware.

3.3.2 Photo

Here's a photo of the Adaptor Board.

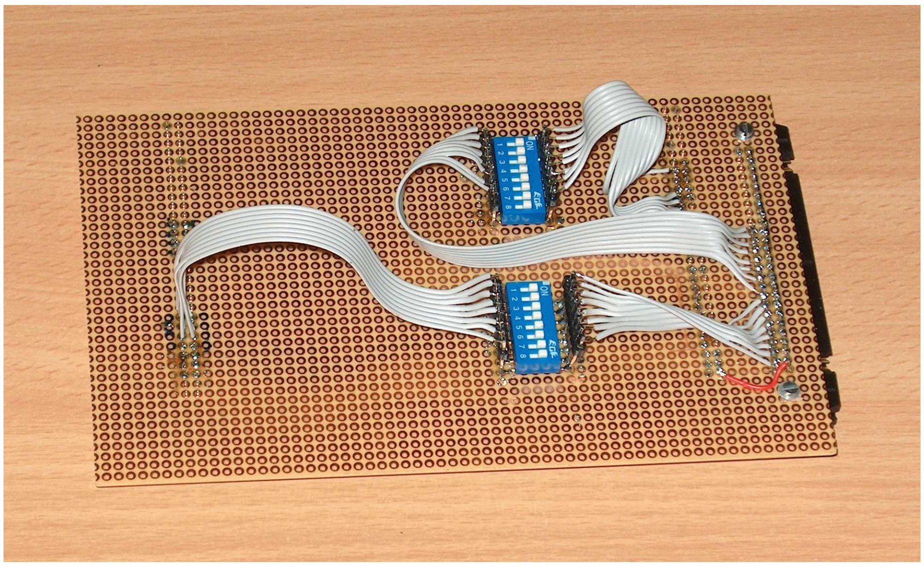


Figure 3.3: Adaptor Board Photo.

3.3.3 Block diagram

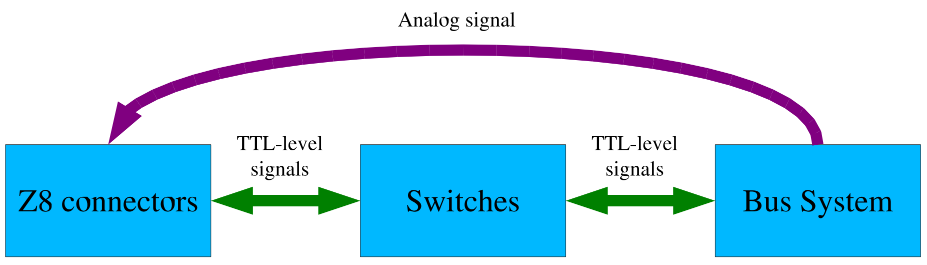


Figure 3.4: Adaptor Board block diagram

3.3.3.1 Description

This board connects the Z8 evaluation board to the Bus System. It does not process any of the signal it transfers. For maintenance purpose it is possible to disconnect the signals by switching them off on the switches.

3.3.4 Bus communication

The Adaptor Board does not perform any processing of any signals. The default setting of operation is when the switches is on.

3.3.5 Calculations

No calculations is needed for this board. Thereś no active components on this board.

3.3.6 Schematics

This is the schematic of the Adaptor Board.

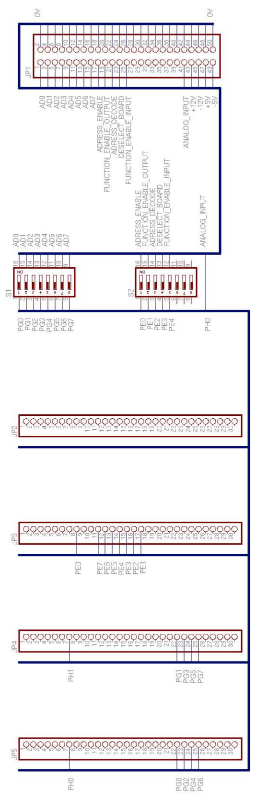


Figure 3.5: Adaptor Board Schematics.

3.3.7 Partlist

This is the partlist for the Adaptor Board.

```
Partlist

Exported from ab.sch at 11/28/2004 13:18:58

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device      Package  Library     Sheet

JP1                     PINHD-2X25  2X25     pinhead     1
JP2                     PINHD-1X30  1X30     pinhead     1
JP3                     PINHD-1X30  1X30     pinhead     1
JP4                     PINHD-1X30  1X30     pinhead     1
JP5                     PINHD-1X30  1X30     pinhead     1
S1                      DA08        DA-08    switch-dil  1
S2                      DA08        DA-08    switch-dil  1
```

3.3.8 Test procedures

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board on the Z8 board and test that all pins have levels according a test program.

Conclusion: The board is working as planned.

3.4 Monitoring and power board

3.4.1 Purpose

The purpose of this board is to monitor the bus activity and to connect the other boards to the external power supplies.

Monitoring is done by observing the LED's on the bus.

3.4.2 Photo

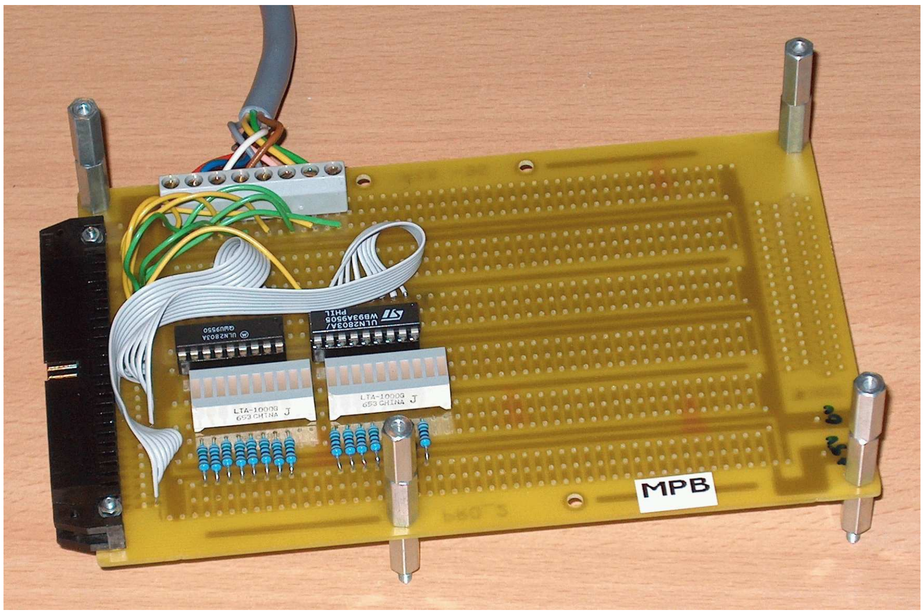


Figure 3.6: Monitoring and Power Board Photo.

3.4.3 Block diagram

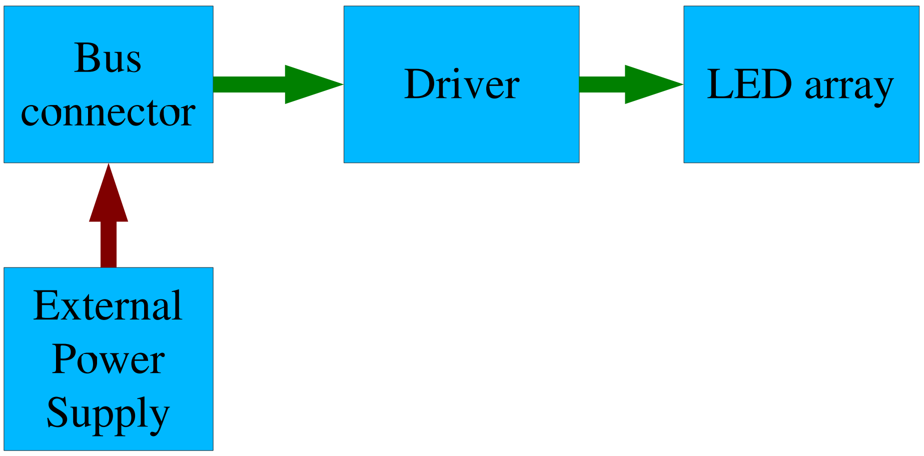


Figure 3.7: Monitoring and Power Board Block Diagram.

3.4.3.1 Description

The external power supply is connected directly to the bus system. The Driver and LED array shows what activity there is on the AD bus and the control bus.

3.4.4 Bus communication

The board does not perform any signal processing at all.

3.4.5 Calculations

3.4.5.1 Darlington output

[3]

$$U_R = 2,5V \tag{3.1}$$

$$U_D = 1,6V \tag{3.2}$$

$$U_{CE} = V_{CE(sat)} = 0,9V \tag{3.3}$$

$$I_B = 0,93mA \tag{3.4}$$

[3]Darlington refers to a coupling of 2 transistors.

$$I_R = 10,0mA \tag{3.5}$$

$$R = 250\Omega \tag{3.6}$$

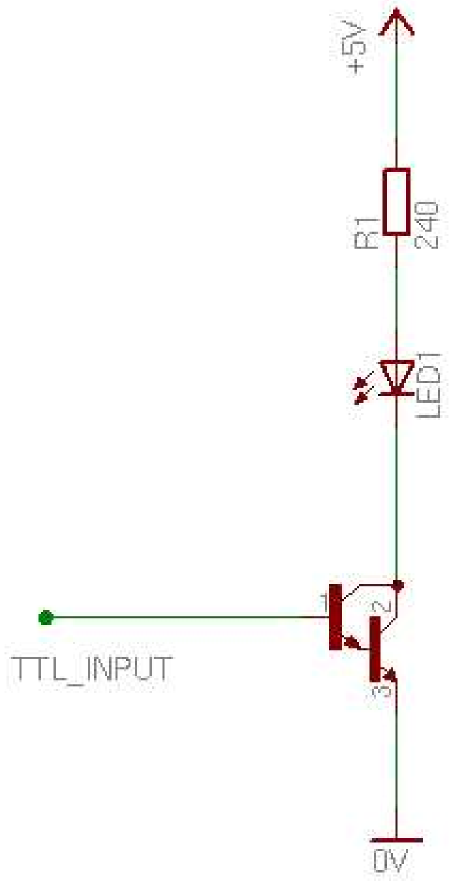


Figure 3.8: Darlington

3.4.6 Schematics of the board

This schematic of the Monitor and Power board.

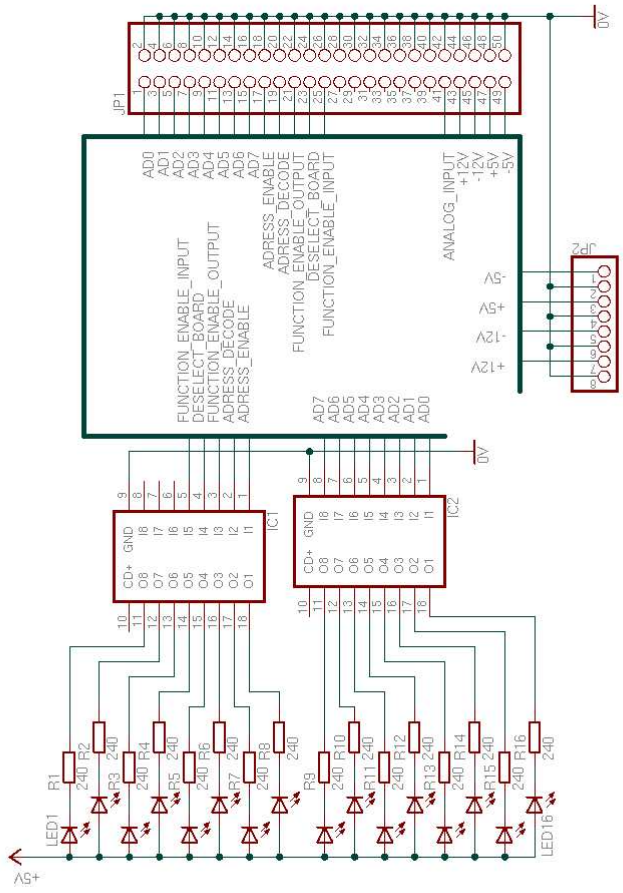


Figure 3.9: Monitoring and Power Board schematic.

3.4.7 Schematics of the power cable

The schematic is of power cable.

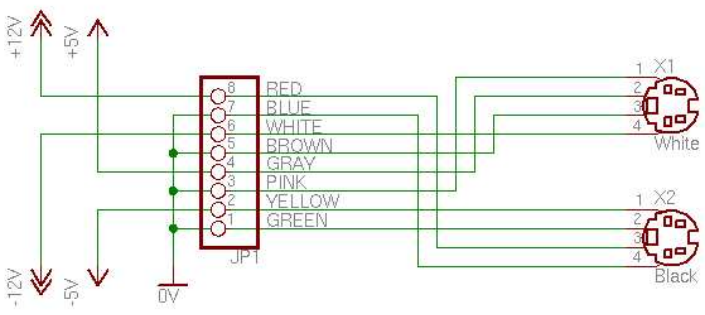


Figure 3.10: Power Cable schematic.

3.4.8 Partlist

This is the partlist of the Monitor and Power Board.

```
Partlist

Exported from mpb.sch at 12/22/2004 18:04:07

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device         Package  Library  Sheet

IC1                     ULN2803A       DIL18    uln-udn  1
IC2                     ULN2803A       DIL18    uln-udn  1
JP1                     PINHD-2X25     2X25     pinhead  1
JP2                     PINHD-1X8      1X08     pinhead  1
LED1                    LED5MM         LED5MM   led      1
LED2                    LED5MM         LED5MM   led      1
LED3                    LED5MM         LED5MM   led      1
LED4                    LED5MM         LED5MM   led      1
LED5                    LED5MM         LED5MM   led      1
LED6                    LED5MM         LED5MM   led      1
LED7                    LED5MM         LED5MM   led      1
LED8                    LED5MM         LED5MM   led      1
LED9                    LED5MM         LED5MM   led      1
LED10                   LED5MM         LED5MM   led      1
LED11                   LED5MM         LED5MM   led      1
LED12                   LED5MM         LED5MM   led      1
LED13                   LED5MM         LED5MM   led      1
LED14                   LED5MM         LED5MM   led      1
LED15                   LED5MM         LED5MM   led      1
LED16                   LED5MM         LED5MM   led      1
R1       240            R-EU_0204/2V 0204V    rcl      1
R2       240            R-EU_0204/2V 0204V    rcl      1
R3       240            R-EU_0204/2V 0204V    rcl      1
R4       240            R-EU_0204/2V 0204V    rcl      1
R5       240            R-EU_0204/2V 0204V    rcl      1
R6       240            R-EU_0204/2V 0204V    rcl      1
R7       240            R-EU_0204/2V 0204V    rcl      1
R8       240            R-EU_0204/2V 0204V    rcl      1
R9       240            R-EU_0204/2V 0204V    rcl      1
R10      240            R-EU_0204/2V 0204V    rcl      1
R11      240            R-EU_0204/2V 0204V    rcl      1
R12      240            R-EU_0204/2V 0204V    rcl      1
R13      240            R-EU_0204/2V 0204V    rcl      1
R14      240            R-EU_0204/2V 0204V    rcl      1
R15      240            R-EU_0204/2V 0204V    rcl      1
R16      240            R-EU_0204/2V 0204V    rcl      1
```

3.4.9 Test procedure

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board to the bus system.
4. Connect the board to external power.
5. Observe the LED array for bus activity.

Conclusion: The board is working as planned.

3.5 Digital Output Board

3.5.1 Purpose

The purpose of this board is to set the TTL level output according to the byte received from the Z8.

3.5.2 Photo

Photo of the Digital Output Board.

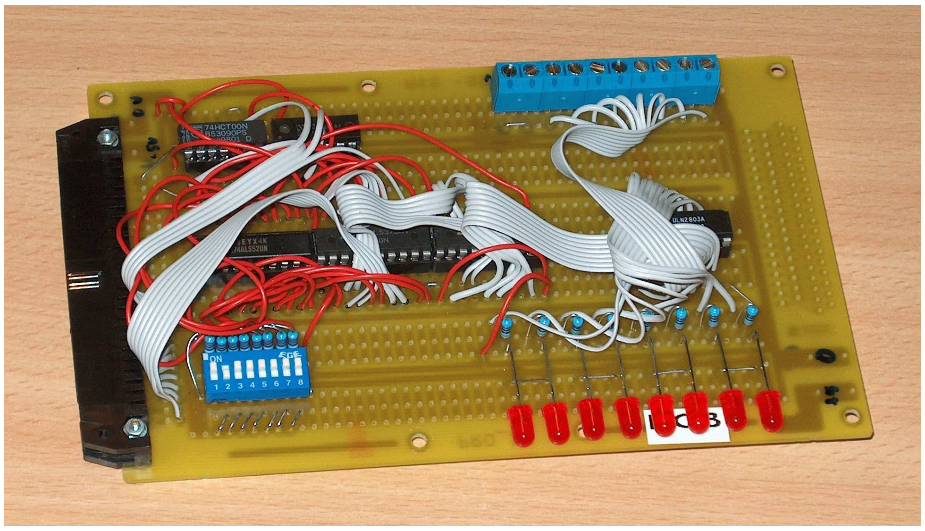


Figure 3.11: Digital Output Board Photo.

3.5.3 Block diagram

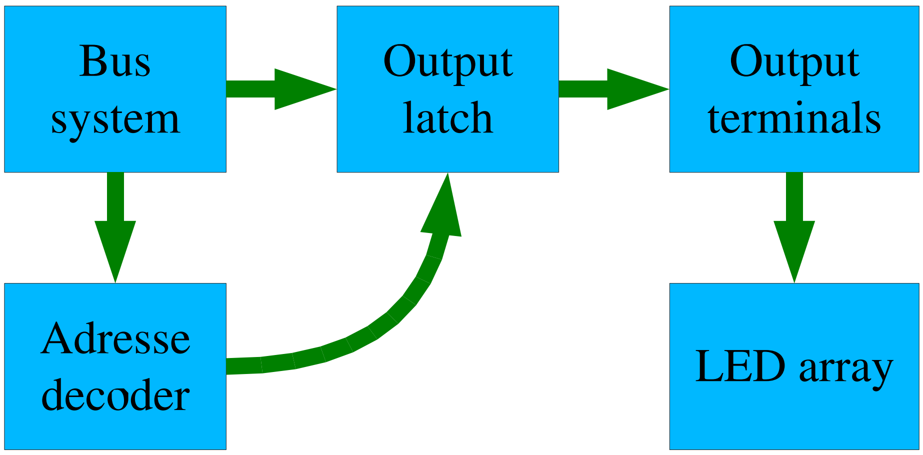


Figure 3.12: Digital Output Board Block Diagram.

Bus system This is connected to the flat cable.

Address decode This module compare the address with the board id.

Output latch This latch is activated when the proper address is received and outputs the data received from the Z8 microcontroller.

Output terminals These are output terminals to be connected to external hardware.

LED array This LED array are for monitoring the output.

3.5.4 Bus communication

Overview of address and control signals to use this board:

1. Address the board
2. Send data to output

3.5.4.1 Adressing a board

Description of signals The tables below are based on TTL-level inputs:

Step The column tells the step in the routine.

AD bus	This column is the 8 bit bus which can hold an adress or data.
Adress Enable	This column is a signal from the Z8 micro controller.
Adress Decode	This column is a signal from the Z8 micro controller.
Deselect Board	This column is a signal from the Z8 micro controller.
State	This column shows whether the board is selected or not.

Selecting a board There is only one way to adress a board and that is to send an adress and decode it.

Selecting a board					
Step	**AD bus**	**Adress Enable**	**Adress Decode**	**Deselect Board**	**State**
1	X	0	0	0	Not selected
2	Valid adress	0	0	0	Not selected
3	Valid adress	1	1	0	Selected
4	Valid adress	0	0	0	Selected
5	X	0	0	0	Selected

Table 3.2: Selecting a board

Explanation First the Z8 send out an adress on the AD-bus. Then the Z8 tells that it is an adress by setting Adress Enable active high. This starts the adress/id comparing. By sending the Adress Decode the result of adress/id compare is stored in a flip-flop. By now the board is selected until a Deselect Board or decoding of an invalid adress.

Deselecting a board method 1

The fastest way to deselect a board is to select another board. When trying to decode an invalid adress, the board deselects it self automatically.

Deselecting a board method 1					
Step	**AD bus**	**Adress Enable**	**Adress Decode**	**Deselect Board**	**State**
1	X	0	0	0	Selected
2	Invalid adress	0	0	0	Selected
3	Invalid adress	1	1	0	Not selected
4	Invalid adress	0	0	0	Not selected
5	X	0	0	0	Not selected

Table 3.3: Deselecting a board method 1

Explanation When the board try to decode an invalid adress and at the same time recieves an Adress Decode, the result is stored in the flip-flop and there by disabling further board functions.

Deselecting a board method 2

The most obvious way to deselect a board is to deselect all boards in the system. This can be done by using a function, but it is not used in the Z8 code, but it is available for future use.

Deselecting a board method 2					
Step	**AD bus**	**Adress Enable**	**Adress Decode**	**Deselect Board**	**State**
1	X	0	0	0	Selected
2	X	0	0	1	Not selected
3	X	0	0	0	Not selected

Table 3.4: Deselecting a board method 2

Explanation A Deselect Board forces the flip-flop into "not selected" state.

3.5.4.2 Output functions to a board

Description of signals The tables below are based on TTL-level inputs:

Step	The column tells the step in the routine.
Board Select State	This column tells if the board is selected or not.
AD bus direction	Which way is data supposed to flow.
AD bus data	What is on the AD bus.
Output Function Enable	Signal from the Z8 to active the board input function, typically a latch.

Output functions to a board			
Step	**Board Selected State**	**AD bus**	**Input Function Enable**
1	Selected	Valid data	0
2	Selected	Valid data	0
3	Selected	Valid data	1
4	Selected	Valid data	0
5	Selected	Valid data	0

Table 3.5: Output functions to a board

Explanation The board uses a second latch to handle data instead of adress. The data is send forward to, typically, an output terminal or a DAC. When data is ready on the AD bus, the board recieves an Output Function Enable signal to enable the second latch.

3.5.5 Calculations

3.5.5.1 8 bit comparator input

$$V_{IL} = 0.8V \tag{3.7}$$

$$U_R = 4.2V \tag{3.8}$$

$$I_R = 0.1mA \tag{3.9}$$

$$R = \frac{U_R}{I_R} = \frac{4.2}{0.1m} = 42K\Omega \tag{3.10}$$

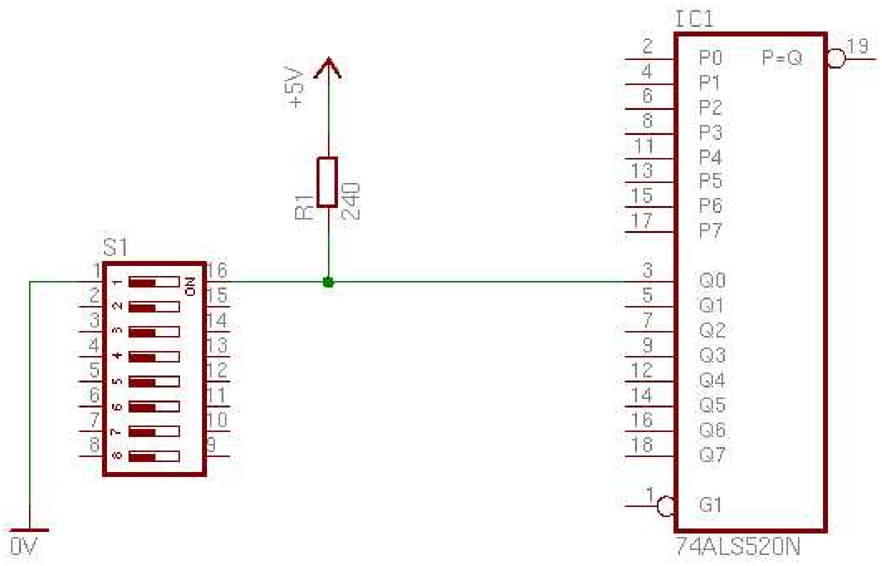


Figure 3.13: 8 bit comparator input

3.5.5.2 Darlington output

See calculations section 3.4.5.1 on page 39.

3.5.6 Schematics

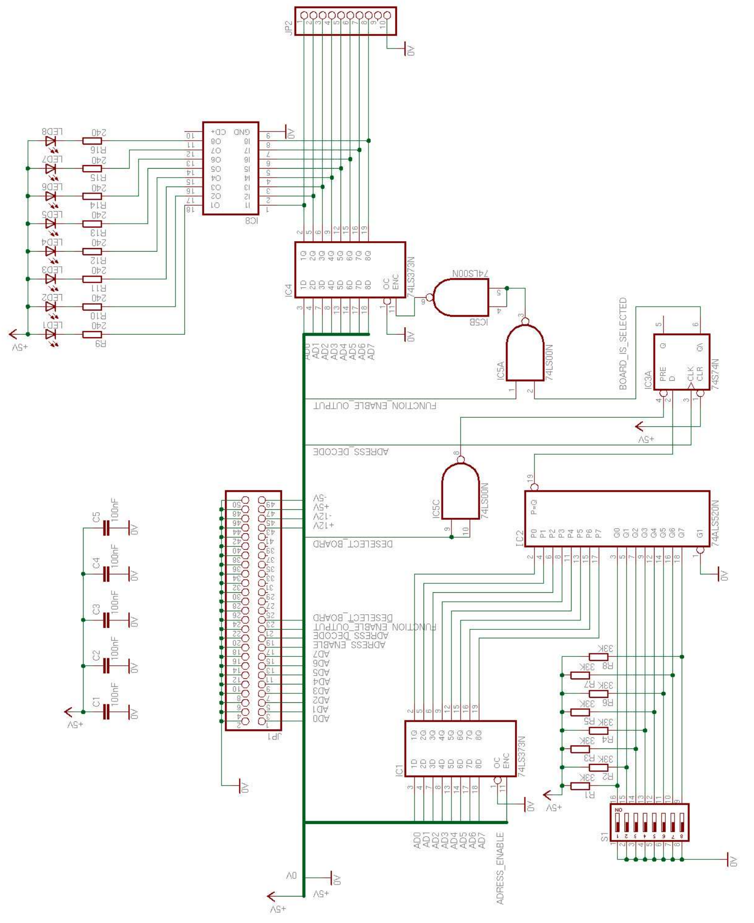


Figure 3.14: Digital Output Board schematic.

3.5.7 Partlist

```
Partlist

Exported from dob.sch at 12/22/2004 17:49:47

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device          Package       Library    Sheet

C1       100nF          C-EU025-024X044 C025-024X044 rcl        1
C2       100nF          C-EU025-024X044 C025-024X044 rcl        1
C3       100nF          C-EU025-024X044 C025-024X044 rcl        1
C4       100nF          C-EU025-024X044 C025-024X044 rcl        1
C5       100nF          C-EU025-024X044 C025-024X044 rcl        1
IC1      74LS373N       74LS373N        DIL20         74xx-us    1
```

IC2	74ALS520N	74ALS520N	DIL20	74xx-eu	1
IC3	74S74N	74S74N	DIL14	74xx-us	1
IC4	74LS373N	74LS373N	DIL20	74xx-us	1
IC5	74LS00N	74LS00N	DIL14	74xx-us	1
IC8		ULN2803A	DIL18	uln-udn	1
JP1		PINHD-2X25	2X25	pinhead	1
JP2		PINHD-1X10	1X10	pinhead	1
LED1		LED5MM	LED5MM	led	1
LED2		LED5MM	LED5MM	led	1
LED3		LED5MM	LED5MM	led	1
LED4		LED5MM	LED5MM	led	1
LED5		LED5MM	LED5MM	led	1
LED6		LED5MM	LED5MM	led	1
LED7		LED5MM	LED5MM	led	1
LED8		LED5MM	LED5MM	led	1
R1	33K	R-EU_0207/10	0207/10	rcl	1
R2	33K	R-EU_0207/10	0207/10	rcl	1
R3	33K	R-EU_0207/10	0207/10	rcl	1
R4	33K	R-EU_0207/10	0207/10	rcl	1
R5	33K	R-EU_0207/10	0207/10	rcl	1
R6	33K	R-EU_0207/10	0207/10	rcl	1
R7	33K	R-EU_0207/10	0207/10	rcl	1
R8	33K	R-EU_0207/10	0207/10	rcl	1
R9	240	R-EU_0204/7	0204/7	rcl	1
R10	240	R-EU_0204/7	0204/7	rcl	1
R11	240	R-EU_0204/7	0204/7	rcl	1
R12	240	R-EU_0204/7	0204/7	rcl	1
R13	240	R-EU_0204/7	0204/7	rcl	1
R14	240	R-EU_0204/7	0204/7	rcl	1
R15	240	R-EU_0204/7	0204/7	rcl	1
R16	240	R-EU_0204/7	0204/7	rcl	1
S1		DA08	DA-08	switch-dil	1

3.5.8 Test procedures

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board to the bus system.
4. Run the GUI program and test the board by visual inspection of the LED and match their lights with the setting in the GUI.

Conclusion: The board is working as planned.

3.6 Digital Input Board

3.6.1 Purpose

The purpose of this board is to transfer 8 bit TTL-level inputs to the Z8 microcontroller.

3.6.2 Photo

This is the Digital Input Board.

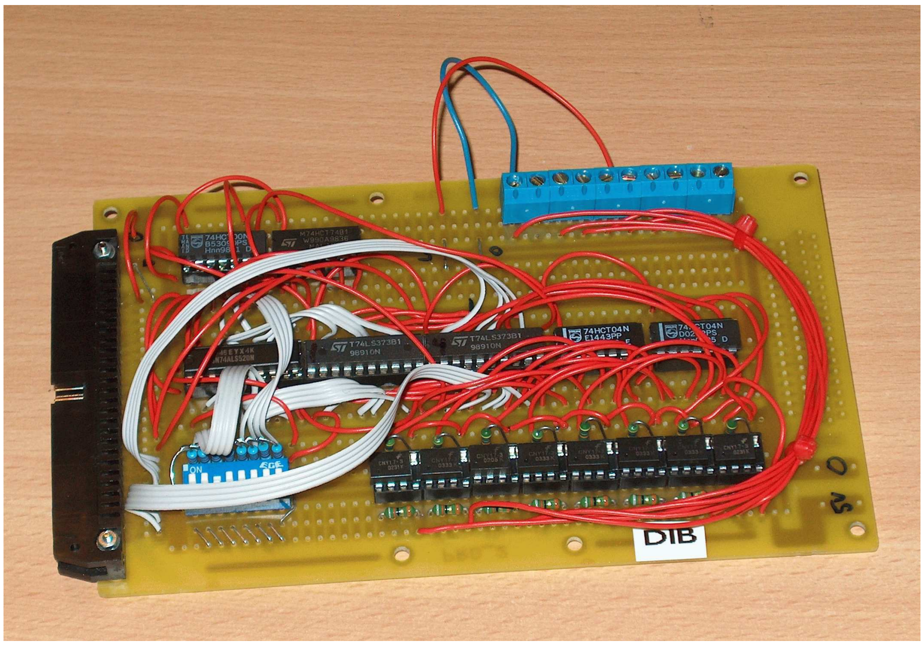


Figure 3.15: Digital Input Board Photo.

3.6.3 Block diagram

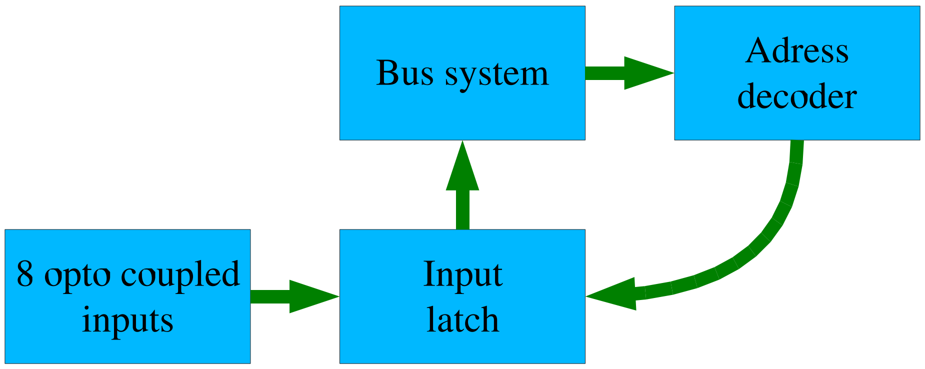


Figure 3.16: Digital Input Board Block Diagram.

3.6.4 Bus communication

Overview of address and control signals to use this board:

1. Address the board
2. Receive data from input

3.6.4.1 Addressing a board

Please look at section 3.5.4.1 on page 45.

Input functions to a board

3.6.4.2 Input functions to a board

Description of signals The tables below are based on TTL-level inputs:

Step The column tells the step in the routine.

Board Select State This column tells if the board is selected or not.

AD bus direction Which way is data supposed to flow.

AD bus data What is on the AD bus.

Input Function Enable Signal from the Z8 to active the board input function, typically a latch.

Input functions to a board				
Step	**Board Selected State**	**AD bus direction**	**AD bus Data**	**Input Function Enable**
1	True	From Z8	Don't care	0
2	True	To Z8	Data from input latch	1
3	True	From Z8	Don't care	0

Table 3.6: Input functions to a board

Explanation When the board recieves an Input Function Enable, the board enables the latch and sends the data back to the Z8 microcontroller. Before the Input Function Enable is send, the Z8 microcontroller turns the bus from output mode to input mode. After the Input Function Enable, the Z8 turns the bus back again to output mode. This is hard coded in the Z8 software to prevent the board bus output and the Z8 microcontroller output to transmit at the same time.

3.6.5 Calculations

3.6.5.1 8 bit comparator input

See calculations in section 3.5.5.1 on page 48.

3.6.5.2 Opto coupler output

$$I_F = 10mA \tag{3.11}$$

$$I_C = 2.5mA \tag{3.12}$$

$$V_{CE(sat)} = 0.4V \tag{3.13}$$

$$U_{R_C} = 4.6V \tag{3.14}$$

$$R_C = \frac{U_{R_C}}{I_C} = \frac{4.6}{10m} = 460\Omega \tag{3.15}$$

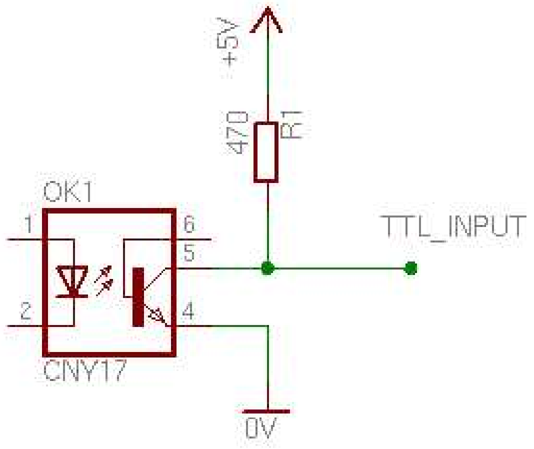


Figure 3.17: Opto coupler output

3.6.5.3 Opto coupler input

$$I_F = 10mA \tag{3.16}$$

$$V_F = 1,15V \tag{3.17}$$

$$U_{in} = 2.0V \tag{3.18}$$

$$U_{R1} = U_{in} - V_F = 2.0 - 1.15 = 0.85V \tag{3.19}$$

$$R1 = \frac{U_{R1}}{I_F} = \frac{0.85}{10m} = 85\Omega \tag{3.20}$$

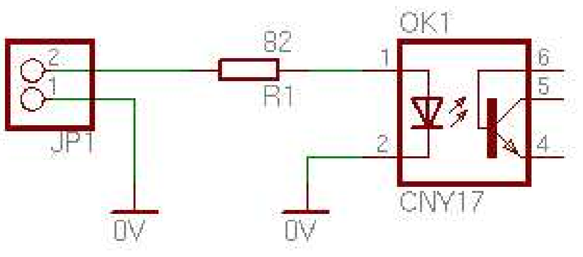


Figure 3.18: Opto coupler input

3.6.5.4 Darlington output

See calculations section 3.4.5.1 on page 39.

3.6.6 Schematics

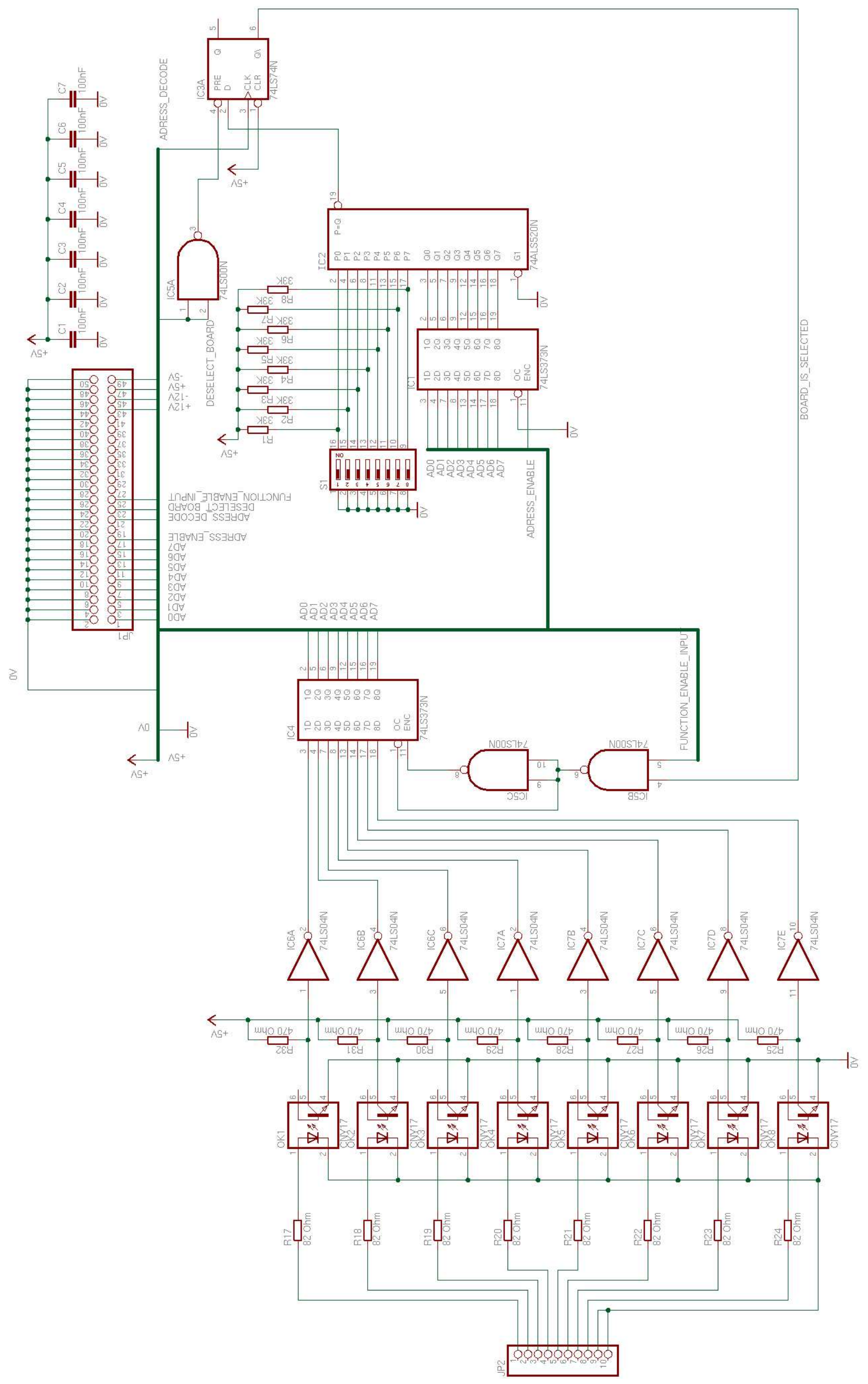


Figure 3.19: Digital Input Board schematic.

3.6.7 Partlist

```
Partlist

Exported from dib.sch at 12/22/2004 17:33:03

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device          Package      Library     Sheet

C1       100nF          C-EU025-024X044 C025-024X044 rcl         1
C2       100nF          C-EU025-024X044 C025-024X044 rcl         1
C3       100nF          C-EU025-024X044 C025-024X044 rcl         1
C4       100nF          C-EU025-024X044 C025-024X044 rcl         1
C5       100nF          C-EU025-024X044 C025-024X044 rcl         1
C6       100nF          C-EU025-024X044 C025-024X044 rcl         1
C7       100nF          C-EU025-024X044 C025-024X044 rcl         1
IC1      74LS373N       74LS373N        DIL20        74xx-us     1
IC2      74ALS520N      74ALS520N       DIL20        74xx-eu     1
IC3      74LS74N        74LS74N         DIL14        74xx-us     1
IC4      74LS373N       74LS373N        DIL20        74xx-us     1
IC5      74LS00N        74LS00N         DIL14        74xx-us     1
IC6      74LS04N        74LS04N         DIL14        74xx-eu     1
IC7      74LS04N        74LS04N         DIL14        74xx-eu     1
JP1                     PINHD-2X25      2X25         pinhead     1
JP2                     PINHD-1X10      1X10         pinhead     1
OK1      CNY17          CNY17           DIL06        optocoupler 1
OK2      CNY17          CNY17           DIL06        optocoupler 1
OK3      CNY17          CNY17           DIL06        optocoupler 1
OK4      CNY17          CNY17           DIL06        optocoupler 1
OK5      CNY17          CNY17           DIL06        optocoupler 1
OK6      CNY17          CNY17           DIL06        optocoupler 1
OK7      CNY17          CNY17           DIL06        optocoupler 1
OK8      CNY17          CNY17           DIL06        optocoupler 1
R1       33K            R-EU_0207/10    0207/10      rcl         1
R2       33K            R-EU_0207/10    0207/10      rcl         1
R3       33K            R-EU_0207/10    0207/10      rcl         1
R4       33K            R-EU_0207/10    0207/10      rcl         1
R5       33K            R-EU_0207/10    0207/10      rcl         1
R6       33K            R-EU_0207/10    0207/10      rcl         1
R7       33K            R-EU_0207/10    0207/10      rcl         1
R8       33K            R-EU_0207/10    0207/10      rcl         1
R17      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R18      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R19      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R20      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R21      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R22      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R23      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R24      82 Ohm         R-EU_0204/2V    0204V        rcl         1
R25      470 Ohm        R-EU_0204/2V    0204V        rcl         1
R26      470 Ohm        R-EU_0204/2V    0204V        rcl         1
R27      470 Ohm        R-EU_0204/2V    0204V        rcl         1
R28      470 Ohm        R-EU_0204/2V    0204V        rcl         1
```

R29	470 Ohm	R-EU_0204/2V	0204V	rcl	1
R30	470 Ohm	R-EU_0204/2V	0204V	rcl	1
R31	470 Ohm	R-EU_0204/2V	0204V	rcl	1
R32	470 Ohm	R-EU_0204/2V	0204V	rcl	1
S1		DA08	DA-08	switch-dil	1

3.6.8 Test procedure

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board to the bus system.
4. Connect the board inputs to an 8 bit signal source.
5. Run the GUI and observe the inputs in GUI window and compare the result with the signal input.

Conclusion: The board is working as planned.

3.7 Analog Output Board

3.7.1 Purpose

The purpose of this board is to generate an analog voltage through a 8 bit DAC. The byte is received from the Z8 microcontroller in a latch, and used as input in the DAC.

3.7.2 Photo

This is the Analog Output Board.

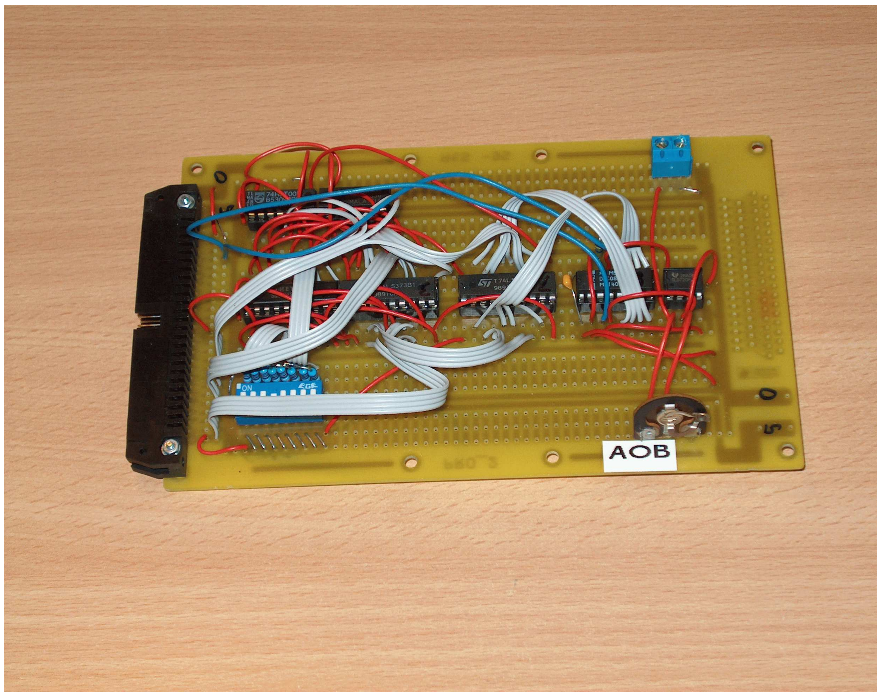


Figure 3.20: Analog Output Board Photo.

3.7.3 Block diagram

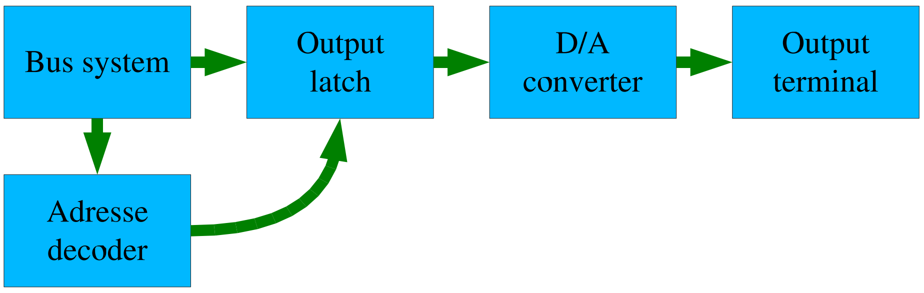


Figure 3.21: Analog Output Board block diagram.

3.7.4 Bus communication

Overview of address and control signals to use this board:

1. Address the board
2. Send data to output

3.7.4.1 Addressing a board

Please look at section 3.5.4.1 on page 45.

3.7.4.2 Output functions to a board

Please look at section 3.5.4.2 on page 47.

3.7.5 Calculations

3.7.5.1 8 bit comparator input

See calculations in section 3.5.5.1 on page 48.

3.7.5.2 Digital to analog converter

R1 and R2 are based on values set in the Application Note. The Application Note suggests the resistor value to be $5K\Omega$, low tolerence. The value of R1 is $4.7K\Omega$ and R2 is $1K\Omega$. R1 is a trim. pot. because the value can not be easily calculated.

Look at page 62 for calibration procedure.

Each resistor is a part in a current loop.

3.7.6 Schematics

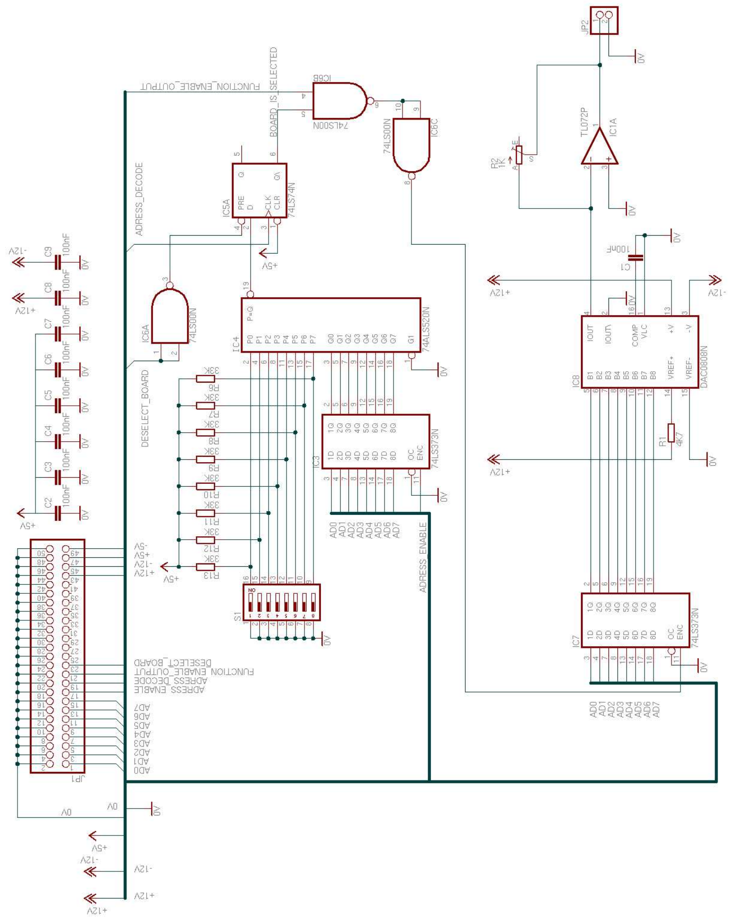


Figure 3.22: Analog Output Board schematic.

3.7.7 Partlist

```
Partlist

Exported from aob.sch at 12/22/2004 17:29:24

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device          Package      Library    Sheet

C1       100nF          C-EU050-024X044 C050-024X044 rcl        1
C2       100nF          C-EU025-024X044 C025-024X044 rcl        1
C3       100nF          C-EU025-024X044 C025-024X044 rcl        1
```

C4	100nF	C-EU025-024X044	C025-024X044	rcl	1
C5	100nF	C-EU025-024X044	C025-024X044	rcl	1
C6	100nF	C-EU025-024X044	C025-024X044	rcl	1
C7	100nF	C-EU025-024X044	C025-024X044	rcl	1
C8	100nF	C-EU025-024X044	C025-024X044	rcl	1
C9	100nF	C-EU025-024X044	C025-024X044	rcl	1
IC1	TL072P	TL072P	DIL08	linear	1
IC3	74LS373N	74LS373N	DIL20	74xx-us	1
IC4	74ALS520N	74ALS520N	DIL20	74xx-eu	1
IC5	74LS74N	74LS74N	DIL14	74xx-eu	1
IC6	74LS00N	74LS00N	DIL14	74xx-us	1
IC7	74LS373N	74LS373N	DIL20	74xx-us	1
IC8	DAC0808N	DAC0800N	DIL16	linear	1
JP1		PINHD-2X25	2X25	pinhead	1
JP2		PINHD-1X2	1X02	pinhead	1
R1	4K7	R-EU_0204/7	0204/7	rcl	1
R2	1K	TRIM_EU-B25P	B25P	pot	1
R6	33K	R-EU_0204/7	0204/7	rcl	1
R7	33K	R-EU_0204/7	0204/7	rcl	1
R8	33K	R-EU_0204/7	0204/7	rcl	1
R9	33K	R-EU_0204/7	0204/7	rcl	1
R10	33K	R-EU_0204/7	0204/7	rcl	1
R11	33K	R-EU_0204/7	0204/7	rcl	1
R12	33K	R-EU_0204/7	0204/7	rcl	1
R13	33K	R-EU_0204/7	0204/7	rcl	1
S1		DA08	DA-08	switch-dil	1

3.7.8 Calibration procedures

To calibrate the output, send hex value "ff" to the board, and adjust the output voltage output to 2 V by trimming R2.

3.7.9 Test procedures

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board to the bus system.
4. Run the GUI program and test the board output with a voltmeter and match value with the voltage set in the GUI.
5. Observe the LED array for bus activity.

Conclusion: The board is working as planned.

3.8 Analog Input Board

3.8.1 Purpose

The purpose of this board is to connect an analog input pin on the Z8 to the Analog Input Board terminal through a surge protection.

The Z8 microcontroller performs an ADC operation on the analog voltage and returns the result to the PC.

3.8.2 Photo

This is the Analog Input Board.

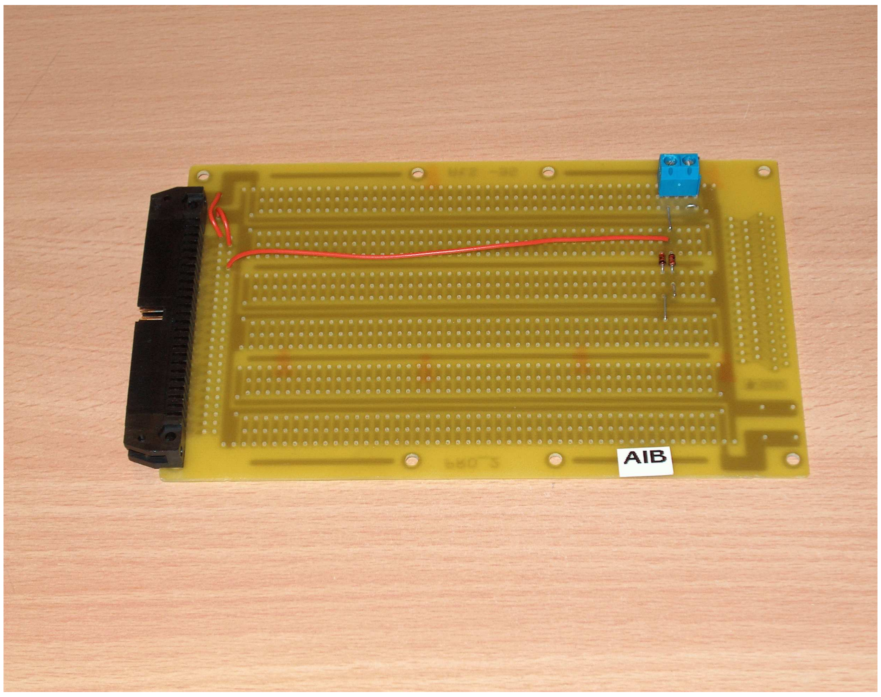


Figure 3.23: Analog Input Board Photo.

3.8.3 Block diagram

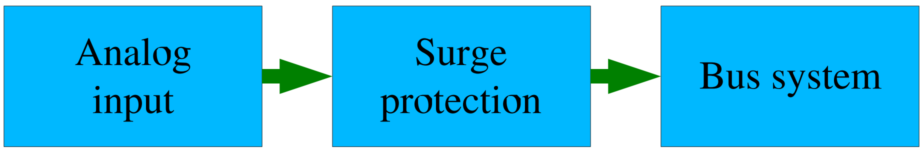


Figure 3.24: Analog Input Board block diagram.

3.8.4 Bus communication

This board does not perform any signal processing.

3.8.5 Calculations

No calculations is made for this board.

3.8.6 Schematics

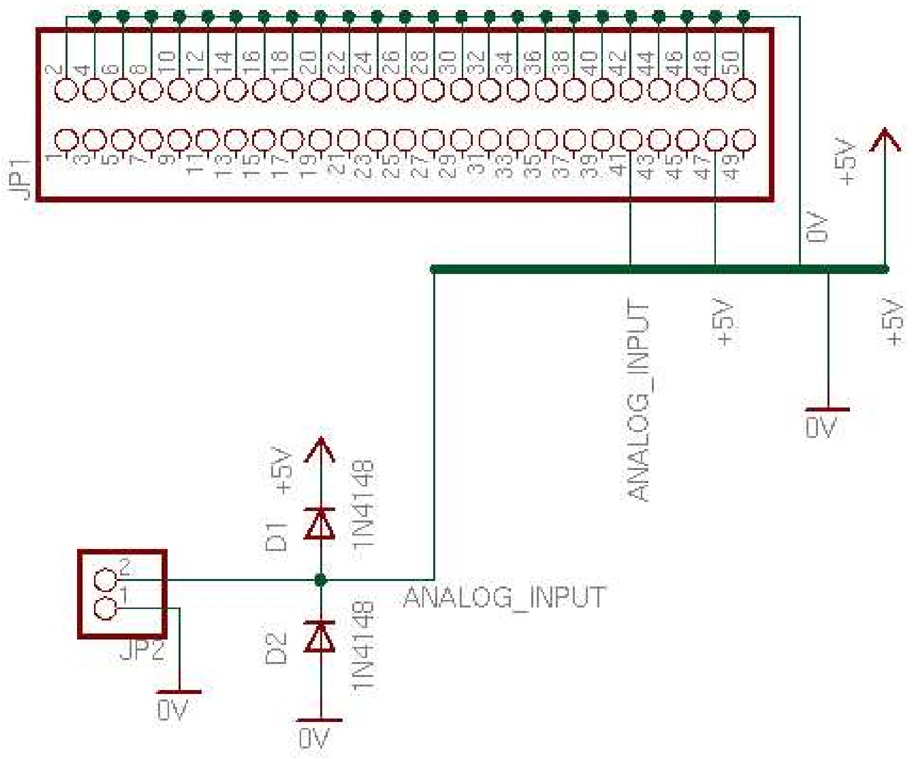


Figure 3.25: Analog Input Board schematic.

3.8.7 Partlist

```
Partlist

Exported from aib.sch at 12/22/2004 14:24:09

EAGLE Version 4.11 Copyright (c) 1988-2003 CadSoft

Part     Value          Device      Package  Library  Sheet

D1       1N4148         1N4148      DO35-10  diode    1
D2       1N4148         1N4148      DO35-10  diode    1
JP1                     PINHD-2X25 2X25     pinhead  1
JP2                     PINHD-1X2  1X02     pinhead  1
```

3.8.8 Test procedures

Procedure:

1. Measure resistance between ground and all the signals and voltage inputs. Resistance should be infinite.
2. Do a visual inspection and look for shorts in the wires and soldering work.
3. Connect the board to the bus system.
4. Run the GUI program and test the board with a voltage on the input and match value with the voltage read in the GUI.

Conclusion: The board is working as planned.

Chapter 4

Software Block/Module description

4.1 Overview

The Rapier software is divided into three parts:

1. GUI on a PC.
2. Embedded code in a microcontroller.
3. Protocol for instructions.

It is important to understand the relationship between the software parts and the hardware parts. Look at this illustration of the relationship.

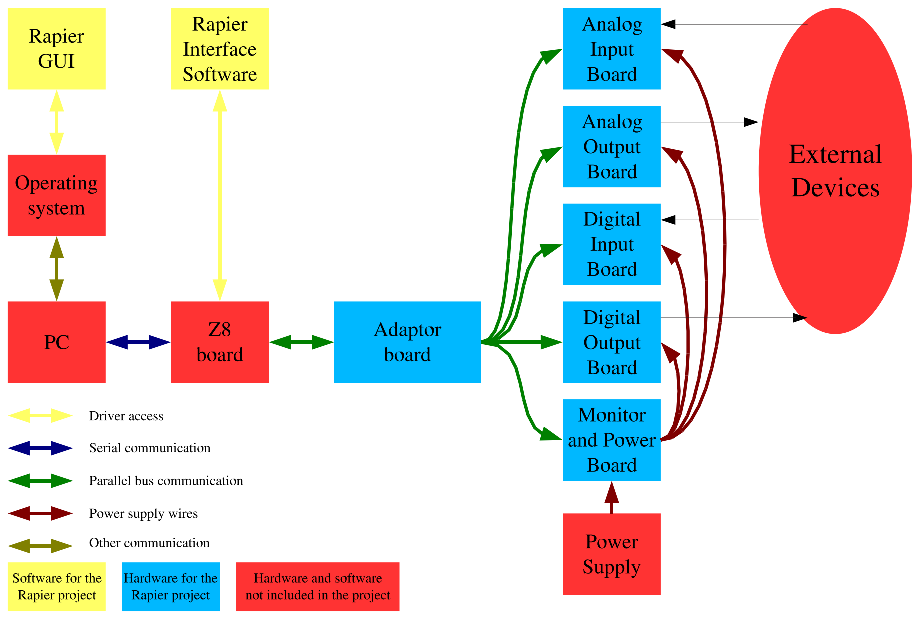


Figure 4.1: Hardware and software block diagram.

Explanation The different colors used in this illustration signifies its type and relation to the project:

Yellow boxes	are software components written for this project.
Red boxes	are software and hardware components used, but not designed or build for this project.
Blue boxes	are hardware components designed and build for this project.
Yellow arrows	are software accessing hardware in a process described in this project.
Brown arrow	are communication not described in this project.
Blue arrow	are serial communication RS232. This protocol is not described in the project.

Green arrows	are parallel communication described in this project.
Red arrows	are power supply to the hardware in this project.

4.1.1 Rapier software communication

The next illustration shows the path between the two parts through which they communicate.

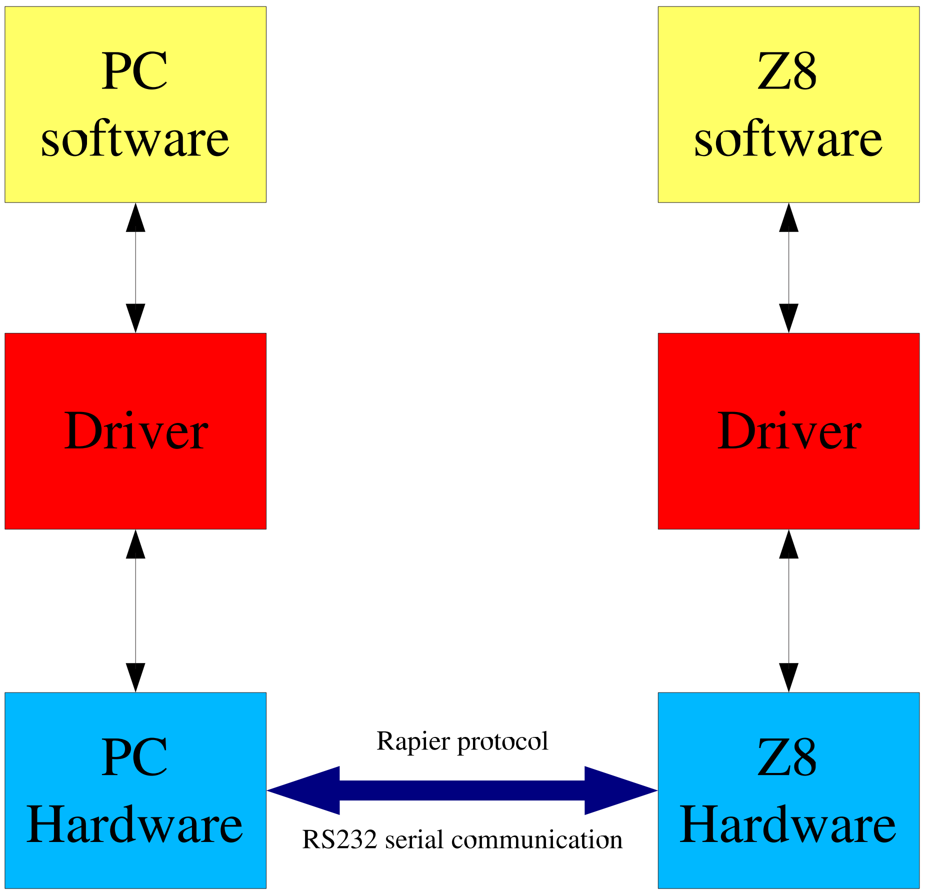


Figure 4.2: Rapier software communication diagram.

4.1.2 Rapier command path

To understand the way instructions flow in the system look at the illustration below:

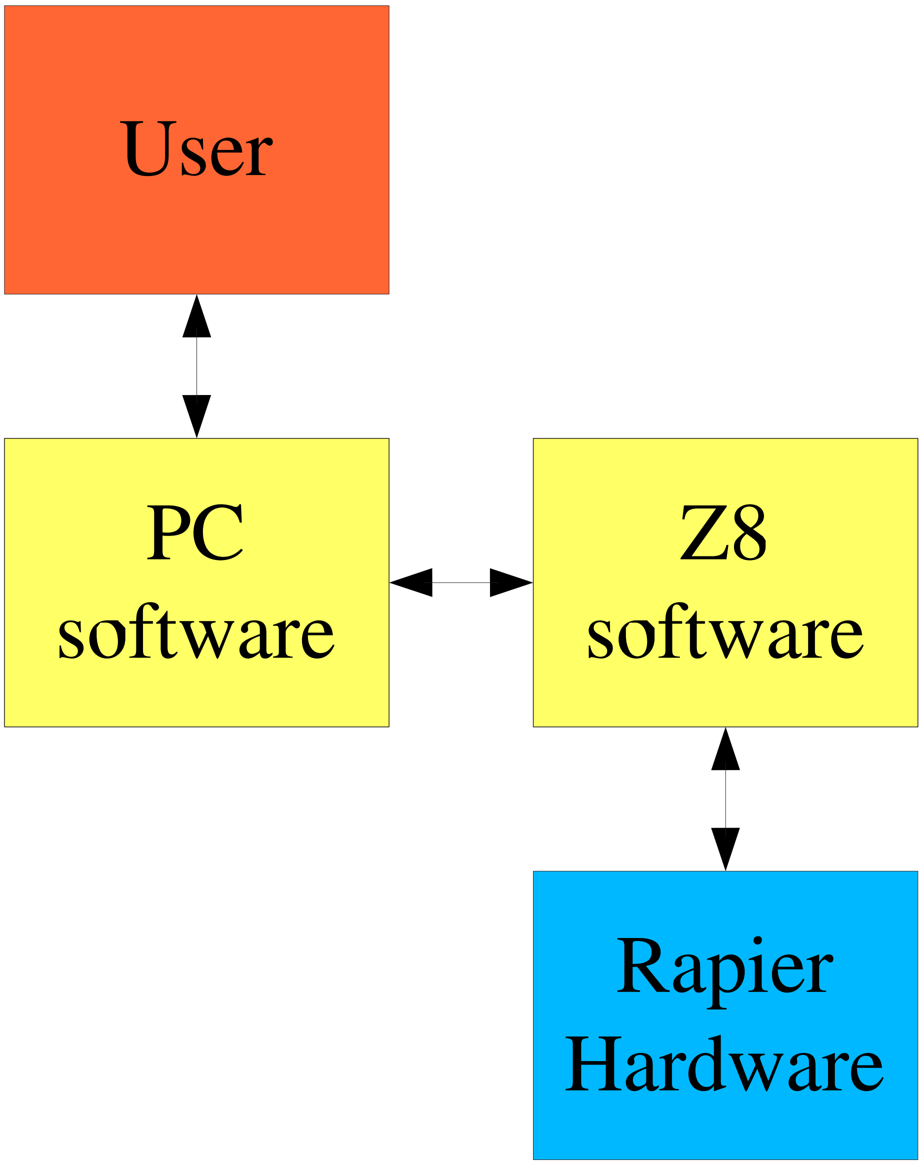


Figure 4.3: Rapier command path diagram.

4.2 Rapier Protocol

[1]

4.2.1 Overview

This protocol is designed to be as simple, yet flexible to use. The protocol meets the following criteria:

1. Easy to distinguish between data and instruction.
2. Easy to emulate the GUI through a terminal program like Hyperterminal in Windows or Minicom in Linux.

With these criteria in mind, it was decided to split data bytes up into 2 nibbles by using hex numbers (0-9 and a-f). The rest of the letters and/or ASCII[2] characters could be used as instructions. But the primary concern was that each instruction and/or data should be reproduced from a terminal application for testing and development purpose.

4.2.2 Data

Data is normally organized in bytes. But since it is next to impossible to enter or display every ASCII character in a terminal program in a way that makes sense, this method of using data in sizes of bytes is not easy applicable. But if the data is split up in nibbles or in hex values, it will make more sense in a terminal program.

The data is then represented as hex numbers 0-9 or a-f. The downside is that a byte has grown to 2 hex numbers in characters which means 2 ASCII characters or 2 bytes. But it make it possible to enter and display in a terminal program in Windows or Linux.

4.2.3 Instructions

The characters left for instructions are the rest of the ASCII table. Only 4 of them are used in the Z8 code. Instructions are only 1 byte long, but could easily be longer if the need should arise in the future. The Z8 code would have to be considerably expanded for that use.

[1]In computing, a protocol is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints. Protocols may be implemented by hardware, software, or a combination of the two.

[2]ASCII (American Standard Code for Information Interchange), generally pronounced 'aski', is a character set and a character encoding based on the Roman alphabet as used in modern English and other Western European languages.

4.2.4 Protocol in use

The protocol in use in this project is shown in the following table:

Protocol in use	
ASCII character	**Description**
0	Value 0
1	Value 1
2	Value 2
3	Value 3
4	Value 4
5	Value 5
6	Value 6
7	Value 7
8	Value 8
9	Value 9
a	Value 10
b	Value 11
c	Value 12
d	Value 13
e	Value 14
f	Value 15
k	AD conversion
s	Addressing operation
i	Input operation
o	Output operation
x	Operation completed

Table 4.1: Protocol in use

4.2.5 Sending instructions or data

Instruction and/or data are combined in the following ways:

k	= is a stand alone instruction.
s	= is followed by an address in hex format.
i	= returns the result in hex format.
o	= is followed by a data byte in hex format.

All replies from the Z8 is followed by a "x" and a \n(newline) as an "end of message" character to tell the PC that this was the end of the transmission. The \n is appended in order for the terminal program to understand that a transmission is ended. This \n is easily disregarded in the GetInput() in the GUI software.

If the Z8 does not receive one of the 4 characters, it just restarts the "while loop" and sends \n back to the PC.

4.2.5.1 Example 1

In this example the user clicks on the "Update Input Voltage" on the GUI window.

- The GUI sends a "k" to the Z8.
- The Z8 performs an AD conversion and prepares the result. For example the hex value "4a".
- The Z8 appends a "x" and a \n to the result.
- The Z8 sends the result: "4ax\n"
- The GUI evaluates and calculates the result and display it in the GUI window.
- The GUI appends the result string in the communication log in the GUI window.

4.2.5.2 Example 2

In this example the user enters a new value in the "Output Voltage" on the GUI window. Let the new voltage has the hex value "c2" and the board id is 02.

- The GUI sends fx. a "s02oc2" to the Z8.
- The Z8 decodes the address in "s02" and sends "x\n" in reply.
- The Z8 decodes the data to the output board in "0c2" and sends "x\n" in reply.
- The Z8 outputs the data "c2" to the Analog Output Board.
- The GUI appends the "x" received twice in the communication log in the GUI window.

4.2.5.3 Example 3

In this example the user clicks on the "Update Digital Input" on the GUI window.Let the board id be 04 and the result be 81.

- The GUI sends fx. a "s04i" to the Z8.
- The Z8 decodes the address in "s04" and sends "x\n" in reply.
- The Z8 decodes the instruction and fetch the data from the input board and sends "81x\n" in reply.

- The GUI evaluates the result and display it in the GUI window with the first and last radiobutton set according the value 81 in binary: 10000001.
- The GUI appends the received string "81x" in the communication log in the GUI window.

4.3 GUI software

[3]

The GUI is written in C++ and uses the QT graphical library. This gives a window which is looking like all other windows in Linux running the KDE window manager.

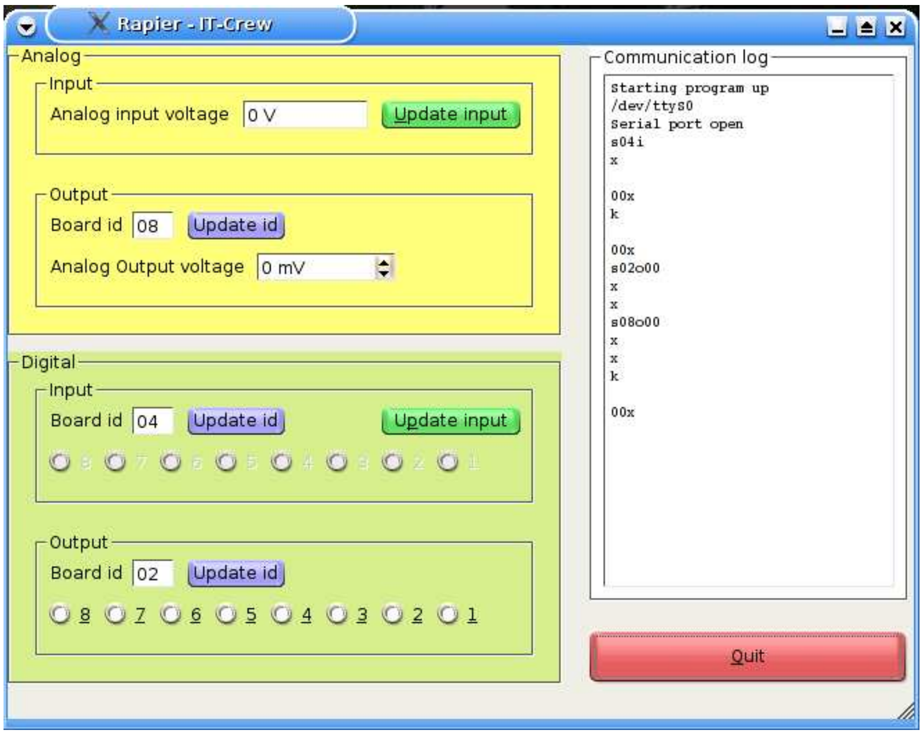
To fully understand the GUI software, it is important to know that the GUI is event-driven based on user input. It does nothing on its own. Every time a graphical widget is clicked or interacted with, it emits a "signal". These signals are connected to "slots"[4]. A slot is a kind of functions. A widget can generate more than one signal, and many widgets can connect to a single slot.

The way a GUI is developed in C++ and QT, is by first drawing the window with all its widgets(ie buttons, text boxes, spinboxes etc), and later fill in the C++ code to make the slots do something.

This is an overview of the GUI:

[3] A graphical user interface (or GUI, pronounced "gooey") is a method of interacting with a computer through a metaphor of direct manipulation of graphical images and widgets in addition to text.

[4] Signals and slots are used for communication between objects. The signal/slot mechanism is a central feature of Qt and probably the part that differs most from other toolkits.



```
Starting program up
/dev/ttyS0
Serial port open
s04i
x

00x
k

00x
s02o00
x
x
s08o00
x
x
k

00x
```

Figure 4.4: Rapier GUI.

For a detailed description of how to use the GUI or understand its layout, please read the users manual starting on page 113.

4.3.1 Main

Nothing exists without a main function. From "main" is the graphical part started up.

4.3.2 Slots

There are 11 slots in the GUI, 8 of them receiving signals from several widgets in the GUI.

The slots can be divided into the following groups:

- Slots responsible for serial port handling.
- Slots responsible for interaction with the user, sending instructions to the Rapier hardware and receiving/interpretate the results and present it in the GUI.

4.3.2.1 Port handling

The serial port on a Linux computer is handled allmost the same way as a file. Everything in a Linux/Unix computer system is files. Even directories are files. All hardware devices are files, the kernel loaded in the RAM is a file. This means that it's the same method you use to access everything in the computer, but with small modifications. When you open a serial port, typically called /dev/ttyS0, you also set up speed and other properties related to serial communication. That part is hard coded into the GUI.

The 4 slots/functions responsible for handling the serial port are:

1. OpenPort()
2. Closeport()
3. WritePort()
4. ReadPort()

4.3.2.2 User interface

The slots responsible for interaction with the user and responsible for sending instructions conforming to protocol:

1. GetInput()
2. UpdateAnalogInput()
3. UpdateAnalogOutput()
4. UpdateDigitalInput()
5. UpdateDigitalOutput()

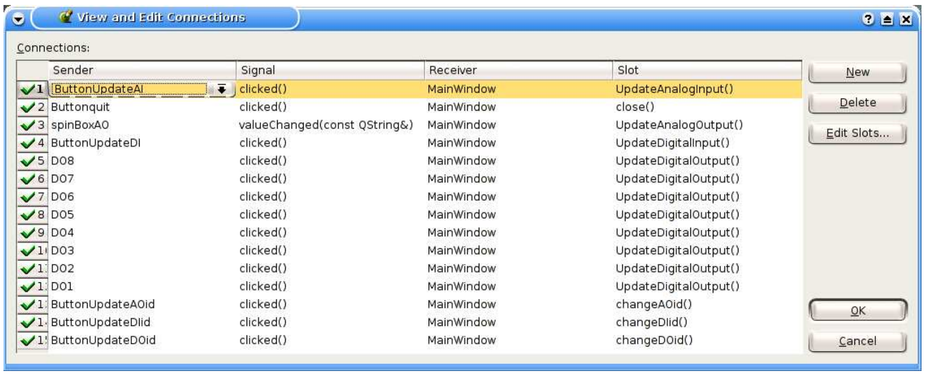
4.3.2.3 Operating system specific functions

These slots are essential to the operating system.

1. init()
2. destroy()

4.3.2.4 Connections

The relationship between widgets sending signals to various slots/functions is shown below.



Connections:

	Sender	Signal	Receiver	Slot
✓ 1	ButtonUpdateAI	clicked()	MainWindow	UpdateAnalogInput()
✓ 2	Buttonquit	clicked()	MainWindow	close()
✓ 3	spinBoxAO	valueChanged(const QString&)	MainWindow	UpdateAnalogOutput()
✓ 4	ButtonUpdateDI	clicked()	MainWindow	UpdateDigitalInput()
✓ 5	DO8	clicked()	MainWindow	UpdateDigitalOutput()
✓ 6	DO7	clicked()	MainWindow	UpdateDigitalOutput()
✓ 7	DO6	clicked()	MainWindow	UpdateDigitalOutput()
✓ 8	DO5	clicked()	MainWindow	UpdateDigitalOutput()
✓ 9	DO4	clicked()	MainWindow	UpdateDigitalOutput()
✓ 1	DO3	clicked()	MainWindow	UpdateDigitalOutput()
✓ 1	DO2	clicked()	MainWindow	UpdateDigitalOutput()
✓ 1	DO1	clicked()	MainWindow	UpdateDigitalOutput()
✓ 1	ButtonUpdateAOid	clicked()	MainWindow	changeAOid()
✓ 1	ButtonUpdateDIid	clicked()	MainWindow	changeDIid()
✓ 1	ButtonUpdateDOid	clicked()	MainWindow	changeDOid()

Figure 4.5: Rapier GUI connections.

4.3.3 Examples of use

4.3.3.1 Updating a board id

If the user wants to update a board id, the user enters the new board id into the relevant field and clicks on "update board id". For example:

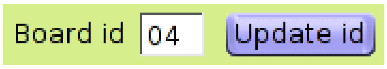


Figure 4.6: Updating board id.

The GUI program reacts to the change by doing this:

1. When the button is clicked it emits a signal to a slot.
2. The slot, which is a function, executes by saving the new board id in a string format in a global variable for later use.

4.3.3.2 Changing the output voltage

If the user wants to update the analog output board voltage, the user either enters a voltage in numbers or clicks on the up arrow or down arrow on the spinbox. For example:

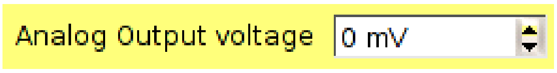


Figure 4.7: Updating output voltage.

The GUI program reacts to the change by doing this:

1. On change of value, the widget emits a signal to UpdateAnalogOutput().
2. This slot/function builds up a string based on board id and the value in the changed widget.
3. UpdateAnalogOutput() sends the string by calling GetInput().

4.3.3.3 Updating the input voltage

If the user wants to update the analog input board voltage, the user clicks update input button. For example:

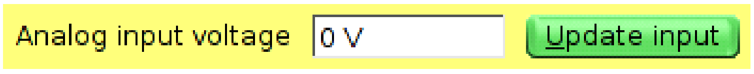


Figure 4.8: Updating input voltage.

The GUI program reacts to the click by doing this:

1. The widget emits a signal to the UpdateAnalogInput() slot.
2. UpdateAnalogInput() sends a string to the Z8 hardware by calling Get-Input().
3. GetInput returns the result from the Z8 hardware.
4. UpdateAnalogInput() converts the returned string into a float.
5. UpdateAnalogInput() presents the new value in the GUI (the picture above).

4.4 Z8 software

4.4.1 Overview

The Z8 code is much simpler than the GUI code. The code performs a simple task of listening to the serial port, evaluate the serial inputs, sending instructions to the Rapier hardware, and returning the result to the PC.

This illustration describes the flow of the Z8 code:

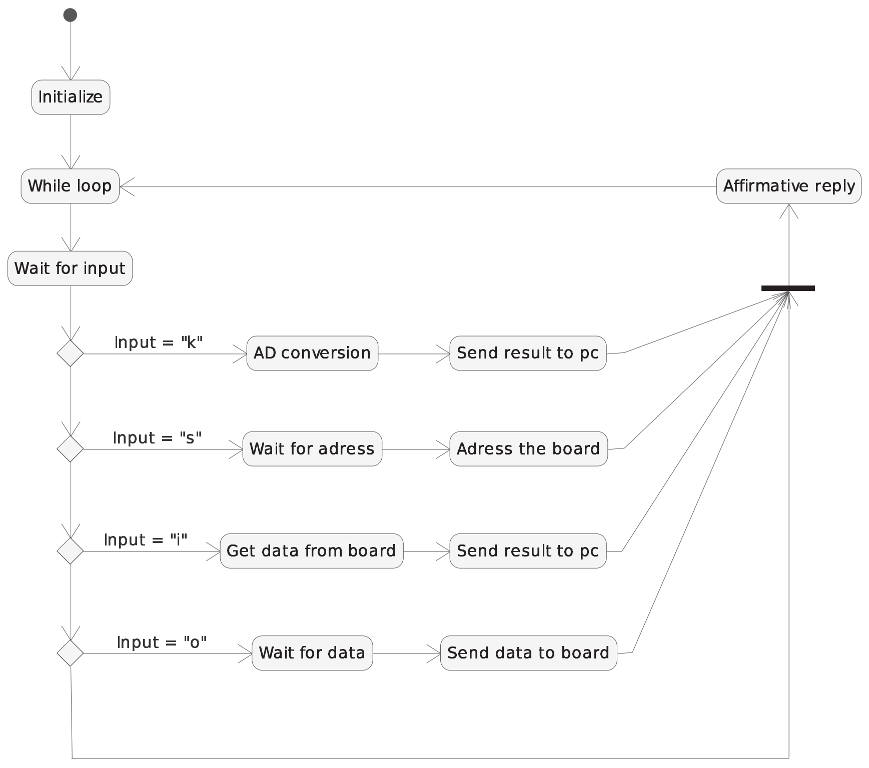


Figure 4.9: Z8 code overview.

This flow diagram describes the main.c, but this will be described in more details in another chapter.

The code is separated into 2 parts:

main.c which takes care of the overall purpose of the Z8 code.

driver.c which takes care of communication with the Rapier hardware.

4.4.2 main.c

main.c is the main function responsible for the overall functionality of the Z8. It runs in an endless while-loop. In each loop the main() listens for input on the serial port and performs different tasks depending of the received characters.

The main function can 4 different actions:

1. Receiving a "k", the Z8 performs an AD conversion and sends the result to the PC.
2. Receiving a "s", the Z8 waits for 2 bytes long address and performs an addressing operation.
3. Receiving a "i", the Z8 performs an input operation from a board and sends the result to the PC.
4. Receiving a "o", the Z8 waits for 2 bytes long data and performs an output operation to a board.

If the Z8 receives a character other than the 4 mentioned above, it exits its current operation and restarts the while-loop.

4.4.3 driver.c

Driver.c is the file with all low level functions concerning the functionality of the Rapier Hardware.

The functions are grouped into categories:

- Low level functions.
- Mid level functions.
- High level functions.

The three levels are described on the next pages.

4.4.3.1 Low level functions

These functions are directly responsible for ADC operations in the Z8, putting data on the Z8 port, which include the AD bus and the Control bus.

Low level functions	
Function	Description
void adbusoutput(char c)	Sets the AD bus with a byte value.
void adress_enable_on( )	Sets the Address Enable high.
void adress_enable_off( )	Sets the Address Enable low.
void adress_decode_on( )	Sets the Address Decode high.
void adress_decode_off( )	Sets the Address Decode low.
void output_function_enable_on( )	Sets the Output Function Enable high.
void output_function_enable_off( )	Sets the Output Function Enable low.
void input_function_enable_on( )	Sets the Input Function Enable high.
void input_function_enable_off( )	Sets the Input Function Enable low.
void deselect_board_on( )	Sets the Deselect Board high.
void deselect_board_off( )	Sets the Deselect Board low.
void setadcinputs( )	Sets up the ADC.
void setadbustoinput( )	Sets the AD bus to input mode.
void setadbustooutput( )	Sets the AD bus to output mode.
void bussetup( )	Sets up Z8 ports E and G for use.

Table 4.2: Low level functions

4.4.3.2 Mid level functions

These functions calls the low level functions in organized routines to handle all the instructions described in the Rapier protocol.

The functions are:

Mid level functions	
Function	Description
void adressingboard(char id)	Addresses a board with the id argument.
void byteout(char data)	Sends the data to the selected boards.
char bytein()	Receives a byte from selected board.
char singleadconverting()	Performs a single AD conversion.

Table 4.3: Mid level functions

4.4.3.3 High level functions

The high level functions was only used during delevopment.

High level functions	
Function	Description
void dob(char id, char data)	Performs a full addressing and output to a Digital Output Board.
char dib(char id)	Performs a full addressing and input to a Digital Input Board.
void aob(char id, char data)	Performs a full addressing and output to a Analog Output Board. This function is basicelly the same as the dob() function but the name was kept for name sake.
char aib( )	Performs an AD conversion like the mid level function singleadconverting( ), but is kept for future use when more analog inputs is added.

Table 4.4: High level functions

For a more detailed description of the functions, look into the appendix on page 129

4.5 Project file management

For this project, there was a need for a tool to manage all the files with schematics, diagrams, photos, latex files.

The program requires Bash[5] and dialog[6].

This kind of user interface was chosen because for this kind of system administrative jobs, a graphical interface is not needed. A text interface/command line interface is often faster to work with than working with a mouse. A program based on a script rather than a compiled executable program is more flexible when it comes to day to day editing and addition of new functionality.

The design philosophy was to automate as many tasks as possible since many of the tasks takes a lot of time. For daily use, the program is started and left alone for about 20 minutes and after that everything is updated in the documentation and the homepage.

4.5.1 Example 1 - a schematic is updated

If a schematic is updated, the user saves the schematic as a bitmap pictures in EPS format and updates the partlist. The management program copies the partlist into the documentation and converts the bitmap picture into EPS for the documentation and JPG for the homepage and places all the new files in the right places.

When uploading the documentation and homepage, only the files changed is uploaded to save time and network load.

4.5.2 Example 2 - a html file is updated

If a html/php file is updated, the management program uploads only the changed files.

4.5.3 Functionality

On the next page is a short review of the functionality of the program.

[5]Bash is a UNIX command shell written for the GNU project. Its name is an acronym for Bourne-again shell - a pun on the Bourne shell (sh), which was an early, important UNIX shell.

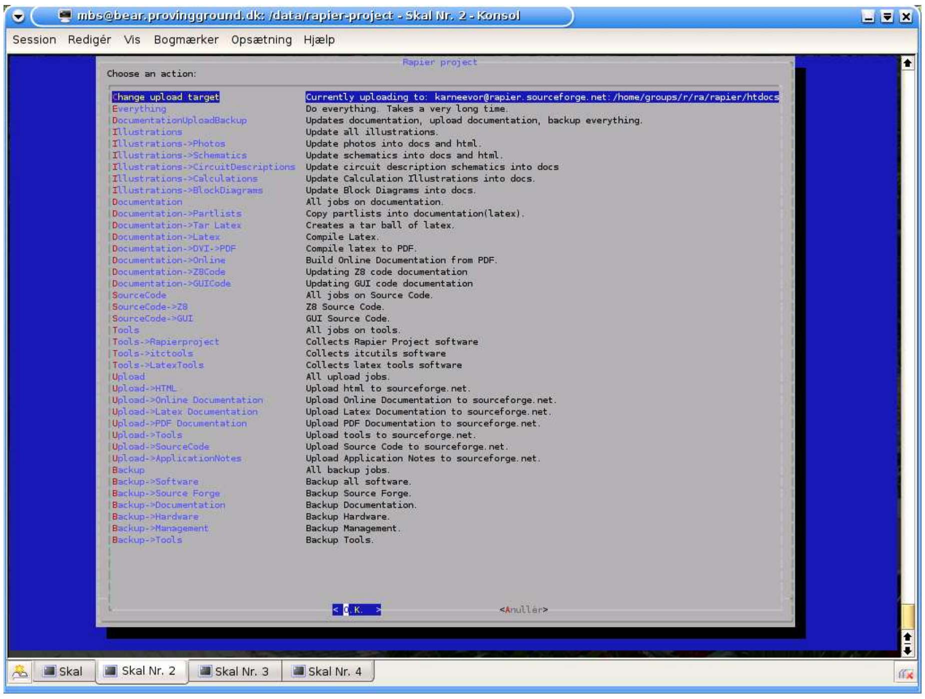
[6]Dialog is a program that will let you to present a variety of questions or display messages using dialog boxes from a shell script.

Rapierproject menu and functions	
Menu item	Description
Change upload target	To either sourceforge.net or local.
Everything	Do everything. Takes a very long time.
DocumentationUploadBackup	Updates documentation, upload documentation, backup everything.
Illustrations	Update all illustrations.
Illustrations->Photos	Update photos into docs and html.
Illustrations->Schematics	Update schematics into docs and html.
Illustrations->CircuitDescriptions	Update circuit description schematics into docs
Illustrations->Calculations	Update Calculation Illustrations into docs.
Illustrations->BlockDiagrams Documentation	Update Block Diagrams into docs. All jobs on documentation.
Documentation->Partlists	Copy partlists into documentation(latex).
Documentation->Tar Latex	Creates a tar ball of latex.
Documentation->Latex	Compile Latex.
Documentation->DVI->PDF	Compile latex to PDF.
Documentation->Online	Build Online Documentation from PDF.
Documentation->Z8Code	Updating Z8 code documentation
Documentation->GUICode	Updating GUI code documentation
SourceCode	All jobs on Source Code.
SourceCode->Z8	Z8 Source Code.
SourceCode->GUI	GUI Source Code.
Tools	All jobs on tools.
Tools->Rapierproject	Collects Rapier Project software
Tools->itctools	Collects itcutils software
Tools->LatexTools	Collects latex tools software
Upload	All upload jobs.
Upload->HTML	Upload html to sourceforge.net.
Upload->Online Documentation	Upload Online Documentation to sourceforge.net.
Upload->Latex Documentation	Upload Latex Documentation to sourceforge.net.
Upload->PDF Documentation	Upload PDF Documentation to sourceforge.net.
Upload->Tools	Upload tools to sourceforge.net.
Upload->SourceCode	Upload Source Code to sourceforge.net.
Upload->ApplicationNotes	Upload Application Notes to sourceforge.net.
Backup	All backup jobs.
Backup->Software	Backup all software.
Backup->Source Forge	Backup Source Forge.
Backup->Documentation	Backup Documentation.
Backup->Hardware	Backup Hardware.
Backup->Management	Backup Management.
Backup->Tools	Backup Tools.

Table 4.5: Ports and busses

4.5.4 The interface

This is the console/text based interface. The program is operated by using the keyboard.


mbs@bear.provingground.dk: /data/rapier-project - Skal Nr. 2 - Konsol
Session Redigér Vis Bogmærker Opsætning Hjælp
Skal
Skal Nr. 2
Skal Nr. 3
Skal Nr. 4

```
                         Rapier project
Choose an action:

Change upload target                 Currently uploading to: karneevor@rapier.sourceforge.net:/home/groups/r/ra/rapier/htdocs
Everything                           Do everything. Takes a very long time.
DocumentationUploadBackup            Updates documentation, upload documentation, backup everything.
Illustrations                        Update all illustrations.
Illustrations->Photos                Update photos into docs and html.
Illustrations->Schematics            Update schematics into docs and html.
Illustrations->CircuitDescriptions   Update circuit description schematics into docs
Illustrations->Calculations          Update Calculation Illustrations into docs.
Illustrations->BlockDiagrams         Update Block Diagrams into docs.
Documentation                        All jobs on documentation.
Documentation->Partlists             Copy partlists into documentation(latex).
Documentation->Tar Latex             Creates a tar ball of latex.
Documentation->Latex                 Compile Latex.
Documentation->DVI->PDF              Compile latex to PDF.
Documentation->Online                Build Online Documentation from PDF.
Documentation->Z8Code                Updating Z8 code documentation
Documentation->GUICode               Updating GUI code documentation
SourceCode                           All jobs on Source Code.
SourceCode->Z8                       Z8 Source Code.
SourceCode->GUI                      GUI Source Code.
Tools                                All jobs on tools.
Tools->Rapierproject                 Collects Rapier Project software
Tools->itctools                      Collects itcutils software
Tools->LatexTools                    Collects latex tools software
Upload                               All upload jobs.
Upload->HTML                         Upload html to sourceforge.net.
Upload->Online Documentation         Upload Online Documentation to sourceforge.net.
Upload->Latex Documentation          Upload Latex Documentation to sourceforge.net.
Upload->PDF Documentation            Upload PDF Documentation to sourceforge.net.
Upload->Tools                        Upload tools to sourceforge.net.
Upload->SourceCode                   Upload Source Code to sourceforge.net.
Upload->ApplicationNotes             Upload Application Notes to sourceforge.net.
Backup                               All backup jobs.
Backup->Software                     Backup all software.
Backup->Source Forge                 Backup Source Forge.
Backup->Documentation                Backup Documentation.
Backup->Hardware                     Backup Hardware.
Backup->Management                   Backup Management.
Backup->Tools                        Backup Tools.

                    < O.K. >                  <Annullér>
```

Figure 4.10: Rapier project file management system.

4.5.5 Source code

Turn to page 156 to se the source code.

Chapter 5

System integration

5.1 Development

5.1.1 Resources of help

Help was found in application notes, in the many advice's from my tutor, on SSLUG [1] email list for Linux programming and countless searches in Google.

5.1.2 Project files management

During the early stages of the development, the need for a project file management system became obvious. For the endless updates of files which should be copied into the documentation and homepage, daily backups and uploads to the web server through scp [2], a program was needed to do those tasks without user intervention, entering password and not uploading unchanged files.

All this led to the programming of the program named rapierproject. But it can easily be adapted to another project for later use. This program has saved a lot of time and worrying of relevant file update. The program was written in bash shell script.[3]

5.1.3 Software tools

Many programs was used in the development of this project:

Z8 Encore is the Integrated Delevopment Environment running in Windows.

Kile for writting this documentation.[4]

[1]Skåne Sjælland Linux User Group
[2]secure copy based on secure shell(ssh)
[3]BASH shell script are a language used in Linux, Unix and Mac OS X.
[4]Kile is a LaTeX 2εfront-end.

Eagle	for drawing all electronic schematics.
QT-designer	for designing the GUI.
GCC	for compiling the GUI code.
Itctools	which was once written. Used for backup and RSA key distribution.
Rapierproject	for managing the project files.
OpenOffice.org	an office suite used for drawing illustrations.
Planner	for project management.
Firefox	browser for searching on Google.
Thunderbird	email client for the SSLUG mailing list.
Umbrello	for UML illustrations.
Countless utilities	for small tasks on my Linux laptop.

5.2 Connecting the hardware

Follow this procedure to connect the hardware:

- Put the Adaptor Board on the Z8 evaluation board.
- Stack up the boards. Preferable with the Monitor and Power board on top.
- Connect the flat cable to the stack of boards.
- Connect the flat cable to the Adaptor Board.
- Connect the serial cable to the Z8 evaluation board and the PC.
- Connect the power cable to the Monitoring and Power Board and the external power supplies.
- Connect the power adaptor to the Z8 evaluation board.
- Turn on power to all devices.
- When the PC has finished booting start up the Rapier GUI.
- Connect the relevant board to relevant input signals generators/simulators, or connect the analog output to the analog input and the 8 digital outputs to the 8 digital inputs.
- The system is ready for use.

The following tests are based on the premise that all the outputs are looped back to the inputs in a sort of self diagnostic mode. Look at the picture below.

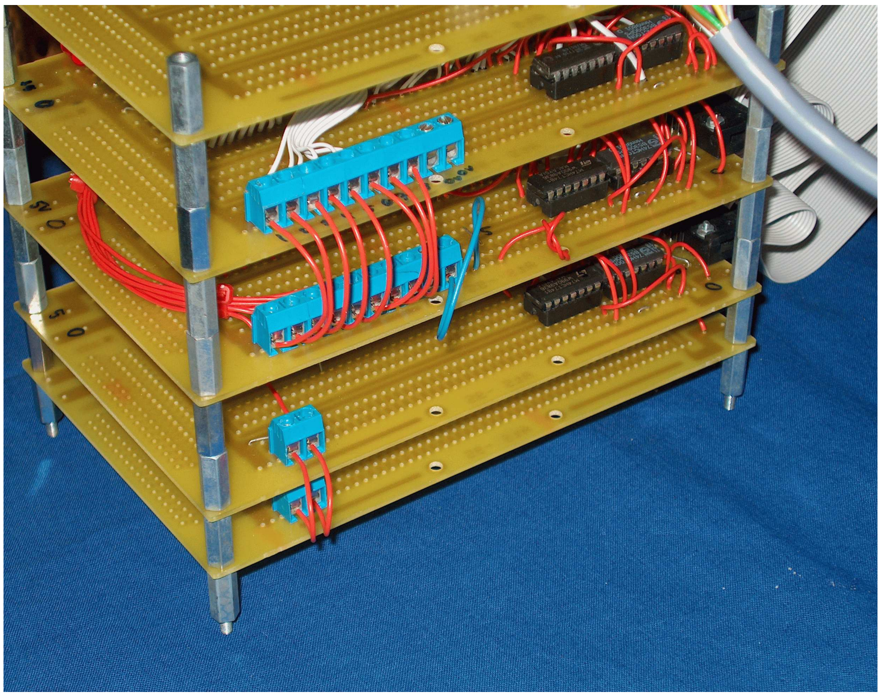


Figure 5.1: Loop back for testing purpose.

5.3 Linearity problems

Through the days of tests and documentation, a problem with linearity was discovered. To explorer this problem, the boards was tested in the range of voltage from 0 V to 2 V in steps of 100 mV. The values tested:

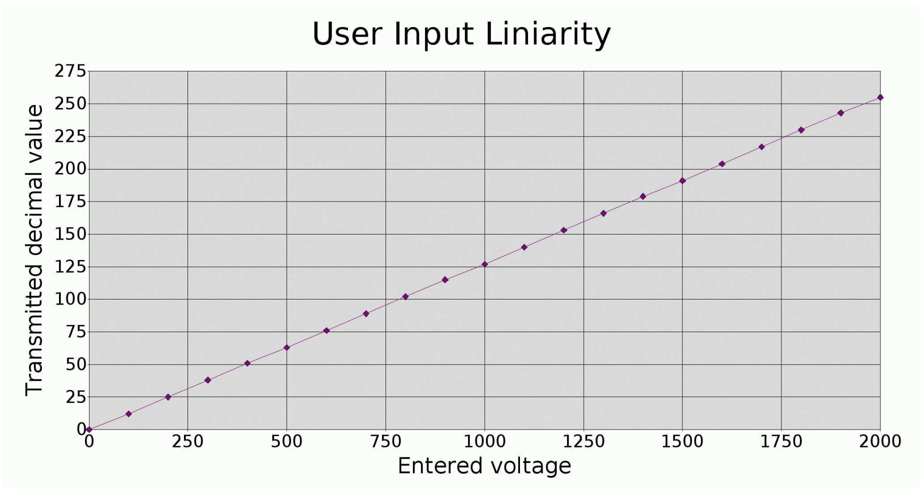


Figure 5.2: Optimal linearity.

This is the optimal linearity.

Output linearity This is the measured output:

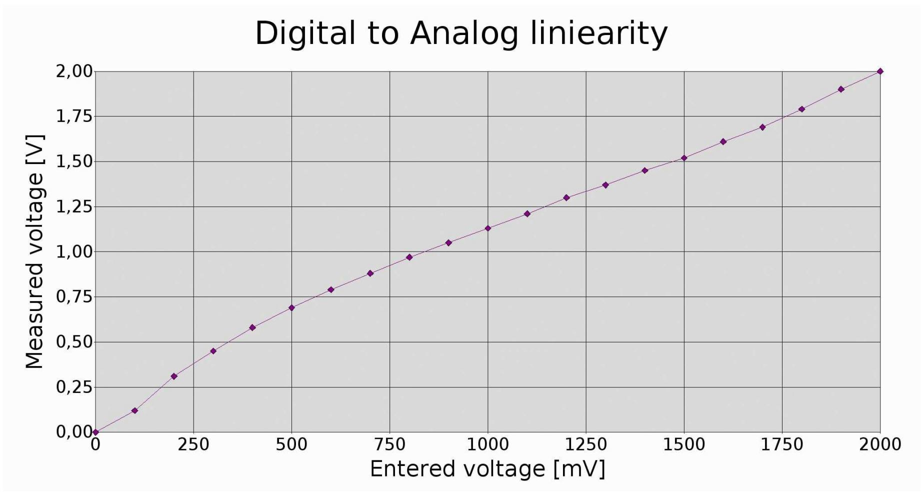


Figure 5.3: Output linearity.

This is not linear.

Output linearity This is the measured input:

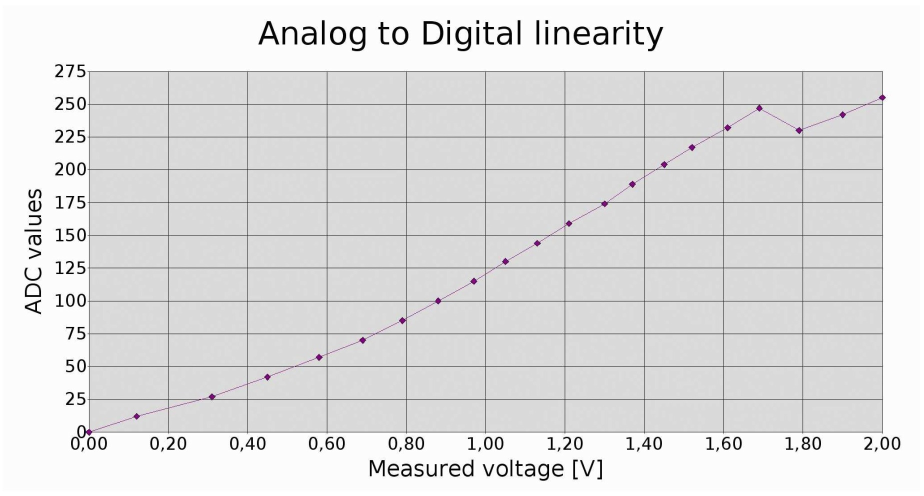


Figure 5.4: Input linearity.

This is not linear.

Output linearity This is the combined linearity:

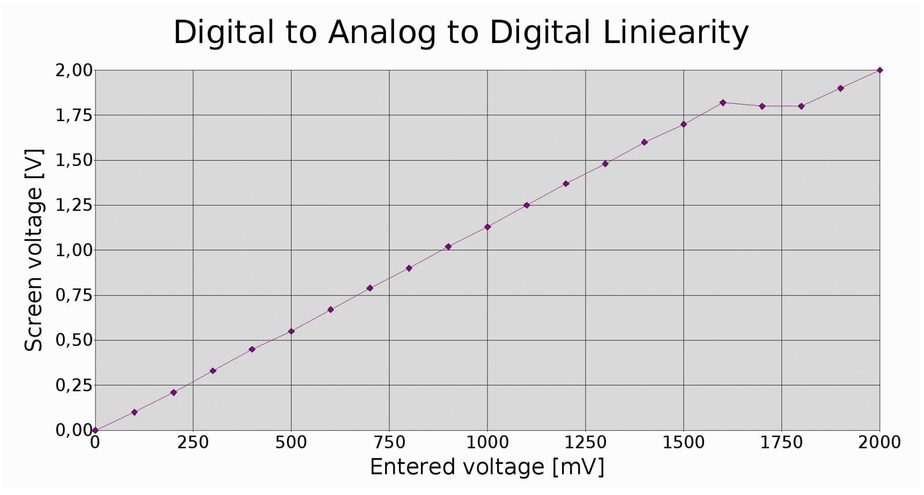


Figure 5.5: Overall linearity.

This is still not linear, but a little better because the 2 non-linearities combined works in opposite direction and works for the better.

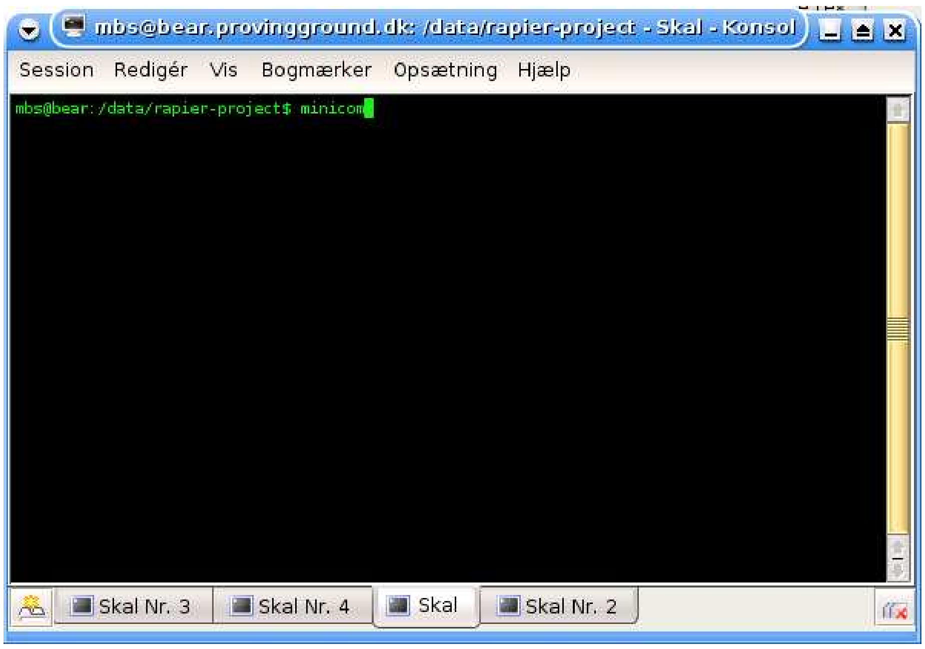
This poor linearity is not explained in the application notes of the components used. But here is a lots of possible reasons:

1. A bad component.
2. Influence from external components and parts.
3. Bad test equipment.

5.4 Tests

5.4.1 Testing the hardware from a terminal program

Easy testing The best and easiest way of testing the hardware, is to connect all the hardware to the PC and run either the GUI or a terminal program. On Linux run the program minicom.[5]

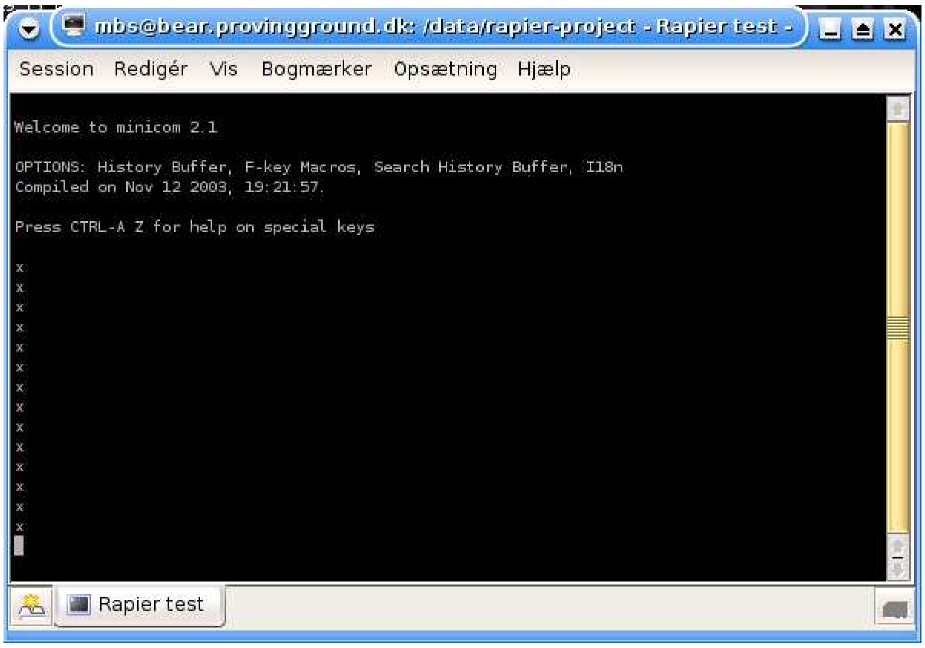


```
mbs@bear:/data/rapier-project$ minicom
```

Figure 5.6: Starting minicom.

Screen output The output will show something like this:

[5]http://alioth.debian.org/projects/minicom/


mbs@bear.provingground.dk: /data/rapier-project - Rapier test -
Session Redigér Vis Bogmærker Opsætning Hjælp
Rapier test

```
Welcome to minicom 2.1

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Nov 12 2003, 19:21:57.

Press CTRL-A Z for help on special keys

x
x
x
x
x
x
x
x
x
x
x
x
x
x
```

Figure 5.7: Minicom after start.

Test procedure If the inputs and output are connected in a loop back mode, perform the following test:

5.4.1.1 Adaptor Board operation test

The Adaptor Board does not perform any tasks or signal processing. Individual testing not possible with the standard software.

5.4.1.2 Monitor and Power Board operation test

The Adaptor Board does not perform any tasks or signal processing. Individual testing not possible with the standard software.

5.4.1.3 Digital Output Board operation test

To test this board do the following:

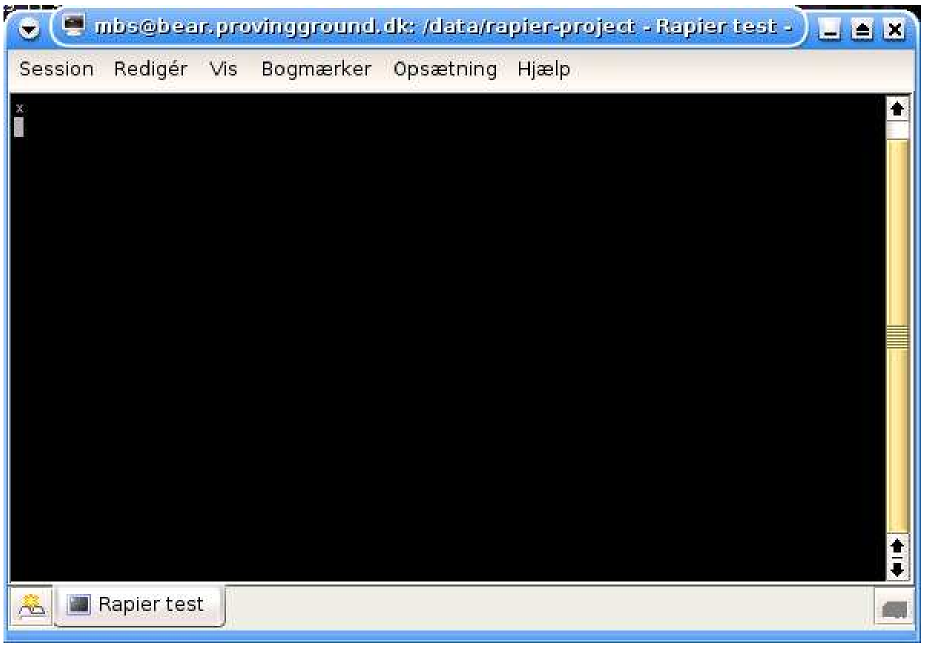
Address the board The board id is hex value 02. To address the board in minicom enter: "s02".

The hardware respond with "x". Not much to look at, but the "x" is important because it's the right response.

Send a byte to the board To write a byte, fx. hex value ff, to the board, send "off".

The hardware respond with "x" again. The "x" is still the right response. But on the Digital Output Board, all the LED[6] is turned on. Other hex values can be written to the board and the LED should turn on corresponding to the binary pattern in the hex value.

The output will show something like this:


mbs@bear.provingground.dk: /data/rapier-project - Rapier test -
Session Redigér Vis Bogmærker Opsætning Hjælp
Rapier test

```
x
```

Figure 5.8: Minicom after addressing a board.

[6]Light Emitting Diode

Partly conclusion The Digital Output Board is working properly as designed.

5.4.1.4 Digital Input Board operation test

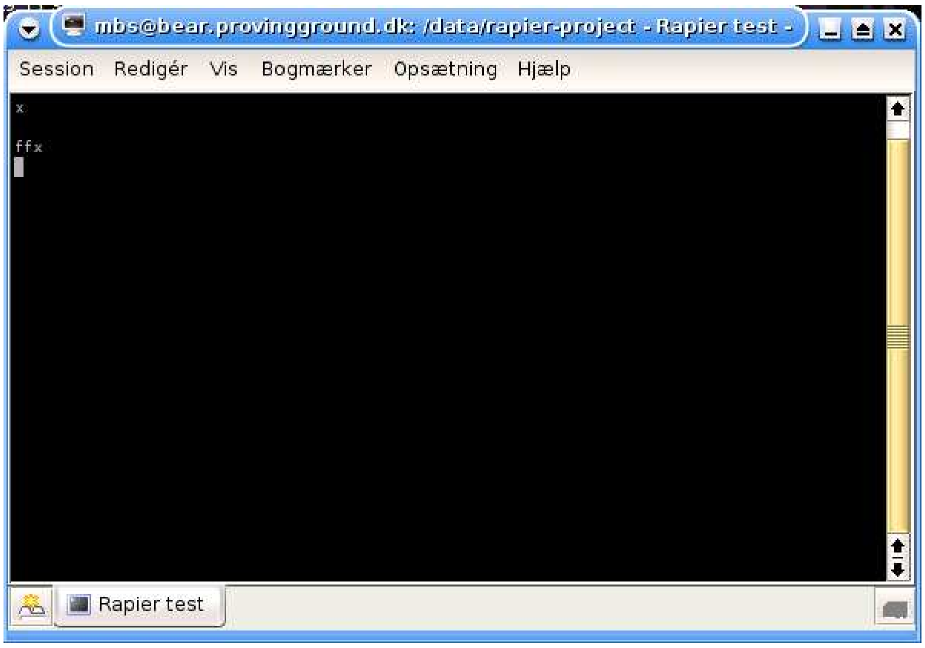
To test this board do the following:

Address the board The board id is hex value 04. To address the board in minicom enter: "s04".

The hardware respond with "x", which is the right response.

Getting the input Send "i" in minicom and see the response:

The output will show something like this:


mbs@bear.provingground.dk: /data/rapier-project - Rapier test -
Session
Redigér
Vis
Bogmærker
Opsætning
Hjælp
Rapier test

```
x

ffx
```

Figure 5.9: minicom after start.

The response is "ffx", which means that "ff" is whats on the input, and the "x" is the end response from the Z8 hardware.

Partly conclusion The input received match the output send earlier. When trying sending other values to the output board, the same values are received on the input board. The Digital Input Board is working properly as designed.

5.4.1.5 Analog Output Board operation test

This test is based on the fact that the board is calibrated properly according to section 3.7.8 on page 62.

Test equipment To test the output connect an volt meter to the output. Expect to measure a voltage of 2 V when sending the value to the board. It should look something like this:

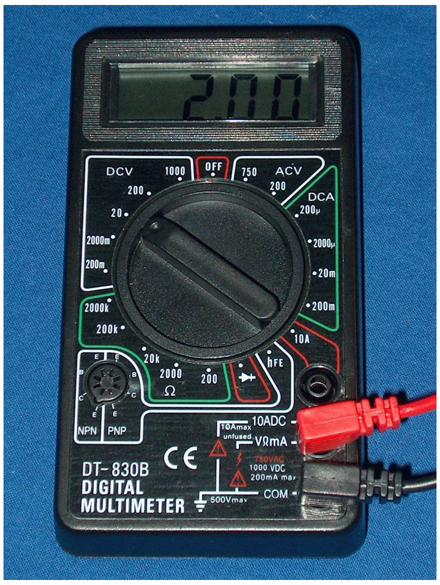


Figure 5.10: Analog output.

Addressing the board The board id is hex value 08. To address the board in minicom enter: "s08".

The hardware respond with "x" which is the right response.

Sending the output To write a byte, fx. hex value ff, to the board, send "off".

The hardware respond with "x" and sets the output voltage to 2 v.

Partly conclusion The Analog Output Board is working properly.

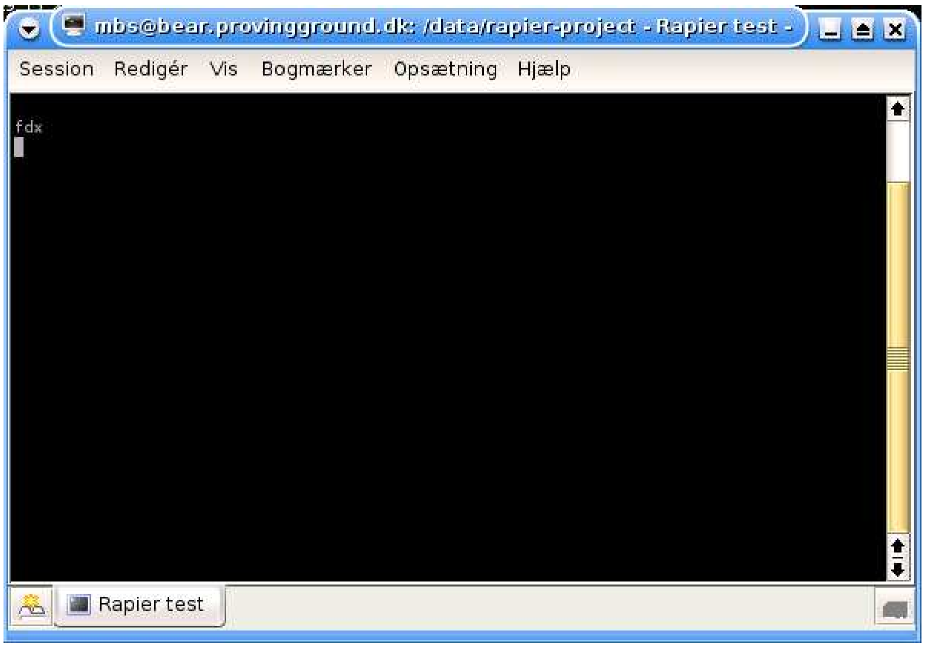
5.4.1.6 Analog Input Board operation test

To test this board do the following:

Address the board The board does not have an id because all functionality is moved inside the Z8 microcontroller.

AD conversion Send "k" in minicom and see the response:

The output will show something like this:


mbs@bear.provingground.dk: /data/rapier-project - Rapier test -
Session Redigér Vis Bogmærker Opsætning Hjælp
Rapier test

```
fdx
```

Figure 5.11: Performing AD conversion.

The response is "fdx". The response may vary because of the analog nature of the function. There can be noise in the moment of AD conversion, the linearity of the ADC[7].

Partly conclusion The input received match the output send earlier. When trying sending other values to the output board, the same values are received on the input board. The Digital Input Board and the Z8 ADC circuitry is working.

[7]Analog to Digital Converter

5.4.2 Testing the hardware from the GUI

Easy testing For this test of the hardware, connect all the hardware to the PC and run the GUI or a terminal program. For a more detailed description of using the GUI, goto the Users Manual starting on page 113.

Test procedure If the inputs and output are connected in a loop back mode, perform the following test:

5.4.2.1 Adaptor Board operation test

The Adaptor Board does not perform any tasks or signal processing. Individual testing not possible with the standard software.

5.4.2.2 Monitor and Power Board operation test

The Adaptor Board does not perform any tasks or signal processing. Individual testing not possible with the standard software.

5.4.2.3 Digital Output Board operation test

To test this board do the following:

Click on one of the 8 radiobuttons in the GUI to activate the output.

The communication log should show the same response as minicom. Observe the output LED's and match them turning on or off matching the input in the GUI.

Partly conclusion The Digital Output Board and GUI is working properly.

5.4.2.4 Digital Input Board operation test

To test this board do the following:

After setting the output on the Digital Output Board, match the input and output in GUI.

Partly conclusion The input match the output. The Digital Input Board and GUI is working properly.

5.4.2.5 Analog Output Board operation test

This test is based on the fact that the board is calibrated properly according to section 3.7.8 on page 62.

Test equipment To test the output connect an volt meter to the output. Expect to measure a voltage of 2 V when sending the value to the board. It should look something like this:

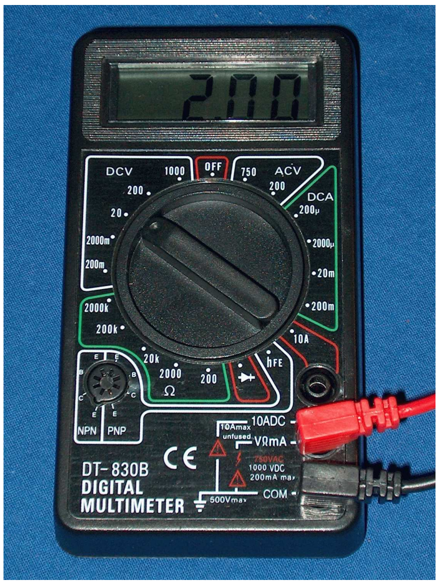


Figure 5.12: Analog output.

Set the voltage output to max value and compare the measured value. The value set and the value measured should be the same.

Partly conclusion The Analog Output Board is working properly.

5.4.2.6 Analog Input Board operation test

Procedure To test this board do the following:

AD conversion Click in the GUI to update analog input.

Analyze result Compare the input voltage with the output voltage. They should match each other or be relatively close. Please read the part on linearity problems on page 93.

Partly conclusion The input received match the output send earlier. When trying sending other values to the output board, the same values are received on the input board. The GUI, the Digital Input Board and the Z8 ADC circuitry is working.

5.5 Test results

Conclusions	
test	test
All Boards using 5V, -5V, 12V or 12V regulated power supply.	Achieved.
Operating temperature: in door temperature.	Achieved.
The Z8 evaluation board	Passing criteria.
Adaptor Board	Conducts signals as designed.
Bus system	Passing criteria and specifications.
Monitoring and Power Board	Passing criteria and specifications.
Digital Input Board	Passing criteria and specifications.
Digital Output Board	Passing criteria and specifications.
Analog Input Board	Passing criteria and specifications.
Analog Input Board	Passing criteria and specifications.
Analog Output Board	Passing criteria and specifications.
Power Supply	Passing criteria and specifications.
Communication protocol	Functionality as required.
GUI	Passing criteria and specifications.
Serial Communication	At no time giving instability or extensive delays.
3rd. party software	Installed on the development PC.

Table 5.1: Conclusions

Chapter 6

Conclusion

6.1 Historical review

This project was developed in 2004. The early start was just before summer vacation when the first test of serial communication was conducted. But the project really took off in the fall 2004 after finishing 3 large assignments in school.

The project was developed in these steps:

1. Research of design solutions in several hardware components.
2. Design of hardware.
3. Assembling the hardware and concurrently developing drivers and serial communication code on Z8.
4. Development of the GUI.

The most difficult part was the GUI and understanding the concept of QT programming. All hardware worked the first time, and no problems of timing and noise was encountered.

The only part that needed redesign was the Analog Input Board. The original solution was based on the same addressing design like the other boards, combined with a Sample and Hold and a reed relay. When the hardware was in a continual AD conversion mode, the reed relay would switch off and on so fast that it produced a high frequency noise.

Milestones in the project

Research of hardware design Finished on the 10th of November, 2004.

ardware assembly and software drivers	Finished on the 13th of December, 2004.
GUI	Finished on the 22nd of December, 2004.
Documentation	Finished on the 3rd of January 2005.

The project would have been finished long before it was if several parts was left out of the project. A lot of ideas was left out:

- LCD driver.
- DDS functionality[1].
- WEB based GUI.
- All serial communication stored in a database for statistical analysis.

Some of those ideas are going to be implemented in the near future after the project is officially finished.

[1]Direct Digital Synthesis is a method to digitally create arbitrary waveforms and frequencies from a single source fixed frequency.

6.2 Final conclusion

This project achieved its goal of designing, building and testing an interface system for a PC running Linux according to specifications within the time table of the project.

Through the project, knowledge of the subject was acquired:

- Design of a large project.
- Extensive research into the many parts of the subject.
- Serial communication between a Linux PC and the Z8 evaluation board.
- GUI programming.
- Planning the project in a time table.

Randers, on the 3rd of January 2005.

Michael Bernhard Sørensen

Chapter 7

Appendix

7.1 Users manual

7.1.1 Prerequisite

7.1.1.1 Hardware

To use the hardware there must be a serial port / COM port. The port must be assigned as /dev/ttyS0 on a working Linux computer. It must support a speed at at least 57600 baud. Typically the speed is 115200 baud on the most computers today.

7.1.1.2 Software

The computer must have the following software installed:

- A working Linux installation or booted from a live CD with graphical user interface enabled.
- KDE desktop installed.
- KDE desktop running or another window manager which supports KDE-applications.

7.1.1.3 User privilege requirements

[1] The user of the GUI must have "write rights" to the /dev/ttyS0 device.

[1] Users must have adequate privilege in the Linuc file system

7.1.2 Installation procedure

7.1.2.1 Installing the software

The GUI can be executed from any location within the file system. But it would be a good idea to place it in the search path, or making a "sym link"[2] to the executable file.

This installation procedure could be used:

1. Copy the executable file "rapier-gui" into directory which are "read only" from the users point of view. For example in /usr/bin/.
2. Run the executable from a shell or from an icon on the Desktop.

Or:

1. Copy the executable file "rapier-gui" into directory which has "read only" from the users point of view. For example in /opt/rapier/. You may need to create the directory yourself.
2. Create a link: `ln /opt/rapier/rapier-gui /usr/bin/rapier`
3. Run the executable `rapier` from a shell or from an icon on the Desktop.

7.1.2.2 Installing the hardware

To install the Rapier hardware follow these instructions:

1. Put the Adaptor Board on the Z8 evaluation board.
2. Connect all the boards to the flat cable.
3. Connect the flat cable to the Adaptor Board.
4. Connect the Z8 to the PC with a serial cable.
5. Connect the Power Cable to the 2 external power supplies. Watch for BLACK and WHITE connector to the right power supply.
6. Connect the Z8 to its own power supply.
7. Turn on power for all units.

Proceed with the software installation.

Setting up for standard use

[2]A new name used instead of the original name of a file.

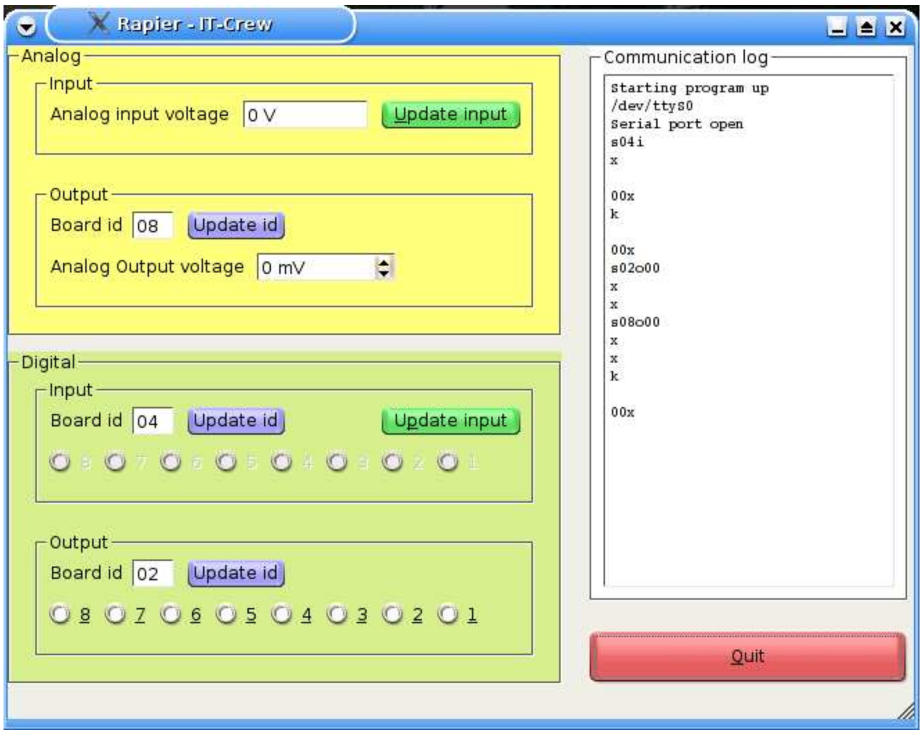
To ensure the right functionality it is important to follow these steps:

1. Look at the boards and note the ID set on the ID switches.
2. Start up the Rapier GUI application.
3. If the board Id's do not match those you've noted in the first step, please reset them according to ID switches.
4. The Rapier system is ready for use.

7.1.2.3 Standard operating procedure

Overview

This is the main window of the GUI.



```
Starting program up
/dev/ttyS0
Serial port open
s04i
x

00x
k

00x
s02o00
x
x
s08o00
x
x
k

00x
```

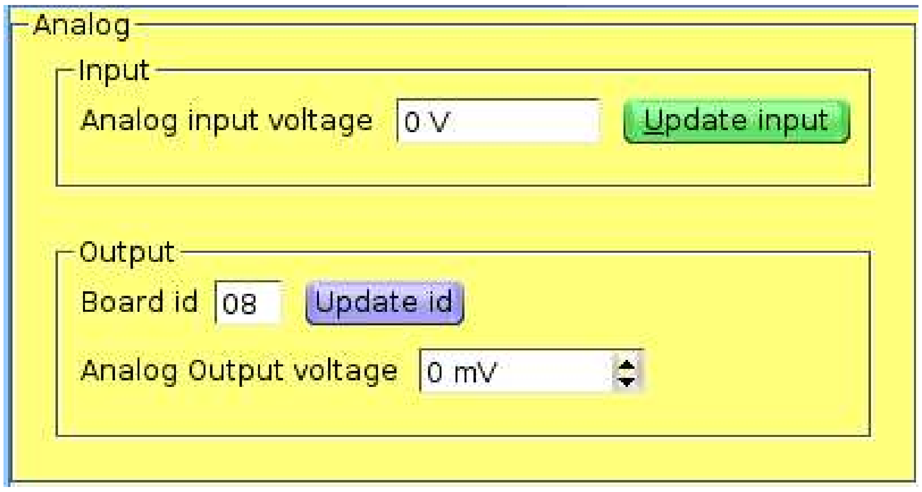
Figure 7.1: Rapier GUI.

It is divided into 4 parts:

- Analog part
- Digital part
- Communication log
- Quit button

The 4 parts are described on the following pages.

7.1.2.4 Analog part



Analog input voltage: 0 V
Board id: 08
Analog Output voltage: 0 mV

Figure 7.2: Rapier GUI analog input and output.

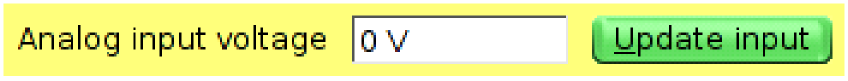
The Analog part is divided into Input and Output.

Input

Input voltage

By clickingthe update button, the analog input voltage on the Rapier hardware is displayed in the field.

This input field is "read only"[3].



Analog input voltage: 0 V

Figure 7.3: Rapier GUI analog input voltage.

[3]"read only" means that the field cannot be edited.

Output

Board id Enter a hex number in this field to set the id matching the board.

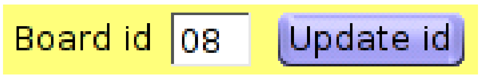


Figure 7.4: Rapier GUI analog output board id.

Output voltage The analog output voltage are set by clicking on the spin box's up and down arrows left of the desired voltage output

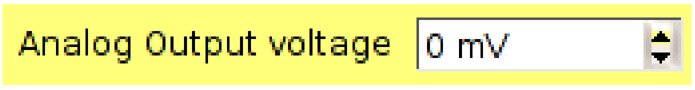


Figure 7.5: Rapier GUI analog output voltage.

7.1.2.5 Digital part

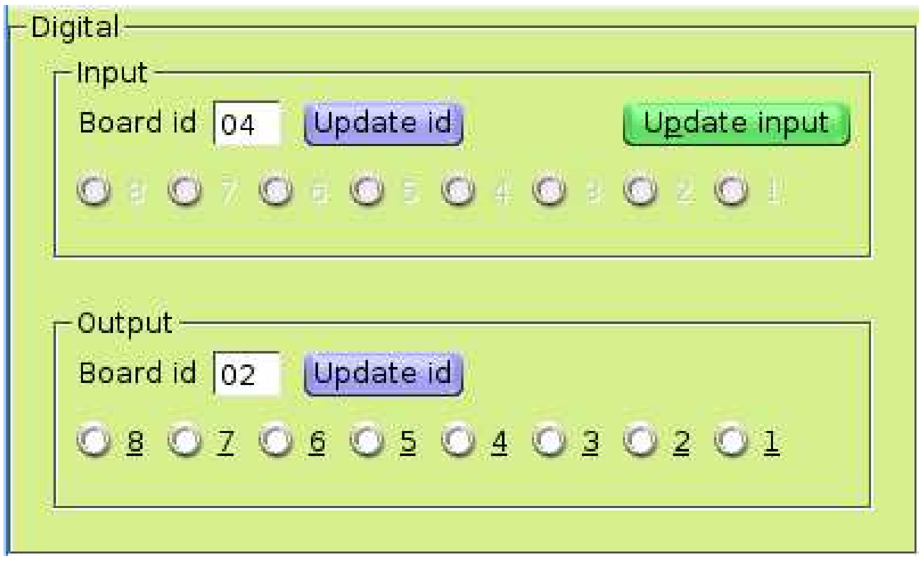


Figure 7.6: Rapier GUI digital input and output.

Input

Board id Enter a hex number in this field to set the id matching the board.

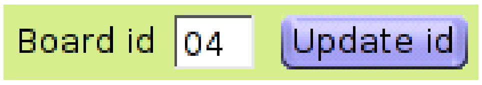


Figure 7.7: Rapier GUI digital input board id.

Input The input on the digital input board is displayed here. Click on the update button to update the display.

This input field is "read only".

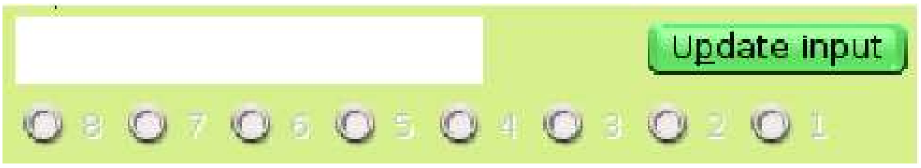


Figure 7.8: Rapier GUI digital input.

Output

Board id Enter a hex number in this field to set the id matching the board.

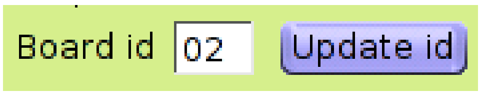


Figure 7.9: Rapier GUI digital output board id.

Output By clicking these radio buttons you can set the digital outputs.

Each radio button is accessible through keyboard short cuts. On the keyboard press ALT+1 through ALT+8 to toggle the radio buttons on and off.

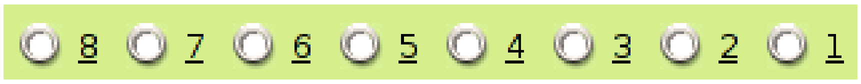
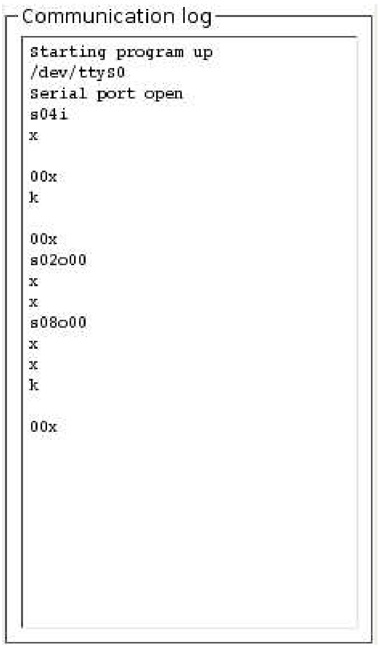


Figure 7.10: Rapier GUI digital output.

7.1.2.6 Communication log



```
Starting program up
/dev/ttyS0
Serial port open
s04i
x

00x
k

00x
s02o00
x
x
s08o00
x
x
k

00x
```

Figure 7.11: Rapier GUI communication log.

The log This is a "read only" part of the program. It is possible to mark the text and copy it into the clip board for later use in another program.

The text and messages are described later in the protocol section.

The most obvious use of this part is seeing that after every instruction send to the hardware, there is an "x" as a confirmation. If there is no "x" after each instruction, then there's something wrong. Typically a problem with the serial speed of the system.

In this part you are able to inspect the communication between the PC and the hardware, and some messages about the operation.

7.1.2.7 Quit

Figure 7.12: Rapier GUI quit button.

This button quits the program.

7.1.2.8 Connecting the hardware

Connect the Analog Input Board to any source of an analog voltage ranging from 0V - 2V.

Connect the Analog Output Board to voltmeter or any device controlled by an analog voltage in the range of 0V - 2V.

Connect the Digital Output Board to any device with up to 8 TTL-level[4] inputs.

Connect the Digital Input Board to any device with up to 8 TTL-level outputs.

To test the Rapier hardware, connect the Analog Input Board to Analog Output Board and connect the Digital Output Board to Digital Input Board. This method can be used for diagnostics and self test.

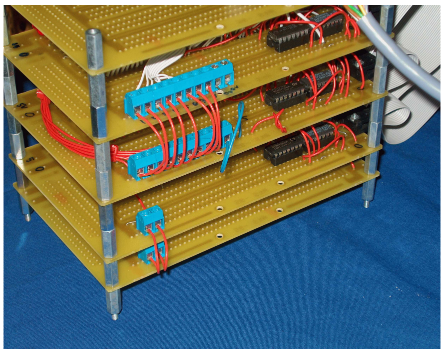


Figure 7.13: Looping output to input.

[4]Transistor-transistor logic (TTL) is a class of digital circuits built from bipolar junction transistors (BJT), and resistors; it is notable for being the base for the first widespread semiconductor integrated circuit (IC) technology. TTL gained almost universal acceptance after Texas Instruments had greatly facilitated the construction of digital systems with their 1962 introduction of the 7400 series of ICs.

7.2 References

This is a list of sources of information on which the project is based on:

- Doxygen at

  `http://www.stack.nl/~dimitri/doxygen/index.html`

- LaTeX 2_εreference at

  `http://www.latex-project.org/`

- C reference at

  `http://www-ccs.ucsd.edu/c/`

- C++ reference at

  `http://www.cplusplus.com/`

- QT-designer

  `http://www.trolltech.com/products/qt/designer.html`

- QT docs at

  `http://doc.trolltech.com/3.3/index.html`

- Bash howto at

  `http://www.tldp.org/LDP/abs/html/`

- Umrello editor (UML) at

  `http://uml.sourceforge.net/index.php`

- RS232 described at

  `http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html`

- 74ALS520 described at

  `http://focus.ti.com/docs/prod/folders/print/sn74als520.html`

- Z8 described at

  `http://www.zilog.com/products/partdetails.asp?id=Z8F6403`

- 74ls373 described at

  `http://focus.ti.com/docs/prod/folders/print/sn74ls373.html`

- 74ls74 described at

  `http://focus.ti.com/docs/prod/folders/print/sn74ls74a.html`

- 74ls00 described at

  `http://focus.ti.com/docs/prod/folders/print/sn74ls00.html`

- TL072 described at

  `http://focus.ti.com/docs/prod/folders/print/tl072.html`

- DAC0808 described at

  `http://www.national.com/pf/DA/DAC0808.html`

- 74ls04 described at

  `http://focus.ti.com/docs/prod/folders/print/sn74ls04.html`

- CNY17 described at

  `http://www.fairchildsemi.com/pf/CN/CNY17-2.html`

- ULN2803A described at

  `http://www.chipdocs.com/datasheets/datasheet-pdf/Allegro-MicroSystems-Inc/ULN280`

- Quanta described at

  `http://quanta.sourceforge.net/`

- Gimp described at

  `http://www.gimp.org/`

- Wikipedia, a free-content encyclopedia

  `http://en.wikipedia.org/wiki/Main_Page`

7.3 main.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <eZ8.h>
#include <sio.h>
#include "driver.h"
#include "main.h"
```

Functions

- char **getinput** ()

  Low nibble of a char. Reveives a character from the serial line.

- char **chartohex** (unsigned char tempchar)

  Converting a char into a number.

- void **bytetotwoascii** (unsigned char tempdata)

  Converts a byte in to 2 nibbles, and send it to the PC.

- void **main** ()

  Main function.

Variables

- char **mych**
- char **chararray** [ ] = "0123456789abcdef"

  Character received from the PC.

- char * **loc**

  String to hold hex values.

- unsigned char **datahigh**

  Hold the position of a hex value in the chararray[].

- unsigned char **datalow**

  High nibble of a char.

7.3.1 Function Documentation

7.3.1.1 void bytetotwoascii (unsigned char *tempdata*)

Converts a byte in to 2 nibbles, and send it to the PC.

Function to convert a byte into 2 nibbles, and each nibble converted into a hex value. Each nibble is send to the PC through the serial connection.

Definition at line 64 of file main.c.

References chararray, datahigh, and datalow.

Referenced by main().

7.3.1.2 char chartohex (unsigned char *tempchar*)

Converting a char into a number.

Function to convert a hex value ranging from 0 - f into a number ranging from 0 - 15.

Definition at line 51 of file main.c.

References chararray, and loc.

Referenced by main().

7.3.1.3 char getinput ()

Low nibble of a char. Reveives a character from the serial line.

Function to receive a single character from the RS232 connection to the PC.

Definition at line 38 of file main.c.

References mych.

Referenced by main().

7.3.1.4 void main ()

Main function.

The main function. This function sets up the AD bus, Control bus and the serial port. Main takes care of the principal functionallity in the system.

Definition at line 79 of file main.c.

References ADBUSOUT, adressingboard(), bussetup(), bytein(), byteout(), bytetotwoascii(), chartohex(), datahigh, datalow, delaycycles, getinput(), mych, singleadconverting(), and writeFlash().

7.3.2 Variable Documentation

7.3.2.1 char chararray[] = "0123456789abcdef"

Character received from the PC.

Definition at line 9 of file main.c.

Referenced by bytetotwoascii(), and chartohex().

7.3.2.2 unsigned char datahigh

Hold the position of a hex value in the chararray[].

Definition at line 11 of file main.c.

Referenced by bytetotwoascii(), and main().

7.3.2.3 unsigned char datalow

High nibble of a char.

Definition at line 12 of file main.c.

Referenced by bytetotwoascii(), and main().

7.3.2.4 char∗ loc

String to hold hex values.

Definition at line 10 of file main.c.

Referenced by chartohex().

7.3.2.5 char mych

Definition at line 8 of file main.c.

Referenced by getinput(), and main().

7.4 main.h File Reference

Functions

- void **writeFlash** (void)
- void **init_flash** (unsigned long freq)

Variables

- char **x**
- long **delaycycles**

7.4.1 Function Documentation

7.4.1.1 void init_flash (unsigned long *freq*)

7.4.1.2 void writeFlash (void)

Definition at line 6 of file z8-system.c.

Referenced by main().

7.4.2 Variable Documentation

7.4.2.1 long delaycycles

Definition at line 4 of file main.h.

Referenced by main().

7.4.2.2 char x

Definition at line 3 of file main.h.

Referenced by writeFlash().

7.5 driver.c File Reference

```
#include <ez8.h>
#include "main.h"
#include "driver.h"
```

Functions

- void **adbusoutput** (char c)
  Byte value to A/D bus.
- void **adress_enable_on** ()
  Enables Address Enable signal.
- void **adress_enable_off** ()
  Disables Address Enable signal.
- void **adress_decode_on** ()
  Enables Address Decode signal.
- void **adress_decode_off** ()
  Disables Address Decode signal.
- void **output_function_enable_on** ()
  Enables Output Function Enable signal.
- void **output_function_enable_off** ()
  Disables Output Function Enable signal.
- void **input_function_enable_on** ()
  Enables Input Function Enable signal.
- void **input_function_enable_off** ()
  Disables Input Function Enable signal.
- void **deselect_board_on** ()
  Enables Deselect Board signal.
- void **deselect_board_off** ()
  Disables Deselect Board signal.
- void **adressingboard** (char id)
  Addressing a board.
- void **byteout** (char data)

Sends a byte to a board.

- char **bytein** ()

  Recieves a byte from a board.

- char **singleadconverting** ()

  A/D converting.

- void **dob** (char id, char data)

  Digital Output Board operation.

- char **dib** (char id)

  Digital Input Board operation.

- void **aob** (char id, char data)

  Analog Output Board operation.

- char **aib** ()

  Analog Input Board operation.

- void **setadcinputs** ()

  Seting up port H to analog inputs.

- void **setadbustoinput** ()

  Setup port G to input.

- void **setadbustooutput** ()

  Setup port G to output.

- void **bussetup** ()

  Seting up Bus system.

7.5.1 Function Documentation

7.5.1.1 void adbusoutput (char *c*)

Byte value to A/D bus.

Function to output a char to the AD bus.

Definition at line 13 of file driver.c.

References ADBUSOUT.

Referenced by adressingboard(), and byteout().

7.5.1.2 void adress_decode_off ()

Disables Address Decode signal.

Function to turn on the Address Decode pin on Control Bus low. Does not affect other signals on the control.

Definition at line 60 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.5.1.3 void adress_decode_on ()

Enables Address Decode signal.

Function to turn on the Address Decode pin on Control Bus high. Does not affect other signals on the control.

Definition at line 48 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.5.1.4 void adress_enable_off ()

Disables Address Enable signal.

Function to turn on the Address Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 36 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.5.1.5 void adress_enable_on ()

Enables Address Enable signal.

Function to turn on the Address Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 24 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.5.1.6 void adressingboard (char *id*)

Addressing a board.

Function to put an address on the AD bus.

Definition at line 145 of file driver.c.

References adbusoutput(), adress_decode_off(), adress_decode_on(), adress_-enable_off(), and adress_enable_on().

Referenced by aob(), dib(), dob(), and main().

7.5.1.7 char aib ()

Analog Input Board operation.

Function to perform a full AD conversion. This function exists to match the other board functions.

Definition at line 247 of file driver.c.

References singleadconverting().

7.5.1.8 void aob (char *id*, char *data*)

Analog Output Board operation.

Function to perform a full board operation for a Analog Output Board.

Definition at line 235 of file driver.c.

References adressingboard(), and byteout().

7.5.1.9 void bussetup ()

Seting up Bus system.

Function to set up the Z8 port E for Control bus use.

Definition at line 329 of file driver.c.

References ADBUSOUT, CONTROLBUSOUT, deselect_board_off(), deselect_-board_on(), setadbustooutput(), and setadcinputs().

Referenced by main().

7.5.1.10 char bytein ()

Recieves a byte from a board.

Function to reveive data from a board. The function sets the AD bus to input mode before requesting data from the board. After data has been reveived, the board is turned off and AD bus set to output mode again.

Definition at line 176 of file driver.c.

References ADBUSIN, input_function_enable_off(), input_function_enable_-on(), setadbustoinput(), and setadbustooutput().

Referenced by dib(), and main().

7.5.1.11 void byteout (char *data*)

Sends a byte to a board.

Function to put data on the AD bus and toggle Output Function enable ON and OFF.

Definition at line 161 of file driver.c.

References adbusoutput(), output_function_enable_off(), and output_function_-enable_on().

Referenced by aob(), dob(), and main().

7.5.1.12 void deselect_board_off ()

Disables Deselect Board signal.

Function to turn on the Deselect Board pin on Control Bus high. Does not affect other signals on the control.

Definition at line 132 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bussetup().

7.5.1.13 void deselect_board_on ()

Enables Deselect Board signal.

Function to turn on the Deselect Board pin on Control Bus high. Does not affect other signals on the control.

Definition at line 120 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bussetup().

7.5.1.14 char dib (char *id*)

Digital Input Board operation.

Function to perform a full board operation for a Digital Input Board.

Definition at line 221 of file driver.c.

References adressingboard(), and bytein().

7.5.1.15 void dob (char *id*, char *data*)

Digital Output Board operation.

Function to perform a full board operation for a Digital Output Board.

Definition at line 208 of file driver.c.

References adressingboard(), and byteout().

7.5.1.16 void input_function_enable_off ()

Disables Input Function Enable signal.

Function to turn on the Input Function Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 108 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bytein().

7.5.1.17 void input_function_enable_on ()

Enables Input Function Enable signal.

Function to turn on the Input Function Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 96 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bytein().

7.5.1.18 void output_function_enable_off ()

Disables Output Function Enable signal.

Function to turn on the Output Function Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 84 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by byteout().

7.5.1.19 void output_function_enable_on ()

Enables Output Function Enable signal.

Function to turn on the Output Function Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 72 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by byteout().

7.5.1.20 void setadbustoinput ()

Setup port G to input.

Function to set up the Z8 port G for AD bus input mode use.

Definition at line 303 of file driver.c.

Referenced by bytein().

7.5.1.21 void setadbustooutput ()

Setup port G to output.

Function to set up the Z8 port G for AD bus output mode use.

Definition at line 316 of file driver.c.

Referenced by bussetup(), and bytein().

7.5.1.22 void setadcinputs ()

Seting up port H to analog inputs.

Function to set up the ADC in the Z8.

Definition at line 288 of file driver.c.

Referenced by bussetup().

7.5.1.23 char singleadconverting ()

A/D converting.

Function to perform an AD conversion in the Z8. Only the 8 highest bit of the 10 bit result is used.

Definition at line 193 of file driver.c.

Referenced by aib(), and main().

7.6 driver.h File Reference

Defines

- #define **ADBUSOUT** PGOUT
- #define **ADBUSIN** PGIN
- #define **CONTROLBUSOUT** PEOUT
- #define **CONTROLBUSIN** PEIN

Functions

- void **adbusoutput** (char c)

  Byte value to A/D bus.
- void **adress_enable_on** ()

  Enables Address Enable signal.
- void **adress_enable_off** ()

  Disables Address Enable signal.
- void **adress_decode_on** ()

  Enables Address Decode signal.
- void **adress_decode_off** ()

  Disables Address Decode signal.
- void **output_function_enable_on** ()

  Enables Output Function Enable signal.
- void **output_function_enable_off** ()

  Disables Output Function Enable signal.
- void **input_function_enable_on** ()

  Enables Input Function Enable signal.
- void **input_function_enable_off** ()

  Disables Input Function Enable signal.
- void **deselect_board_on** ()

  Enables Deselect Board signal.
- void **deselect_board_off** ()

  Disables Deselect Board signal.
- void **setadcinputs** ()

  Seting up port H to analog inputs.

- void **setadbustoinput** ()

  Setup port G to input.

- void **setadbustooutput** ()

  Setup port G to output.

- void **bussetup** ()

  Seting up Bus system.

- char **nightridertest** ()
- void **testcontrols** ()
- void **adressingboard** (char id)

  Addressing a board.

- void **byteout** (char data)

  Sends a byte to a board.

- char **bytein** ()

  Recieves a byte from a board.

- char **singleadconverting** ()

  A/D converting.

- void **dob** (char id, char data)

  Digital Output Board operation.

- char **dib** (char id)

  Digital Input Board operation.

- void **aob** (char id, char data)

  Analog Output Board operation.

- char **aib** ()

  Analog Input Board operation.

- void **longdelay** ()

7.6.1 Define Documentation

7.6.1.1 #define ADBUSIN PGIN

Definition at line 2 of file driver.h.

Referenced by bytein().

7.6.1.2 #define ADBUSOUT PGOUT

Definition at line 1 of file driver.h.

Referenced by adbusoutput(), bussetup(), and main().

7.6.1.3 #define CONTROLBUSIN PEIN

Definition at line 4 of file driver.h.

Referenced by adress_decode_off(), adress_decode_on(), adress_enable_off(), adress_enable_on(), deselect_board_off(), deselect_board_on(), input_function_-enable_off(), input_function_enable_on(), output_function_enable_off(), and output_function_enable_on().

7.6.1.4 #define CONTROLBUSOUT PEOUT

Definition at line 3 of file driver.h.

Referenced by adress_decode_off(), adress_decode_on(), adress_enable_off(), adress_enable_on(), bussetup(), deselect_board_off(), deselect_board_on(), input_function_enable_off(), input_function_enable_on(), output_function_-enable_off(), and output_function_enable_on().

7.6.2 Function Documentation

7.6.2.1 void adbusoutput (char *c*)

Byte value to A/D bus.

Function to output a char to the AD bus.

Definition at line 13 of file driver.c.

References ADBUSOUT.

Referenced by adressingboard(), and byteout().

7.6.2.2 void adress_decode_off ()

Disables Address Decode signal.

Function to turn on the Address Decode pin on Control Bus low. Does not affect other signals on the control.

Definition at line 60 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.6.2.3 void adress_decode_on ()

Enables Address Decode signal.

Function to turn on the Address Decode pin on Control Bus high. Does not affect other signals on the control.

Definition at line 48 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.6.2.4 void adress_enable_off ()

Disables Address Enable signal.

Function to turn on the Address Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 36 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.6.2.5 void adress_enable_on ()

Enables Address Enable signal.

Function to turn on the Address Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 24 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by adressingboard().

7.6.2.6 void adressingboard (char *id*)

Addressing a board.

Function to put an address on the AD bus.

Definition at line 145 of file driver.c.

References adbusoutput(), adress_decode_off(), adress_decode_on(), adress_-enable_off(), and adress_enable_on().

Referenced by aob(), dib(), dob(), and main().

7.6.2.7 char aib ()

Analog Input Board operation.

Function to perform a full AD conversion. This function exists to match the other board functions.

Definition at line 247 of file driver.c.

References singleadconverting().

7.6.2.8 void aob (char *id*, char *data*)

Analog Output Board operation.

Function to perform a full board operation for a Analog Output Board.

Definition at line 235 of file driver.c.

References adressingboard(), and byteout().

7.6.2.9 void bussetup ()

Seting up Bus system.

Function to set up the Z8 port E for Control bus use.

Definition at line 329 of file driver.c.

References ADBUSOUT, CONTROLBUSOUT, deselect_board_off(), deselect_-board_on(), setadbustooutput(), and setadcinputs().

Referenced by main().

7.6.2.10 char bytein ()

Recieves a byte from a board.

Function to reveive data from a board. The function sets the AD bus to input mode before requesting data from the board. After data has been reveived, the board is turned off and AD bus set to output mode again.

Definition at line 176 of file driver.c.

References ADBUSIN, input_function_enable_off(), input_function_enable_-on(), setadbustoinput(), and setadbustooutput().

Referenced by dib(), and main().

7.6.2.11 void byteout (char *data*)

Sends a byte to a board.

Function to put data on the AD bus and toggle Output Function enable ON and OFF.

Definition at line 161 of file driver.c.

References adbusoutput(), output_function_enable_off(), and output_function_-enable_on().

Referenced by aob(), dob(), and main().

7.6.2.12 void deselect_board_off ()

Disables Deselect Board signal.

Function to turn on the Deselect Board pin on Control Bus high. Does not affect other signals on the control.

Definition at line 132 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bussetup().

7.6.2.13 void deselect_board_on ()

Enables Deselect Board signal.

Function to turn on the Deselect Board pin on Control Bus high. Does not affect other signals on the control.

Definition at line 120 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bussetup().

7.6.2.14 char dib (char *id*)

Digital Input Board operation.

Function to perform a full board operation for a Digital Input Board.

Definition at line 221 of file driver.c.

References adressingboard(), and bytein().

7.6.2.15 void dob (char *id*, char *data*)

Digital Output Board operation.

Function to perform a full board operation for a Digital Output Board.

Definition at line 208 of file driver.c.

References adressingboard(), and byteout().

7.6.2.16 void input_function_enable_off ()

Disables Input Function Enable signal.

Function to turn on the Input Function Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 108 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bytein().

7.6.2.17 void input_function_enable_on ()

Enables Input Function Enable signal.

Function to turn on the Input Function Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 96 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by bytein().

7.6.2.18 void longdelay ()

7.6.2.19 char nightridertest ()

7.6.2.20 void output_function_enable_off ()

Disables Output Function Enable signal.

Function to turn on the Output Function Enable pin on Control Bus low. Does not affect other signals on the control.

Definition at line 84 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by byteout().

7.6.2.21 void output_function_enable_on ()

Enables Output Function Enable signal.

Function to turn on the Output Function Enable pin on Control Bus high. Does not affect other signals on the control.

Definition at line 72 of file driver.c.

References CONTROLBUSIN, and CONTROLBUSOUT.

Referenced by byteout().

7.6.2.22 void setadbustoinput ()

Setup port G to input.

Function to set up the Z8 port G for AD bus input mode use.

Definition at line 303 of file driver.c.

Referenced by bytein().

7.6.2.23 void setadbustooutput ()

Setup port G to output.

Function to set up the Z8 port G for AD bus output mode use.

Definition at line 316 of file driver.c.

Referenced by bussetup(), and bytein().

7.6.2.24 void setadcinputs ()

Seting up port H to analog inputs.

Function to set up the ADC in the Z8.

Definition at line 288 of file driver.c.

Referenced by bussetup().

7.6.2.25 char singleadconverting ()

A/D converting.

Function to perform an AD conversion in the Z8. Only the 8 highest bit of the 10 bit result is used.

Definition at line 193 of file driver.c.

Referenced by aib(), and main().

7.6.2.26 void testcontrols ()

7.7 main.cpp File Reference

`#include "mainwindow.h"`

`#include <qapplication.h>`

Functions

- int **main** (int argc, char **argv)
  Main function in the GUI application.

7.7.1 Function Documentation

7.7.1.1 int main (int *argc*, char ** *argv*)

Main function in the GUI application.

Function to start the GUI.

Definition at line 13 of file main.cpp.

```
13                                 {
14   QApplication app(argc, argv );
15   MainWindow window;
16   window.show();
17   app.connect( &app, SIGNAL( lastWindowClosed() ), &app, SLOT( quit() ) );
18   return app.exec();
19 }
```

7.8 mainwindow.ui.h File Reference

`#include <unistd.h>`

`#include <string.h>`

`#include <errno.h>`

`#include <sys/types.h>`

`#include <sys/stat.h>`

`#include <fcntl.h>`

`#include <termios.h>`

`#include <stdio.h>`

Variables

- int **fd** = 0
- char **sCmd** [254]
- char **sResult** [254]
- QString **serialport** = "/dev/ttyS0"
- QString **aoid** = "08"
- QString **doid** = "02"
- QString **diid** = "04"

7.8.1 Variable Documentation

7.8.1.1 QString aoid = "08"

Definition at line 25 of file mainwindow.ui.h.

7.8.1.2 QString diid = "04"

Definition at line 27 of file mainwindow.ui.h.

7.8.1.3 QString doid = "02"

Definition at line 26 of file mainwindow.ui.h.

7.8.1.4 int fd = 0

Definition at line 21 of file mainwindow.ui.h.

7.8.1.5 char sCmd[254]

Definition at line 22 of file mainwindow.ui.h.

7.8.1.6 QString serialport = "/dev/ttyS0"

Definition at line 24 of file mainwindow.ui.h.

7.8.1.7 char sResult[254]

Definition at line 23 of file mainwindow.ui.h.

```
/****************************************************************************
 ** ui.h extension file, included from the uic-generated form implementation.
 **
 ** If you want to add, delete, or rename functions or slots, use
 ** Qt Designer to update this file, preserving your code.
 **
 ** You should not define a constructor or destructor in this file.
 ** Instead, write your code in functions called init() and destroy().
 ** These will automatically be called by the form's constructor and
 ** destructor.
 *****************************************************************************/
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>

int fd = 0;
char sCmd[254];
char sResult[254];
QString serialport = "/dev/ttyS0";
QString aoid = "08";
QString doid = "02";
QString diid = "04";



/**
 * @brief Initializes the GUI.
 * @return Errorlevel as an int.
 */
/*!
  Function to start up the GUI and setting some default values.
*/
int MainWindow::init(){
  CommunicationLog->append("Starting program up");
  //QRegExp regExp("[-]{0,1}\\d*\\.\\d+");
  //QRegExp regExp("^[\xhhhh]{1,2}$");
  //  AIid->setValidator(new QRegExpValidator(regExp, this));
  //  DIid->setValidator(new QRegExpValidator(regExp, this));
  //  DOid->setValidator(new QRegExpValidator(regExp, this));
  //  system("/bin/setserial /dev/ttyS0");
  AOid->setText(aoid);
  DIid->setText(diid);
  DOid->setText(doid);
  if (OpenPort() < 0) {
    return 0;
    }
    CommunicationLog->append("Serial port open");
    UpdateDigitalInput();
    UpdateAnalogInput();
    UpdateDigitalOutput();
    UpdateAnalogOutput();
    return 0;
}

/**
 * @brief
 * @param
 * @return
 */
/*!
    Function to
*/
void MainWindow::destroy(){
    CommunicationLog->append("Closing program down");
}

/**
 * @brief
 * @param
 * @return
 */
/*!
    Function to
*/
QString MainWindow::GetInput( QString command ){
    CommunicationLog->append(command);
    QString toreturn = "";
    QString tempstr = "";
    WritePort(command);
    usleep(100000);
    if (ReadPort(sResult,254) > 0) {
        toreturn = sResult;
        CommunicationLog->append(toreturn);
    }
    for(unsigned int i = 0; i<=toreturn.length() ;i++){
        if(toreturn[i] != 'x'){
            tempstr+=toreturn[i];
        }
    }
    toreturn = tempstr.simplifyWhiteSpace();
    return toreturn;
}

/**
 * @brief
 * @param
 * @return
 */
/*!
   Function to
*/
void MainWindow::UpdateAnalogInput(){
   QString hexstr;
   QString outputstr;
   // Sends instuction to do a single A/D conversion
   hexstr = GetInput("k");
   // Strips off the unused characters in string
   hexstr = hexstr.left( 2 );

   // Prepends characters before the Hex to Dec conversion
   hexstr = hexstr.prepend("0x");
   // Doing the Hex to Dec conversion
   bool ok;
   int voltage1 =  hexstr.toInt(&ok, 16);
   // Typecasting the voltage value to a double
   double voltage2 = (double) voltage1 * 0.007843137;
   voltage2 *= 100.0;
   double param, fractpart, intpart;
   param = voltage2;
   fractpart = modf (param , &intpart);
   if((fractpart*10.0)>4){
   intpart++;
   }
   voltage2 = (double) intpart * 0.01;
   outputstr.setNum(voltage2);
   outputstr.append(" V");
   LineEditAI->setText(outputstr);
}

/**
 * @brief
 * @param
 * @return
 */
/*!
   Function to
*/
void MainWindow::UpdateAnalogOutput(){
   bool ok;
   //QString output = "s08";
   QString output = "s";
   output.append(aoid);

   QString voltage = spinBoxAO->text();
   QString stripvoltage = voltage.left(voltage.find(" "));
```

```
    stripvoltage = stripvoltage.simplifyWhiteSpace();
    int ivoltage = stripvoltage.toInt(&ok, 10);
    double dvoltage = ((double)ivoltage / 7.81640625);
    ivoltage = (int)dvoltage;
    QString str = QString( "%1" ).arg( ivoltage, 0, 16 );
    output.append("o");
    if(str.length() == 1){
       output.append("0");
    }
    output.append(str);
    GetInput(output);
    UpdateAnalogInput();
    //output.append( voltage );
}

/**
 * @brief
 * @param
 * @return
 */
/*!
    Function to
*/
int MainWindow::OpenPort(){
    CommunicationLog->append(serialport.ascii());
    //char sPortName[64]="/dev/ttyS0";
    //char sPortName[64]=serialport.latin1();
    ClosePort();
    //fd = open(sPortName, O_RDWR | O_NOCTTY | O_NDELAY);
    fd = open(serialport.ascii(), O_RDWR | O_NOCTTY | O_NDELAY);
    if (fd < 0) {
  printf("open error %d %s\n", errno, strerror(errno));
    } else {
  struct termios my_termios;
  tcgetattr(fd, &my_termios);
  tcflush(fd, TCIFLUSH);
  my_termios.c_cflag = B9600 | CS8 |CREAD | CLOCAL | HUPCL;
    cfsetospeed(&my_termios, B57600);
    //cfsetospeed(&my_termios, B38400);
  tcsetattr(fd, TCSANOW, &my_termios);
    }
    return fd;
}


/**
 * @brief
 * @param
 * @return
 */
```

```
/*!
   Function to
*/
void MainWindow::ClosePort(){
   if (fd > 0) {
  close(fd);
   }
}


/**
 * @brief
 * @param
 * @return
 */
/*!
   Function to
*/
int MainWindow::ReadPort( char * psResponse, int iMax ){
   int iIn;
   if (fd < 1) {
  printf(" port is not open\n");
  return -1;
   }
   strncpy (psResponse, "N/A", iMax<4?iMax:4);
   iIn = read(fd, psResponse, iMax-1);
   if (iIn < 0) {
  if (errno == EAGAIN) {
     return 0;
  } else {
     printf("read error %d %s\n", errno, strerror(errno));
  }
   } else {
  psResponse[iIn<iMax?iIn:iMax] = '\0';
   }
   return iIn;
}


/**
 * @brief
 * @param
 * @return
 */
/*!
   Function to
*/
int MainWindow::WritePort( QString psOutput ){
   int iOut;
   if (fd < 1) {
```

```
 printf("Port is not open\n");
 return −1;
  }
  iOut = write(fd, psOutput.ascii(), strlen(psOutput));
  if (iOut < 0) {
  printf("write error %d %s\n", errno, strerror(errno));
  } else {
  }
  return iOut;
}



/**
 * @brief
 * @param
 * @return
 */
/*!
  Function to
*/
void MainWindow::UpdateDigitalInput(){
  QString hexstr;
  QString str = "s";
  str.append(diid);
  str.append("i");
  hexstr = GetInput(str);

  hexstr = hexstr.left( 2 );
  hexstr = hexstr.prepend("0x");
  bool ok;
  int value =  hexstr.toInt(&ok, 16);

  if(value >= 128){
    DI8−>setChecked(1);
    value −= 128;
  } else {
    DI8−>setChecked(0);
  }
  if(value >= 64){
    DI7−>setChecked(1);
    value −= 64;
  } else {
    DI7−>setChecked(0);
  }
  if(value >= 32){
    DI6−>setChecked(1);
    value −= 32;
  } else {
    DI6−>setChecked(0);
    }
    if(value >= 16){
        DI5->setChecked(1);
        value -= 16;
    } else {
        DI5->setChecked(0);
    }
    if(value >= 8){
        DI4->setChecked(1);
        value -= 8;
    } else {
        DI4->setChecked(0);
    }
    if(value >= 4){
        DI3->setChecked(1);
        value -= 4;
    } else {
        DI3->setChecked(0);
    }
    if(value >= 2){
        DI2->setChecked(1);
        value -= 2;
    } else {
        DI2->setChecked(0);
    }
    if(value >= 1){
        DI1->setChecked(1);
        value -= 1;
    } else {
        DI1->setChecked(0);
    }
}

/**
 * @brief
 * @param
 * @return
 */
/*!
    Function to
*/
void MainWindow::UpdateDigitalOutput(){
    QString str = "s";
    str.append(doid);
    str.append("o");
    int value = 0;
    if(DO8->isChecked()){
        value += 128;
    }
    if(DO7->isChecked()){
        value += 64;
    }
    if(DO6->isChecked()){
        value += 32;
    }
    if(DO5->isChecked()){
        value += 16;
    }
    if(DO4->isChecked()){
        value += 8;
    }
    if(DO3->isChecked()){
        value += 4;
    }
    if(DO2->isChecked()){
        value += 2;
    }
    if(DO1->isChecked()){
        value += 1;
    }
    QString tempstr = QString( "%1" ).arg( value, 0, 16 );
    if(tempstr.length() == 1){
        tempstr.prepend("0");
    }
    str.append(tempstr);
    GetInput(str);
}

/**
 * @brief Sets the new Analog Output Board id.
 * @return Void.
 */
/*!
    Function to set the new Analog Output Board id by setting the id in a QString and inforn
*/
void MainWindow::changeAOid(){
    aoid = AOid->text();
    CommunicationLog->append("Analog Output id changed");
    CommunicationLog->append(aoid);
}

/**
 * @brief Sets the new Digital Input Board id.
 * @return Void.
 */
/*!
    Function to set the new Digital Input Board id by setting the id in a QString and inform
*/
void MainWindow::changeDIid(){
    diid = DIid->text();
```

```
   CommunicationLog->append("Digital Input id changed");
   CommunicationLog->append(diid);
}

/**
 * @brief Sets the new Digital Output Board id.
 * @return Void.
 */
/*!
   Function to set the new Digital Output Board id by setting the id in a QString and inform the user
*/
void MainWindow::changeDOid(){
   doid = DOid->text();
   CommunicationLog->append("Digital Output id changed");
   CommunicationLog->append(doid);
}
```

7.9 Rapierproject code

This is the source code for the project file management system:

```
#!/bin/bash
DIALOG=${DIALOG=dialog}
tempfile=‘tempfile 2>/dev/null‘ || tempfile=/tmp/test$$
choice=1
#WEBTARGET="/var/www/sfmirror"
WEBTARGET="karneevor@rapier.sourceforge.net:/home/groups/r/ra/rapier/htdocs"
trap "rm -f $tempfile" 0 1 2 5 15

function toggletarget(){
    if [ "$WEBTARGET" == "karneevor@rapier.sourceforge.net:/home/groups/r/ra/rapier/htdocs" ]
    then
WEBTARGET="/var/www/sfmirror"
    else
WEBTARGET="karneevor@rapier.sourceforge.net:/home/groups/r/ra/rapier/htdocs"
    fi
}

function UpdateZ8Documentation(){
    echo "Getting Z8 documentation"
    echo "running doxygen in Z8 code"
    cd /mnt/c-drev/rapier
    doxygen doxygen.conf
    echo "Updating the Z8 code documentation into latex"
    rsync -vut /mnt/c-drev/rapier/docs/latex/*   /data/rapier-project/documentation/latex/z8/
    echo "Updating the Z8 code documentation into html"
    rsync -vut /mnt/c-drev/rapier/docs/html/*   /data/rapier-project/sourceforge/html/documentation/z8/
    cd /data/rapier-project
}

function UpdateGUIDocumentation(){
    echo "Getting GUI documentation"
    echo "Running doxygen in GUI code"
    cd /data/rapier-project/software/gui
    doxygen doxygen.conf

    echo "Updating the GUI code documentation into latex"
    rsync -vut /data/rapier-project/software/gui/docs/latex/*   /data/rapier-project/documentation/latex/gui/
    echo "Updating the GUI code documentation into html"
    rsync -vut /data/rapier-project/software/gui/docs/html/*   /data/rapier-project/sourceforge/html/documentation/gui/

    cd /data/rapier-project

    sleep 4
}

function BackupTools(){
    echo "Backing up Tools ..."
    itctools --dir-full tools/
}

function BackupManagement(){
    echo "Backing up Management ..."
    itctools --dir-full management/
}

function BackupHardware(){
    echo "Backing up Hardware ..."
    itctools --dir-full hardware/
}

function BackupSoftware(){
    echo "Backing up software ..."
    itctools --dir-full software/
}

function BackupSourceForge(){
    echo "Backing up source forge ..."
    itctools --dir-full sourceforge/
}

function BackupDocumentation(){
    echo "Backing up documentation ..."
    itctools --dir-full documentation/
}


function MakeLatexTools(){
    echo "Making latex tools."
    cd tools/dds/
    tar czvf ../../sourceforge/html/tools/unitycircle.tgz unitycircle/*
    tar czvf ../../sourceforge/html/tools/sinuswave.tgz   sinuswave/*
    cd ../..
}

function CleanUpBackups(){
    echo "Cleaning out backups."
    find . -name *~ -print -exec rm -v {} \;
    find . -name *backup -print -exec rm -v {} \;
    sleep 1
}
```

```
function MakeTarlatex(){
    echo "Making a tar ball from the latex documentation."
    cd documentation
    tar czvf ../sourceforge/html/documentation/rapier.latex.tgz latex/*
    cd ..
}

function Getrapierproject(){
    echo "Updating rapierproject."
    rsync -vut rapierproject                             sourceforge/html/tools/rapierproject
    rsync -vut rapierproject                             documentation/latex/sw-bmd/rapierproject
}

function Getitctools(){
    echo "Updating itctools."
    rsync -vut /usr/bin/itctools                           sourceforge/html/tools/itctools
}

function Z8code(){
    echo "Collecting the Z8 source code."
    rsync -vut /mnt/c-drev/rapier/* software/z8/
    CleanUpBackups
    cd software/
    tar czvf ../sourceforge/html/sourcecode/z8.tgz       z8/*
    cd ..
}

function GUIcode(){
    echo "Collecting the GUI code."
    rsync -vut /var/www/banshee/*         software/web-gui/
    CleanUpBackups
    cd software/
    tar czvf ../sourceforge/html/sourcecode/gui.tgz  gui/*
    cd ..
}

function BuildOnlineDocumentationFromPDF(){
    echo "Building online documentaion from PDF."
    cd sourceforge/html/documentation/online/
    cp ../rapier.pdf ./
    pdftohtml -c rapier.pdf
    rm rapier.pdf
    cd ../../../..
}

function CompileLatex(){
    echo "Compile Latex."
    cd documentation/latex/
    echo "Compiling LaTeX files"
    latex rapier.tex
    cd ../..
}

function DVI2PDF(){
    echo "Creating PDF from DVI."
    cd documentation/latex/
    echo "DVI -> PDF"
    dvipdf rapier.dvi
    cd ../..
    mv documentation/latex/rapier.pdf sourceforge/html/documentation/
}

function UploadHomepage(){
    echo "Uploading homepage"
    echo "html"
    rsync -vut -e ssh sourceforge/html/*html               $WEBTARGET/
    echo "jpg"
    rsync -vut -e ssh sourceforge/html/*jpg                $WEBTARGET/
    echo "php"
    rsync -vut -e ssh sourceforge/html/*php                $WEBTARGET/
    echo "pictures/"
    rsync -vut -e ssh sourceforge/html/pictures/*          $WEBTARGET/pictures/
    echo "pics/"
    rsync -vut -e ssh sourceforge/html/pics/*              $WEBTARGET/pics/
}

function UploadOnlineDocumentation(){
    echo "Uploading Online Documentation"
    echo "online"
    rsync -vut -e ssh sourceforge/html/documentation/online/*     $WEBTARGET/documentation/online/
    echo "z8"
    rsync -vut -e ssh sourceforge/html/documentation/z8/*         $WEBTARGET/documentation/z8/
    echo "gui"
    rsync -vut -e ssh sourceforge/html/documentation/gui/*         $WEBTARGET/documentation/gui/
}

function UploadLatexDocumentation(){
    echo "Uploading Latex documenation"
    rsync -vut -e ssh sourceforge/html/documentation/rapier.latex.tgz $WEBTARGET/documentation/
}

function UploadPdfDocumentation(){
    echo "Uploading PDF documenation"
    rsync -vut -e ssh sourceforge/html/documentation/rapier.pdf $WEBTARGET/documentation/
}

function UploadSourceCode(){
    echo "Uploading Source Code"
    rsync -vut -e ssh sourceforge/html/sourcecode/*        $WEBTARGET/sourcecode/
```

```
}

function UploadTools(){
    echo "Uploading Tools"
    rsync -vut -e ssh sourceforge/html/tools/*             $WEBTARGET/tools/
}

function UploadApplicationNotes(){
    echo "Uploading Application Notes"
    rsync -vut -e ssh sourceforge/html/application_notes/* $WEBTARGET/application_notes/
}

function CopyPartlists(){
    echo "Getting Partlists"
    cp -v hardware/*/*partlist.txt documentation/latex/hw-bmd/
}

function UpdatePhotos(){
    BASEDIRNAME[1]="documentation/photo"
    BASEDIRNAME[2]="documentation/photo"
    BASEDIRNAME[3]="documentation/photo"
    BASEDIRNAME[4]="documentation/photo"
    BASEDIRNAME[5]="documentation/photo"
    BASEDIRNAME[6]="documentation/photo"
    BASEDIRNAME[7]="documentation/photo"
    BASEDIRNAME[8]="documentation/photo"
    BASEDIRNAME[9]="documentation/photo"
    BASEDIRNAME[10]="documentation/photo"
    BASEDIRNAME[11]="documentation/photo"
    BASEDIRNAME[12]="documentation/photo"

    BASEDIRNAME[13]="test_and_measurement/analog_IO"
    BASEDIRNAME[14]="test_and_measurement/analog_IO"
    BASEDIRNAME[15]="test_and_measurement/analog_IO"
    BASEDIRNAME[16]="test_and_measurement/analog_IO"

    BASEFILENAME[1]="ab-buttom-photo"
    BASEFILENAME[2]="ab-top-photo"
    BASEFILENAME[3]="aib-photo"
    BASEFILENAME[4]="aob-photo"
    BASEFILENAME[5]="dib-photo"
    BASEFILENAME[6]="dob-photo"
    BASEFILENAME[7]="full-system1-photo"
    BASEFILENAME[8]="full-system2-photo"
    BASEFILENAME[9]="mpb-photo"
    BASEFILENAME[10]="z8-photo"
    BASEFILENAME[11]="loop_output_to_input"
    BASEFILENAME[12]="aob2v"

    BASEFILENAME[13]="ad-linearity"
    BASEFILENAME[14]="da-linearity"
    BASEFILENAME[15]="dad-linearity"
    BASEFILENAME[16]="user-input-linearity"

    for index in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
    do
      echo ""
      echo ""
      echo ""
      echo "Managing ${BASEDIRNAME[index]} \ ${BASEFILENAME[index]}"
      echo ""
      BASEDIRNAME="${BASEDIRNAME[index]}"
      BASEFILENAME="${BASEFILENAME[index]}"
      convert -verbose                                           $BASEDIRNAME/$BASEFILENAME.jpg sourceforge/html/pictures/$BASEFILENAME.jpg
      convert -verbose -resize 800x600                           $BASEDIRNAME/$BASEFILENAME.jpg sourceforge/html/pictures/$BASEFILENAME-half.jpg
      convert -verbose -size 240x240 -resize 240x240             $BASEDIRNAME/$BASEFILENAME.jpg sourceforge/html/pictures/$BASEFILENAME-thumbnail.jpg
      convert -verbose -colors 256 -density 75x75 -compress JPEG $BASEDIRNAME/$BASEFILENAME.jpg documentation/latex/pictures/$BASEFILENAME.eps
    done
}

function UpdateSchematics(){
    echo "Updating pictures"

    BASEDIRNAME[1]="hardware/adaptor_board"
    BASEDIRNAME[2]="hardware/analog_input_board"
    BASEDIRNAME[3]="hardware/analog_output_board"
    BASEDIRNAME[4]="hardware/digital_input_board"
    BASEDIRNAME[5]="hardware/digital_output_board"
    BASEDIRNAME[6]="hardware/monitoring_and_power_board"
    BASEDIRNAME[7]="hardware/monitoring_and_power_board"

    BASEFILENAME[1]="ab-schematic"
    BASEFILENAME[2]="aib-schematic"
    BASEFILENAME[3]="aob-schematic"
    BASEFILENAME[4]="dib-schematic"
    BASEFILENAME[5]="dob-schematic"
    BASEFILENAME[6]="mpb-schematic"
    BASEFILENAME[7]="power_cable"

    for index in 1 2 3 4 5 6 7
    do
      echo ""
      echo ""
      echo ""
      echo "Managing ${BASEDIRNAME[index]} \ ${BASEFILENAME[index]}"
      echo ""
      BASEDIRNAME="${BASEDIRNAME[index]}"
      BASEFILENAME="${BASEFILENAME[index]}"
      convert -verbose -colors 4 -compress BZip +antialias -density 150x150 +dither $BASEDIRNAME/$BASEFILENAME.png documentation/latex/pictures/$BASEFILENAME.eps
      convert -verbose -size 240x240 -resize 240x240 $BASEDIRNAME/$BASEFILENAME.png sourceforge/html/pictures/$BASEFILENAME-thumbnail.jpg
      convert -verbose $BASEDIRNAME/$BASEFILENAME.png sourceforge/html/pictures/$BASEFILENAME.jpg
    done
}

function UpdateCircuitDescriptionSchematics(){
    echo "Updating pictures"

    BASEFILENAME[1]="adress-comparator"
    BASEFILENAME[2]="board-selection"
    BASEFILENAME[3]="bus-input"
    BASEFILENAME[4]="bus-output"
    BASEFILENAME[5]="current-to-voltage"
    BASEFILENAME[6]="da-converter"
    BASEFILENAME[7]="led-driver"
    BASEFILENAME[8]="opto-coupled-input"
    BASEFILENAME[9]="surge-protection"

    for index in 1 2 3 4 5 6 7 8 9
    do
      echo ""
      echo ""
      echo ""
      echo "Managing hardware/circuit-description \ ${BASEFILENAME[index]}"
      echo ""
      BASEDIRNAME="hardware/circuit-description"
      BASEFILENAME="${BASEFILENAME[index]}"
      convert -verbose -colors 4 -compress BZip +antialias -density 150x150 +dither $BASEDIRNAME/$BASEFILENAME.png documentation/latex/pictures/$BASEFILENAME.eps
    done
}

function UpdateCalculationIllustrations(){
    echo "Updating calculation illustrations"
    rsync -vut hardware/calculation_illustrations/*eps documentation/latex/calculations/illustrations/
}

function UpdateBlockDiagrams(){
    echo "Updating block diagrams"
    rsync -vut documentation/illustrations/*eps documentation/latex/pictures/
    rsync -vut documentation/software/gui/*eps documentation/latex/pictures/
}

while [ "$choice" != "q" ]
do
  $DIALOG --clear --title "Rapier project" \
        --menu "Choose an action:" 49 130 44 \
        "Change upload target"               "Currently uploading to: $WEBTARGET." \
        "Everything"                         "Do everything. Takes a very long time." \
        "DocumentationUploadBackup"          "Updates documentation, upload documentation, backup everything." \
        "Illustrations"                      "Update all illustrations." \
        "Illustrations->Photos"              "Update photos into docs and html." \
        "Illustrations->Schematics"          "Update schematics into docs and html." \
"Illustrations->CircuitDescriptions" "Update circuit description schematics into docs" \
        "Illustrations->Calculations"        "Update Calculation Illustrations into docs." \
        "Illustrations->BlockDiagrams"       "Update Block Diagrams into docs." \
        "Documentation"                      "All jobs on documentation." \
        "Documentation->Partlists"           "Copy partlists into documentation(latex)." \
        "Documentation->Tar Latex"           "Creates a tar ball of latex." \
        "Documentation->Latex"               "Compile Latex." \
        "Documentation->DVI->PDF"            "Compile latex to PDF." \
        "Documentation->Online"              "Build Online Documentation from PDF." \
        "Documentation->Z8Code"              "Updating Z8 code documentation" \
        "Documentation->GUICode"             "Updating GUI code documentation" \
        "SourceCode"                         "All jobs on Source Code." \
        "SourceCode->Z8"                     "Z8 Source Code." \
        "SourceCode->GUI"                    "GUI Source Code." \
        "Tools"                              "All jobs on tools." \
        "Tools->Rapierproject"               "Collects Rapier Project software" \
        "Tools->itctools"                    "Collects itcutils software" \
        "Tools->LatexTools"                  "Collects latex tools software" \
        "Upload"                             "All upload jobs." \
        "Upload->HTML"                       "Upload html to sourceforge.net." \
        "Upload->Online Documentation"       "Upload Online Documentation to sourceforge.net." \
        "Upload->Latex Documentation"        "Upload Latex Documentation to sourceforge.net." \
        "Upload->PDF Documentation"          "Upload PDF Documentation to sourceforge.net." \
        "Upload->Tools"                      "Upload tools to sourceforge.net." \
        "Upload->SourceCode"                 "Upload Source Code to sourceforge.net." \
        "Upload->ApplicationNotes"           "Upload Application Notes to sourceforge.net." \
        "Backup"                             "All backup jobs." \
        "Backup->Software"                   "Backup all software." \
        "Backup->Source Forge"               "Backup Source Forge." \
        "Backup->Documentation"              "Backup Documentation." \
        "Backup->Hardware"                   "Backup Hardware." \
        "Backup->Management"                 "Backup Management." \
        "Backup->Tools"                      "Backup Tools." \
        2> $tempfile

  retval=$?

  choice=‘cat $tempfile‘

  case $retval in
    0)
#######################################################
      if [ "$choice" == "Change upload target" ]
      then
        toggletarget
      fi
#######################################################
```

```
      if [ "$choice" == "Illustrations->Photos" -o "$choice" == "Everything" -o "$choice" == "Illustrations" ]
      then
        UpdatePhotos
      fi
      if [ "$choice" == "Illustrations->Schematics" -o "$choice" == "Everything" -o "$choice" == "Illustrations" ]
      then
        UpdateSchematics
      fi
      if [ "$choice" == "Illustrations->CircuitDescriptions" -o "$choice" == "Everything" -o "$choice" == "Illustrations" ]
      then
        UpdateCircuitDescriptionSchematics
      fi
      if [ "$choice" == "Illustrations->Calculations" -o "$choice" == "Everything" -o "$choice" == "Illustrations" ]
      then
        UpdateCalculationIllustrations
      fi
      if [ "$choice" == "Illustrations->BlockDiagrams" -o "$choice" == "Everything" -o "$choice" == "Illustrations" ]
      then
        UpdateBlockDiagrams
      fi
########################################################
      if [ "$choice" == "Documentation->Partlists" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        CopyPartlists
      fi
      if [ "$choice" == "Documentation->Latex" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        CompileLatex
      fi
      if [ "$choice" == "Documentation->DVI->PDF" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        DVI2PDF
      fi
      if [ "$choice" == "Documentation->Online" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BuildOnlineDocumentationFromPDF
      fi
      if [ "$choice" == "Documentation->Tar Latex" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        MakeTarlatex
      fi
      if [ "$choice" == "Documentation->Z8Code" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UpdateZ8Documentation
      fi
      if [ "$choice" == "Documentation->GUICode" -o "$choice" == "Everything" -o "$choice" == "Documentation" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UpdateGUIDocumentation
      fi
########################################################
      if [ "$choice" == "SourceCode->Z8" -o "$choice" == "Everything" -o "$choice" == "SourceCode" ]
      then
        Z8code
      fi
      if [ "$choice" == "SourceCode->GUI" -o "$choice" == "Everything" -o "$choice" == "SourceCode" ]
      then
        GUIcode
      fi
########################################################
      if [ "$choice" == "Tools->Rapierproject" -o "$choice" == "Everything" -o "$choice" == "Tools" ]
      then
        Getrapierproject
      fi
      if [ "$choice" == "Tools->itctools" -o "$choice" == "Everything" -o "$choice" == "Tools" ]
      then
        Getitctools
      fi
      if [ "$choice" == "Tools->LatexTools" -o "$choice" == "Everything" -o "$choice" == "Tools" ]
      then
        MakeLatexTools
      fi
########################################################
      if [ "$choice" == "Upload->HTML" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadHomepage
      fi
      if [ "$choice" == "Upload->Online Documentation" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadOnlineDocumentation
      fi
      if [ "$choice" == "Upload->Latex Documentation" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadLatexDocumentation
      fi
      if [ "$choice" == "Upload->PDF Documentation" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadPdfDocumentation
      fi
      if [ "$choice" == "Upload->Tools" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadTools
      fi
      if [ "$choice" == "Upload->SourceCode" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadSourceCode
      fi
      if [ "$choice" == "Upload->ApplicationNotes" -o "$choice" == "Everything" -o "$choice" == "Upload" -o "$choice" == "DocumentationUploadBackup" ]
      then
        UploadApplicationNotes
      fi

#######################################################
      if [ "$choice" == "Backup->Software" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupSoftware
      fi
      if [ "$choice" == "Backup->Source Forge" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupSourceForge
      fi
      if [ "$choice" == "Backup->Documentation" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupDocumentation
      fi
      if [ "$choice" == "Backup->Hardware" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupHardware
      fi
      if [ "$choice" == "Backup->Management" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupManagement
      fi
      if [ "$choice" == "Backup->Tools" -o "$choice" == "Everything" -o "$choice" == "Backup" -o "$choice" == "DocumentationUploadBackup" ]
      then
        BackupTools
      fi


      echo "Jobs done!!"
      sleep 1
      ;;

    1)
      choice="q";;
    255)
      choice="q";;
  esac

done

clear
```

7.10 Unnumbered pages

The last part of the project is pages copied from various sources. These pages are not numbered.