USB enabled PDP8 computer

Johan van de Pol, Martin Mul, Georgi Gaydadjiev and Stamatis Vassiliadis
Computer Engineering Laboratory,
Electrical Engineering, Mathematics and Computer Science Department,
Delft University of Technology,
Delft, The Netherlands
{jvandepol, mpmul, georgi}@Dutepp0.ET.TUDelft.NL
http://ce.et.tudelft.nl

Abstract— **The PDP-8 family of minicomputers was built by Digital Equipment Corporation between 1965 and 1990. The PDP8 computers were one of the first computers that were affordable by a broader range of customers; that contributed in the success of these machines. In 1976, Intersil developed a PDP8 chipset consisting of the IM6100 processor [4], the I/O extension (IM6101) and the UART (IM6402). The PDP8 utilizes a small instruction set. Only eight basic instructions are implemented, which provide enough functionality to compose complete programs. Since there are many PDP8 legacy programs available on paper and magnetic tapes and not that many operational PDP8 machines left, the PDP8 project was initiated. The Intersil chipset is used together with a 80C32 microcomputer in a single board device. The PDP8 device enables the usage of a real PDP8 computer as in the days around 1970, including native I/O connectivity. In addition, the PDP8 device is controlled by a host computer. Original PDP8 peripherals such as teleprinters and magnetic tape readers can be connected to the PDP8 device. In this way original PDP8 programs stored on paper or magnetic tapes can be loaded and executed into the PDP8 device. The 80C32 has control over the memory and the system state of the PDP8 processor. The 80C32 communicates with the host computer via USB connection. The USB connection is implemented by the PDIUSB12 controller from Philips. The whole PDP8 device is under control of a dedicated application running on a Microsoft Windows host computer. A command/response interface between the host and the PDP8 device was developed for this purpose and implemented. This article describes the development process and the design of the PDP8 device hardware, the device software and the host software.**

I. Introduction

The PDP8 family of microcomputers were built by Digital Equipment Corporation. These computers have a historical value, because they were the first computers that were affordable by a broader range of customers. The PDP8 processors were quite special because of their small instruction set: only eight basic instructions were supported. These eight basic instructions are sufficient to provide enough functionality to compose complete applications. Due to its historical value and the fact that most of the PDP8 computers are no longer operational, a PDP8 computer that can be controlled from a host computer is designed and created. In particular, the host computer can control and manipulate the memory and the state of the PDP8 computer. This article describes the design of this system.

The main contributions of the work described hereafter are: 1) The specification and the design of the PDP8 device hardware. 2) The definition and the development of the software for both the PDP8 device and the host computer. 3) The set up and implementation of dedicated command/responce communication protocol for data exchange between PDP8 device and the host.

This article is arranged as follows. A small historical overview of the PDP8 machine is given in Section II. Next, the system specifications are presented in Section III. The PDP8 hardware device is described in Section IV and Section V will describe the software for the host application controlling the PDP8 device and the device software. Finally, in Section VI a summary of this article is given.

II. PDP history

In 1957, Ken Olson and Harlan Anderson founded Digital Equipment Corporation (DEC), capitalized at $100.000, and 70 percent owned by American Research and Development Corporation [7]. They wanted to call their company Digital Computer Corporation, but the venture capitalists insisted that they avoid the term Computer and hold off on building computers, because of the stereotype that computers were big and expensive, needing a computer center and a large staff.

DEC's first computer, the PDP1 had a price label of only $120.000 at a time when other computers sold for over $1.000.000. In addition, DEC quoted prices as low as $85.000 for minimal models. The term computer was also avoided by using the term Programmable Data Processor or PDP for their products. For over a decade, all digital computers sold by DEC were called PDP's.

The PDP8 family of minicomputers was built by Digital Equipment Corporation between 1965 and 1990. The PDP8 was largely upward compatible with the PDP5, a machine that was unveiled in 1963. All of these machines were characterized by a 12 bit word, typically 4K words of memory, and simple but powerful instruction sets.

By late 1973, the PDP8 family was the best selling computer in the world. Most models of the PDP8 set new records as the least expensive computer on the market at the time of their introduction. The PDP8 has been described as the model-T of the computer industry because it was the first computer to be mass produced at a cost that just about anyone could afford.

In 1976 Intersil, offered the first PDP8 processor to occupy a single chip using CMOS technology: the IM6100. DEC verified that it was a PDP8 and began to apply it to its products in the fall of 1976 [4].

III. PDP8 DEVICE SPECIFICATIONS

Before starting with prototyping a system (hardware/software), a clear specification of the system is required. This specification is needed to be able to check the design at the end of the whole design process. This section describes the specification of the PDP8 device hardware and the host and PDP8 device software.

The project requirements where as follows: 1) The PDP8 device in this project will be a small device, in contrast to the early days, where the PDP8 computers were reasonable big machines. In addition it should be easy to install and set up. 2) The PDP8 device should be controlled by a host computer. This means that the host computer should be able to read and write the entire PDP8 memory and should be able to start, stop, step and reset the PDP8 processor. 3) Legacy I/O devices, such as teletype printers and tape readers, should be able to communicate with the PDP8 device. 4) There should be reasonable communication speed. Transferring the complete memory content of the PDP8 to (or from) the host computer should not take more than 500 ms. These requirements will be translated in hardware and software specifications here after.

A. Hardware specifications

The PDP8 device should be a small device and connecting the device to a host computer is envisioned to be easy. In this way the PDP8 device will be usable with a range of different computers. The PDP8 device should at least contain the following components: PDP8 processor, Memory for the PDP8, I/O extension and an interface to communicate with the host.

Considering the connection between the host computer and the PDP8 device to transfer data the following options were considered: 1) PCI bus; 2) Parallel connection; 3) Serial connection; 4) USB connection. Because the device should be easy to install, the first option is abandoned quickly. Installing a PCI card takes a significant time compared with inserting a connector. The options left are all options where a cable is used. Almost every modern computer has one or more USB ports. Considering that not all computers have a free serial or parallel port any longer and that USB is faster than a serial or parallel connection, an USB [9] connection is chosen.

B. Software specifications

The design specification gives a list of items the final system should comply to. This list of specifications is summarized hereafter. The host and the device software must be able to: 1) Start the PDP8 processor. 2) Stop the PDP8 processor. 3) Step the PDP8 processor. 4) Reset the PDP8 processor. 5) Set the switch register of the PDP8 computer. 6) Read the value of the switch register and display this value. 7) Read status information of the PDP8 processor. 8) Write data to a specified location of the PDP8 RAM. 9) Read data from a specified location of the PDP8 RAM. The read data must be displayed in an appropriate way. 10) Store data from the pdp8 RAM in a binary file. This file must be in a common format. 11) Load a binary file in a common format into the RAM of the PDP8 processor. 12) Provide an editor to edit PAL assembly files. 13) Be able to load a PAL file into the editor. 14) Be able to save the editor text into a pal file. 15) Be able to compile the text in the editor and show the error messages of the compiler. 16) Be able to write the binary output of the compiler to the RAM of the PDP8 processor. In addition to the support of the above issues the device software should be able to: 1) Configure the 80C32. 2) Setup the PDIUSB12. 3) Initialize the PDP8 sub-system. 4) Handle reception of incoming USB packets from the host: A) handle reception of predefined packets (these packets will be introduced in Section V); and B) handle reception of any undefined packets (error handling).

IV. PDP8 HARDWARE DEVICE

The main part of the PDP8 hardware device is the PDP8 computer [6]. To provide control over this computer, the device is connected to a host computer using an USB interface as required by the specifications. A microcomputer system, using a 80C32 processor, takes care of the interface between the host and the PDP8 computer. The RAM of the PDP8 must also be available for the 80C32 processor, but the 80C32 and the PDP8 processor both run independent of each other. They don't share a data and address bus, which means that some kind of multiplexer for

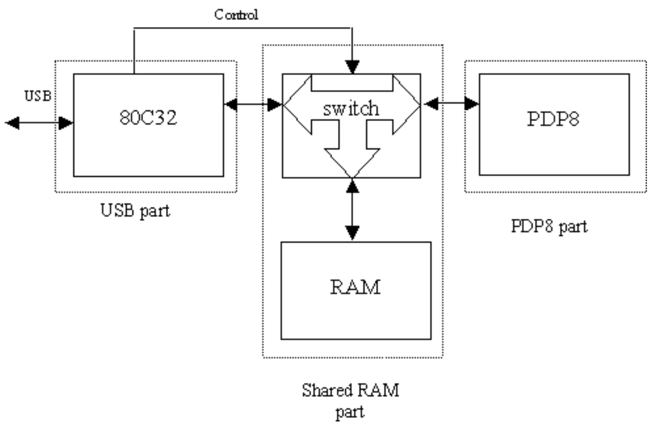

Fig. 1. PDP8 device hardware overview

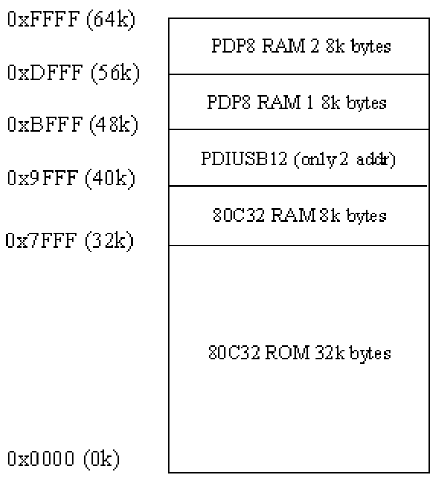

Fig. 2. The distribution of the available memory space

switching the RAM is needed. The hardware design is split into three parts: the USB part, the shared RAM part and the PDP8 part. An overview of the hardware as planned is given in Figure 1. The remaining of this section describes these three parts.

A. PDP8 part

The PDP8 system must be a complete basic system of a PDP8 machine as it was used in earlier times. This means that the system has to contain at least the following components: PDP8 Processor, 4k words memory space, Parallel Interface Element (PIE) and UART.

The switch register that these machines contained needs also to be implemented and the PDP8 computer should function as it did in the past to have compatibility. The switch register is a register that was build out of twelve switches, which were used to set the value of this particular register.

The PDP8 is a 12-bit processor. Both the data bus and the address bus are 12-bit. Memory modules with this size are not widely available if they exist at all. Therefore, two 8kx8 memory modules are used to create a memory space of 8kx16. The PDP8 however can only address 4kx12 of this memory. Section IV-C contains more details about the memory of the PDP8.

B. USB part

The 80C32 processor is a ROM-less device containing:

- 256 x 8 RAM;
- 32 general purpose I/O lines;
- three 16-bit counter/timers;
- six-source, four-priority level nested interrupt structure.

The 64k of memory addresses available are divided between the following components:

- 32k x 8 EEPROM for the 80C32: In the EEPROM the device software is stored. Section V-B will present the device software. This will be the largest amount of data, so the bottom half of the available space is allocated to the EEPROM as shown in Figure 2.
- RAM for the 80C32: The external RAM for 80C32 occupies the memory space from 32k to 40k, as shown in figure 2.
- RAM for the PDP8: Two 8k x 8 bit RAM devices are used. Section IV-C will describe the memory of the PDP8 in more detail. As shown in Figure 2, the first RAM (RAM1) is in the memory space from 48k-56k and the second RAM (RAM2) from 56k-64k.
- PDIUSB12: Two location are needed for the PDIUSB12. One is needed to write command instructions and another is needed to read and write data. As shown in Figure 2, the PDIUSB12 takes two memory addresses in the memory space between 40k and 48k.

In Figure 3 the timing of the 80C32, IM6100, RAM and PDIUSB12 is shown. The timing of the ROM is equal to the timing of the read part of the RAM. The 8 least significant bits of the address output of 80C32 are latched on the negative edge of *ALE*. The address bus of IM6100 is latched on the negative edge of *LXMAR*. The address will be available until the end of the read or write cycle. The control signals for the memory devices of the 80C32 are: *chip enable* for the ROM, RAM and PDIUSB12 is created by external logic from the address bus, *write enable* for the RAM and PDIUSB12 is the *Write* output of the 80C32 and *output enable* of the ROM, RAM and PDIUSB12 is the output of the AND port with the inputs *PSEN* and *Read* of the 80C32. The control signals for the RAM of the PDP8 are: *chip enable* for the RAM is the *MEMSEL* output of the IM6100, *output enable* is the inverse of the *XTA* output and *write enable* is the *XTC* output.

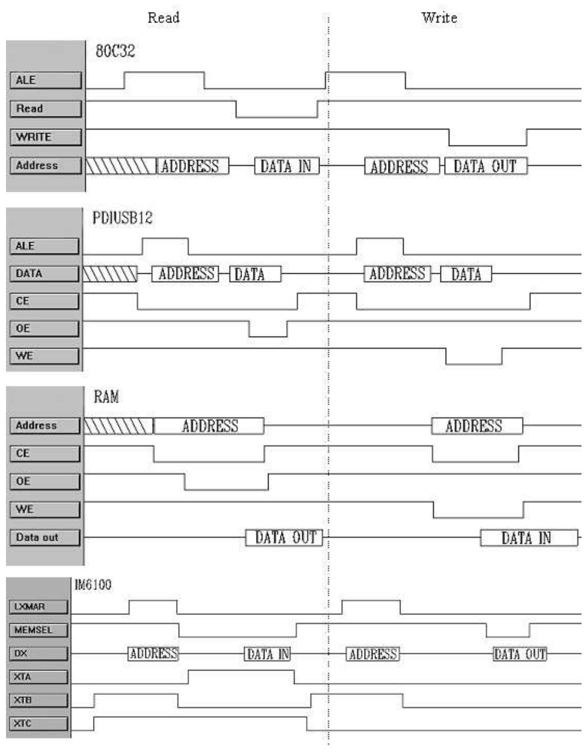

Fig. 3. Timing diagram

Timer 2 of the 80C32 is used as an oscillator. The output of this oscillator is connected to I/O pin 1.0. This pin is connected to the clock input of the UART of the PDP8 (IM6402). The baudrate of the UART should be equal to 110 bps. This is the baudrate used in communication with external devices, such as teleprinters and tape readers. The baudrate of the IM6402 is 1/16 of the clock input, therefore the clock input should be 1,76 kHz.

The USB connection is implemented by the PDIUSB12 controller from Philips [8]. The PDIUSB12 uses the full speed (12 Mbits/s) mode of USB version 1.1. The PDIUSB12 takes two addresses from the memory space of the 80C32. One address for data and one for commands. Three endpoints are supplied by the PDIUSB12. Endpoints can be described as sources and sinks of data. Because a device can have more then one endpoint, several data channels can be set up between a device and host. The PDP8 device uses two endpoints, one for communicating with the driver in Microsoft Windows XP and one to communicate with the host software. A USB driver is needed at the host to communicate with an USB device. In Microsoft Windows XP, several USB class drivers exist. Writing a device driver takes a lot of time in comparison with creating a few descriptors [5], that are needed for a standard driver. Descriptors describe the device for the host. The descriptors give information such as the USB version used, number of endpoints and type of endpoints. The PDP8 device uses the Human Interface Device (HID) class. A HID transfers data in reports. Reports are predefined data blocks with controllable size.

The PDIUSB12 has a programmable clock out pin, the clock output on this pin can have a frequency between 4MHz and 48MHz. The frequency depends on the value written after a *set mode* command from the 80C32. The clock output is connected to the clock input of the 80C32. The maximal operational frequency of the 80C32 used, is 16 MHz. This frequency is used as the clock output frequency.

C. Shared RAM part

The RAM on the board has to be available for both the C32 and PDP8 processor. For the C32, this RAM is just some RAM in the memory space; the C32 does not need this RAM for its normal operation. For the PDP8 however, this RAM is the main memory and it also replaces the ROM normally found in these computers. The PDP8 and the C32 both have to access the RAM, but a few differences exist between these processors. These differences consist of different data and address bus width and a small difference in timing.

The PDP8 is capable of addressing 4k words of memory, because the processor has only 12 address lines. The data bus of the PDP8 is 12 bits wide and RAM chips with this format are not available. This means that several RAM chips must be used to build the RAM for the PDP8. Chips containing 4k words are not widely available, so two chips with 8k bytes are used. This gives a memory of 8k x 16 bit. The 13th address bit of the RAM chips is controlled by the C32. Therefore the C32 can let the PDP8 operate on the top or bottom half of the memory.

Because both processors must have access to the same RAM, some logic is needed for these chips. Only one processor can have access to the RAM at the same time, which processor has control over the RAM is controlled by the C32 processor. The RAM switch that has to be designed must switch between the data bus, address bus and all timing and other signals that control the RAM. To select different inputs, a simple multiplexer is needed. For the data lines a simple multiplexer will not be sufficient, because data flows in both directions. Therefore some transmission gates are used for switching the data lines of the RAM. These transmission gates are tri-stated, which means that, when not enabled, the connected lines will be high-impedance and not be tight low or high. This is important, because otherwise the data bus is disturbed and wouldn't function. The C32 and PDP8 processor both have an own set of timing signals. The only difference between those sets is that the C32 computer provides two chip select signals for the two RAM chips and the PDP8 one, because the PDP8 will always select both RAM chips at the same time. Together with the read and write signal, this gives four signals per processor. These two sets of signals are fed into a multiplexer. One of these sets is selected by the switching signal, which is also used by the enabling and disabling of the transmission gates. Also the 12 address bits are fed into multiplexers and the switching signal selects one of the addresses.

V. Software

A. Host software

The host software provides the user control over the PDP8 processor and its memory. Therefore the host software has to communicate with the software in the embedded system. This is realized by an USB connection. The USB protocol is incorporated into the operating system Microsoft Windows [3], which means that the host software uses USB functions included in the operating system and no additional drivers need to be designed. The standard USB drivers for a Human Interface Device (HID) are used. The Host software is written in C++ using Borland C++ Builder [2], because this package provides an easy way of creating graphical user interfaces using standard widgets. The host software also provides some extra controls to edit and cross compile PAL assembly files. The binary output of the compiler can be uploaded to the memory of the PDP8 computer.

B. Device software

The device software takes care of the configuration of the 80C32, PDP8 and the PDIUSB12 and handles the incoming packages on the PDIUSB12. Before the host program can send data packets to the device, an USB connection has to be established. When a USB plug is connected to a computer running Microsoft Windows, Microsoft Windows will try to enumerate the device. Enumeration is the process of determining what device has just been connected to the bus and what parameters it requires such as power consumption, number and type of endpoint(s), class of product etc. The host will then assign the device an address and enable a configuration allowing the device to transfer data on the bus. After the enumeration is completed, the host software can read and write packets of data from and to the device.

C. Commands from and to the host software

The commands for the communication between the host software and the PDP8 device were additionally created. These commands enter the PDP8 device in endpoint *two out*. The host and device communicate with each other by using reports with a length of 64 bytes. If bigger reports are used, they will be send in separate packages. The first byte of the messages is a message header. A message header is a single byte, enabling 255 different messages. The following 63 bytes depend on the message header. Most of the time the bytes will contain first some arguments and then some data. In more detail the messages are as follows:

- Read RAM: This message contains a start address and length. The start address is the address of the first byte that should be send to the host. The length is the total amount of data that should be send. The reply on this message is the content of the requested part of the RAM. After sending the requested RAM content is completed, a CRC is calculated over the data and send to the host with the *CRC after read* command. The host will calculate the CRC over the data of the RAM part and compare it with the CRC that is received. In case the two CRC's are different, the same part of the RAM will be requested again. The CRC that is used is the CRC-16 with the polynomial $x^{16}+x^{15}+x^2+1$.
- Write RAM: This message contains a start address and a number of 12 bit data words that will be send in this message.
- CRC of data: A data write is always followed by a CRC check to verify if the data is written correctly. This CRC check contains the start address, length and the CRC of the data that the host calculated. This is send back extended with the CRC that is calculated by the 80C32 with the CRC after write message.
- Write switch register: This messages contains 12 bits data that has to be written into the switch register.
- Read switch register: The switch register can not directly be read by the 80C32. Therefore a copy is maintained in the RAM of the 80C32. The only way to change the value of the switch register is by a write switch register command. When there is a write switch register command, the value is updated in the RAM.
- Start PDP8: This message will start the PDP8 processor if the processor is stopped. This is done with a pulse on the run/hlt input of the IM6100. If the processor already runs, no action is taken.
- Stop PDP8: This message will stop the PDP8 processor if the processor is running. This is done with a pulse on the run/hlt input of the IM6100. The processor will finish the instruction it was executing and increase the program counter, before it stops. If the processor was already stopped, no action is taken.
- Reset PDP8: This message will initiate a reset for the PDP8 chips. A pulse similar to the power-on reset will be given to the reset inputs.

• Step PDP8: This message let the PDP8 execute one instruction if the PDP8 processor was stopped. This is done by a pulse on the run/hlt input of the IM6100, followed by another pulse on this pin. If the PDP8 processor runs when this command is received, no action is taken.

• Read PDP8 status: When this command is received at the PDP8 device the PDP8 device should send a message back with the content of PDP8 registers and indication bits. These are the MQ register, accumulator, link bit, program counter and content of the memory at the position of the program counter. To load the values of the registers and indication bits a special program has to be transferred to the PDP8 RAM. Therefore the original PDP8 program has to be stopped and the 13th address bit should be switched to one. Now the bottom half of the RAM can be written. When the bottom half of the PDP8 RAM is not used, the program can be stored there. When the bottom half is used, the status command can not be used, otherwise it will overwrite the original program. There are PDP8 instructions to load the content of the MQ register, accumulator and link bit. An instruction to load the program counter is not provided. After the original program is stopped, the following actions have to be taken to load the program counter and registers: 1) Set 13th address bit to one, the bottom half of the RAM is selected. 2) The bottom half of the RAM is totally filled with jump to zero instructions. 3) One step is executed. When the jump to zero instruction is executed, the program counter will be one and at address zero the return address is written. The return address is equal to the address of the last executed jump instruction. This address is equal to the program counter. 4) In the bottom half of the PDP8 RAM, a PDP8 program is stored at address 1. This program contains instructions to save the content of the registers and indication bits. 5) The program is stopped with a halt instruction. When the program is stopped, one step is executed. This step is a jump instruction to the address that is stored in address zero, this is the value of the original program counter. In such way the original program counter is restored. 6) The 13th address bit is set to zero. The original program can be continued.

• Send acknowledge: When the host software wants to read the buffer for incoming messages it will use the Windows function Readfile. This function will not return until data is read from the USB. Therefore before reading the USB device, a message is send to the device. This is the send acknowledge message, it returns a report with only the header send acknowledge. In this way it is assured that when the host reads the PDP8 device, there is always a message in the buffer.

• Error message: A string of maximum 63 bytes including a '\0' to terminate the string is send to the host. The host will display this string to the user.

• Debug message: A string of maximum 63 bytes including a '\0' to terminate the string is send to the host. This string was used during the development the software.

VI. Summary

This article presented the design of the PDP8 device. This PDP8 device is controllable from a host computer via an USB connection. Data can be written in the switch register and the RAM of the PDP8. External devices such as a teletype printer can be connected to the PDP8 device. Programs stored on paper tape can be read into the PDP8 memory and stored on the host computer. In this way original PDP8 programs can run on the PDP8 device. In addition memory dumps of those programs can be made to preserve them on more reliable information carriers, e.g. CD roms for the future.

The operating system OS 8, that was used on PDP8 computers could not work on the PDP8 device. The reason was that OS 8 incorporates opcode's that the IM6100 processor does not support. To allow OS 8 to operate on the PDP8 device a different Intersil processor (6120) is needed. Using the tape reader of a teletype printer, we loaded a Trac compiler from a paper tape into the PDP8 device. We couldn't execute this compiler, because the compiler uses some additional I/O codes. Further work is required, to find out exactly for what I/O devices these codes are. To connect extra I/O devices to the PDP8 device additional I/O extension chips (IM6101) and UART (IM6402) should be connected to the PDP8 processor. This will require PCB redesign however. We only have saved the compiler on the host computer, and executed it inside a PDP8 simulator [1]. On this simulator the Trac compiler worked as expected.

References

[1] Bernhard Baehr. *PDP-8/e simulator*. Website: http://home.t-online.de/home/bernhard.baehr/pdp8e.html, 2004.

[2] Borland. *Borland: Leading Provider of Technology for Software Applications*. Website: http://www.borland.com, 2004.

[3] Microsoft Corporation. *Microsoft Corporation*. Website: http://www.microsoft.com, 2004.

[4] J. Craig Mudge Gordon C. Bell and John E. McNamara. *Computer Engineering, a DEC view of hardware system design*. Digital Press, 1978.

[5] John Hyde. *USB Design By Example*. Intel Press, 2001.

[6] Intersil. *Datasheet: IM6100 CMOS 12 bit microprocessor*. Intersil Inc., 1979.

[7] Douglas W. Jones. *PDP-8 Frequently Asked Questions*. Website: http://www.faqs.org/faqs/dec-faq/pdp8, 2001.

[8] Philips. *Datasheet: PDIUSB12*. Philips, 2001.

[9] USB.org. *USB.org - Welcome*. Website: http://www.usb.org, 2004.