



Fachhochschule Karlsruhe
University of Technology
Department of Sensor Systems Technology



,The Game Boy Group‘

THE GAME BOY PROJECT

T A B L E O F C O N T E N T S

0. Introduction	Page 4
0.1. The Game Boy - only a rusty game console ?	Page 4
0.2. Who are we ?	Page 5
1. What is inside the Game Boy	Page 6
1.1. Technical Details	Page 6
1.2. .Address Overview	Page 9
1.3. Memory Mapping	Page 11
1.4. Cartridge types	Page 11
2. The Project Idea	Page 13
2.1. What is missing in the GB-hardware ?	Page 13
3. Building the hardware	Page 14
3.1. How to connect an Eprom	Page 14
3.1.1. The GB-Adapter	Page 15
3.2. The 8255a PI/O - Chip	Page 17
3.3. How to get the right hardware address	Page 19
3.4. The GB-I/O interface board	Page 20
4. Creating the Software	Page 22
4.1. Software (development) Tools	Page 22
4.1.1. GBBasic	Page 22
4.1.2. GBDK	Page 22
4.1.3. GBDEV Studio	Page 23
4.1.4. NO\$GMB	Page 23
4.1.5. GBTD & GBMB	Page 23
4.2. First steps using GB-Basic	Page 24
4.3. The program	Page 25
5. APPENDIX	Page 26
5.1. References	Page 26
5.2. Downloads	Page 26
5.2.1. Layouts	Page 26
5.2.2. Software tools	Page 27

FIGURE INDEX

Figure 0: the ready etched PI/O-board	Page 3
Figure 1: GB Pocket series	Page 4
Figure 2: 'The Game Boy Group' Mirko, Steffen, Rony, Marc (left to right)	Page 5
Figure 3: General hardware schematic	Page 7
Figure 4: Game Boy Internals (CPU section)	Page 8
Figure 5: address overview	Page 9
Figure 0: more detailed addresses overview	Page 10
Figure 1: Inside a cartridge (Type 10h)	Page 11
Figure 2: Game Boy with adapter & ROM	Page 13
Figure 3: GameLink- and GamePak-connector pinout	Page 14
Figure 4: Wiring diagram for a 32kB EPROM	Page 15
Figure 5: Board layout for the GB-Adapter	Page 16
Figure 6: GB Adapter J1 pinout	Page 16
Figure 7: Block diagram of a 82C55A	Page 17
Figure 8: The control word for port definition	Page 18
Figure 9: Wiring diagram for the device switching logic	Page 19
Figure 10: PI/O-Board J1 pinout	Page 20
Figure 17: GB-I/O interface board layout	Page 21
Figure 18: GBBasic accessing the 82C55A	Page 24
Figure 19: Screen shot of the testing program	Page 25

T A B L E I N D E X

Table 0: Game Boy features	Page 4
Table 1: Technical Details	Page 6
Table 2: Different possible cartridge configurations	Page 12
Table 3: Parts for the GB-I/O interface board	Page 20

ABSTRACT

This document provides detailed constructional information for the realisation of three 8 bit bidirectional parallel ports in a Nintendo¹ Game Boy² cartridge. Only limited hardware knowledge is required, but elementary laboratory equipment is expected. To rebuild the shown cartridges a manufacturing capability for PCBs is required with an programming capability for EPROM or Flash. The job should be done in about less than 15h.

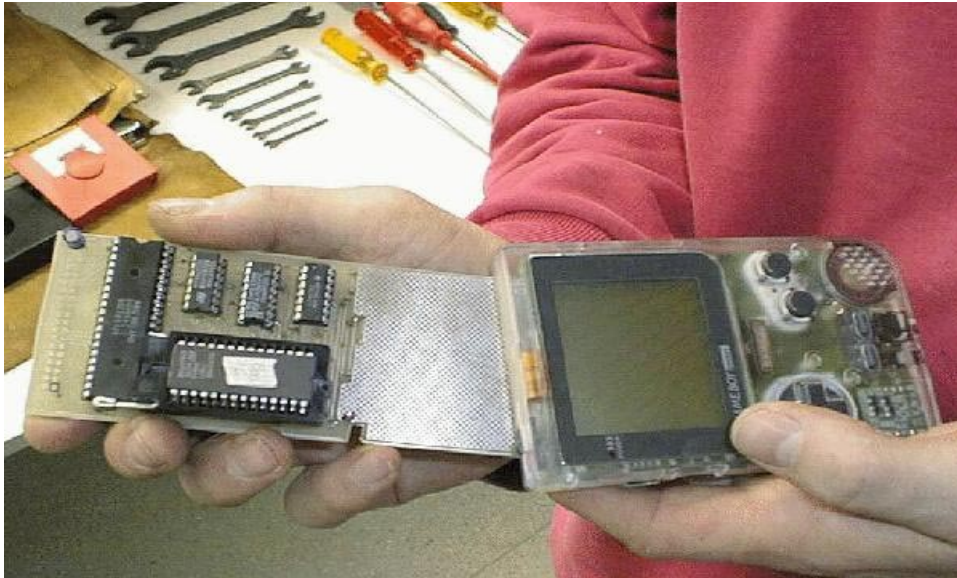


Figure 0: the ready etched PI/O-board

¹ Nintendo is a registered trademark of Nintendo Co., LTD.

² Game Boy and Game Boy color are registered trademarks of Nintendo Co., LTD.

0. INTRODUCTION

This document was written to aid in the development of hardware and software for use with the Nintendo Game Boy . It provides a guide to the known hardware information pertaining to the hand held Game Boy console. Also included is an 8-bit parallel port example project using an Harris™ 82C55A in an add in GB-cartridge. Complete constructional details including PCB layout plans are provided.



Figure 1: GB Pocket series

0.1. THE GAME BOY - ONLY A RUSTY GAME CONSOLE ?

Years ago I received a Nintendo Game Boy as a birthday present and had a lot of fun playing games with it. As the years passed I grew up and forgot about the little console as it gathered dust on a shelf. One day in `97 it caught my eye and seeing it from an engineering viewpoint I realized that it had more potential beyond playing games due to a wealth of attributes and features including the following.

☞ gray scale or color display	☞ good software development tools available as freeware!
☞ Z80 based system	☞ simple keyboard
☞ not that slow (8 MHz)	☞ portable
☞ ready to use	☞ development support by cartridge usage
☞ low cost	☞ stereo sound system
☞ serial port	☞ RAM & ROM support

Table 0: Game Boy features

I realized that this combination of features could form an ideal basis for many applications which would however require additional hardware and the relevant software. I searched the Internet and found further information which reinforced my own feelings and prompted me to try and instigate a project based on the Game Boy as part of my university course. My own enthusiasm was shared by several fellow students and so 'The Game Boy Group' was formed.

0.2. WHO ARE WE ?

All six team members are former fifth Semester students studying Sensor-Systems-Technology at Karlsruhe University of Technology. In the fifth semester all students have the opportunity to work in small groups on an electronics project. After convincing our professor that the basic idea of a project based around the Game Boy was viable, planning started. Several ideas were considered ranging from a small multimeter up to an oscilloscope. In the end it was decided to realize the hardware for an 8 bit parallel port together with the relevant software which would form a basis for future extensions.

The work was divided between the team members as follows:

- searching for information in the WWW: Marc Rawer, Andreas Schmack
- hardware development and build: Marc Rawer, Mirko Smuk
- software development: Steffen Kratochwill
- presentation in the WWW: Marc Rawer, Andreas Schmack
- documentation: Ronny Tomschitz
- manufacturing boards (& playing GB): Uwe Hill
- project supervisor: Prof. Dr. Michael Bantel
- continued support: Marc.Rawer@gmx.de



*Figure 2: 'The Game Boy Group'
Mirko, Steffen, Rony, Marc (left to right)*

1. WHAT IS INSIDE THE GAME BOY

The Game Boy is a portable device with simple operation. The user interface comprises four buttons, a joypad, integrated four color gray scale screen (color with the 'Game Boy Color'³) and a speaker. Stereo sound is available from the headphone jack. The internal hardware includes a Z80 processor derivative together with system RAM and ROM.

These features make the Game Boy ideal for menu controlled applications. Programming a home telephone system for example would be much easier using the Game Boy LCD and joypad than using the telephone pad.

1.1. TECHNICAL DETAILS

The following table provides more detailed technical information on the hardware inside the Game Boy:










 CPU:	8-bit Z80-like CPU running at 4.194304MHz
 BUSES:	8-bit data-BUS, 16-bit address-BUS
 RAM:	8kB/32kB ⁴ internal (CGB, GBP / GBC) also 8kB adressroom for external RAM
 Video RAM:	8kB/16kB internal (CGB, GBP / GBC)
 ROM:	32kByte reserved in addressroom for external ROM
 Sound:	4 channels. Each of which can be mapped either to the left or to the right or to both speakers.
 Video:	Display: Reflective LCD 160x144 dots (physically) Colors: 4 shades of gray (GBC: max. 56 of 32768) Sprites: 40 sprites (8x8 / 8x16)
 Com:	One serial port with 8kbps ⁵
 Power:	Classic: 6 Volts, 0.7 Watts, 4 AA Batteries - 35 hours Pocket: 3 Volts, 0.9 Watts, 2 AAA Batteries - about 25 hours color: 3 Volts, 0.3 Watts, 2 AA Batteries - ??

Table 1: Technical Details

3 Note: GB= Game Boy; CGB= classic GB; GBP= GB-Pocket; GBC= GB Color

4 Note: B= Byte; b= bit; kB= kilo Byte; kb= kilobit

5 Note: kbps= kilo bit per second

Further information can be found in „gbspec.txt [\[1\]](#)“, titled „Everything You Always Wanted To Know About Game Boy but were afraid to ask“ (also known as the '-PAN-/Anthrox'-Document). It is one of the best sources of information for both hardware and software including useful algorithms and common problems/pitfalls.

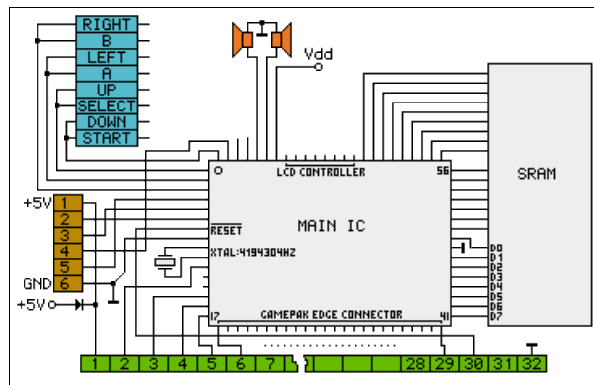


Figure 3: General hardware schematic

Figure 3 shows a schematic of the Game Boy internal hardware. Starting from the left, there are the buttons and joyypad, the serial port, an oscillator, the speakers, the CPU and the main memory. On the lower side the connector to the cartridge is shown.

For further information on the Game Boy internal hardware and logic see the schematic drawn by Jeff Frohwein [\[2\]](#) on the next page.

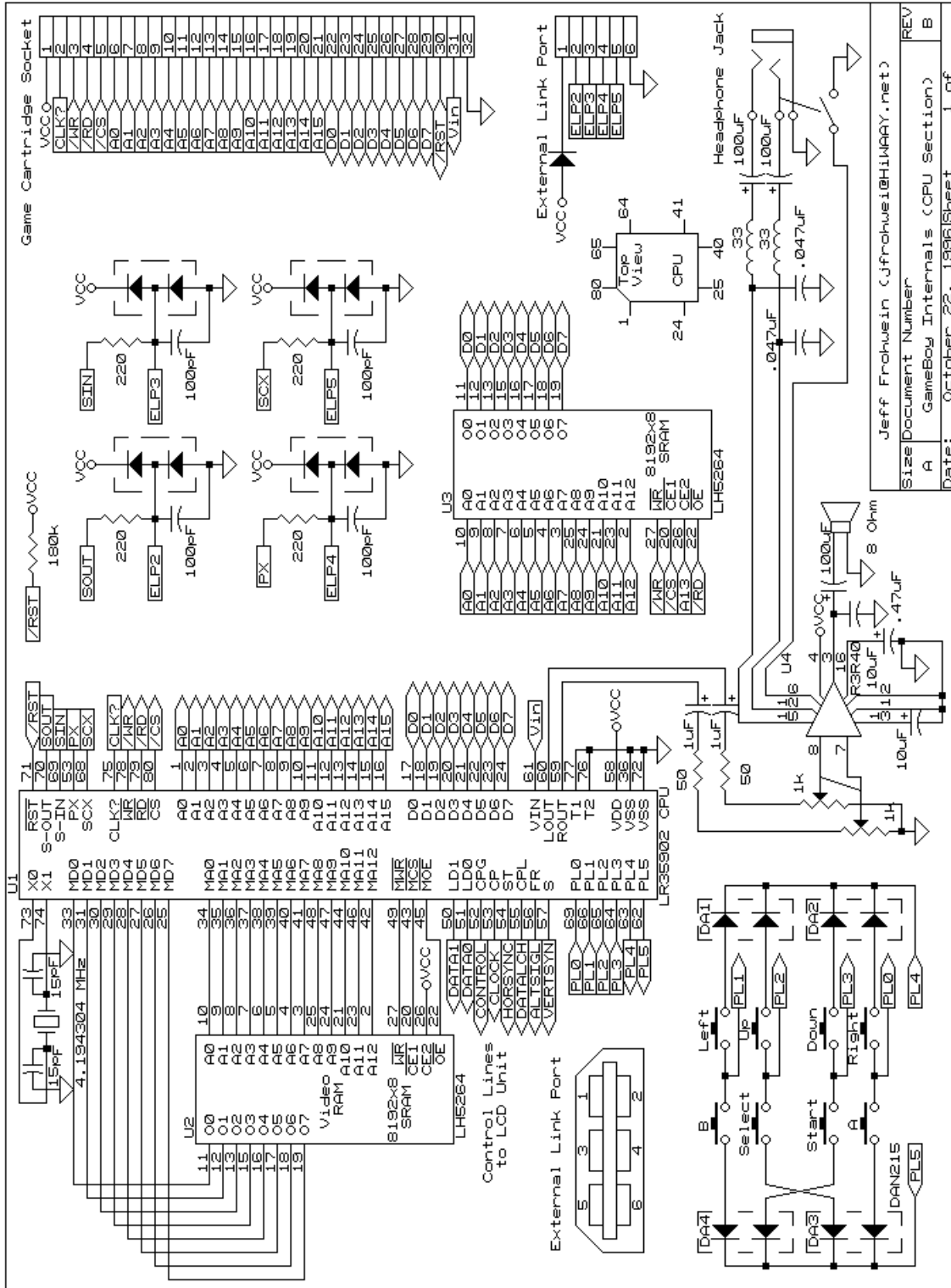


Figure 4: Game Boy Internals (CPU section)

1.2. ADDRESS OVERVIEW

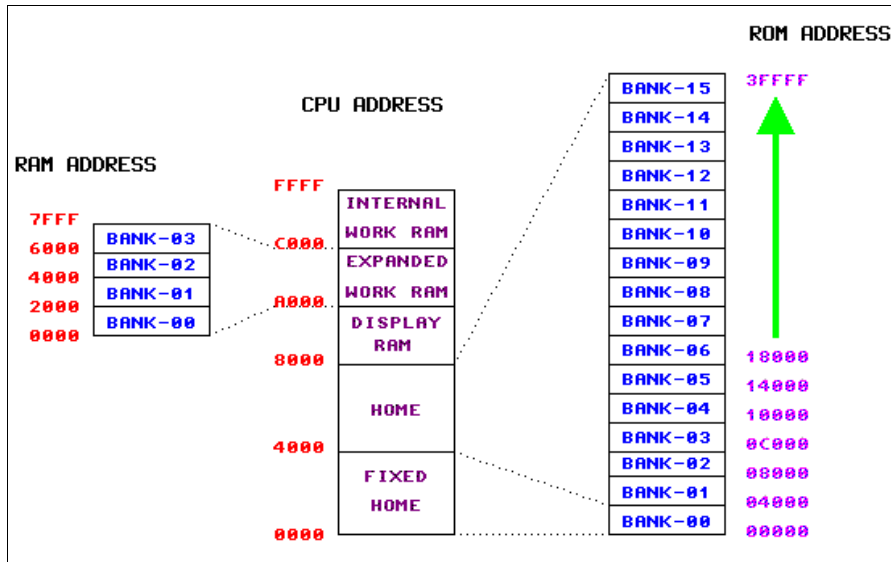


Figure 5: address overview

As described under 1.1 Technical details the Game Boy has a 16-bit address-BUS. Giving an addressable range of 65535 Byte (or 64kB). All hardware components of the Game Boy including RAM, ROM, Video RAM and I/O Ports are memory mapped as shown below.

Figure 5 shows the general definition for hardware devices. The 32K area between 0000h and 7FFFh is the "user program area" where the program ROM is located. The user program area is divided into home & fixed home .

Above this area the video RAM is situated from 8000h to 9FFFh (8kByte). On top of this is the work RAM area from A000h to FFFFh. The address space from A000h to C000h can be used for externally located RAM in a cartridge.

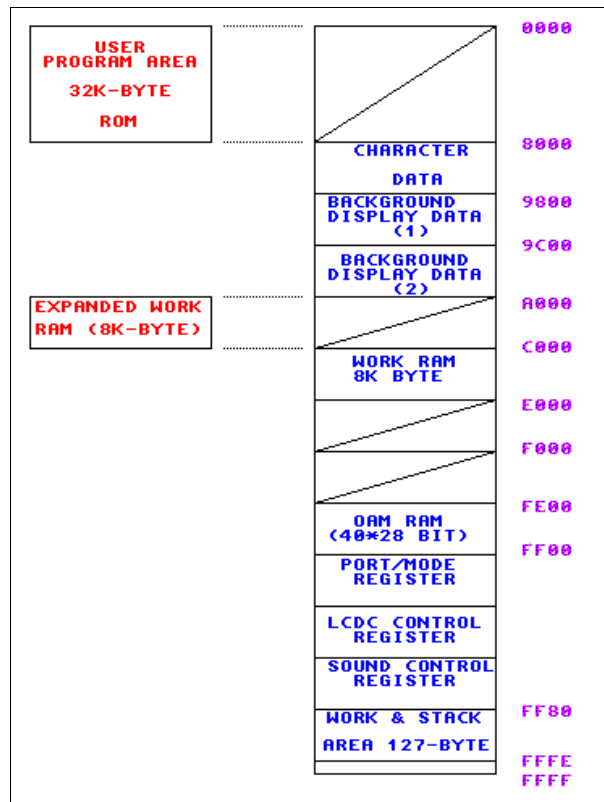


Figure 0: more detailed addresses overview

Figure 0 shows a more detailed overview of the hardware addresses. Further information on address definition can be found in "gbspec.txt [\[1\]](#)" ("Reserved Memory Locations").

The user program area consists of home and fixed home blocks each of these being 16kByte in length. These memory blocks are called "pages" or "banks" (see 1.3 Memory mapping for more information). The '.gb'-files will always be 2^x pages in length as checksumming takes place on a 16kByte block basis.

A user program starts in the 'fixed home' page 00. Small programs in a 16 kByte ROM run entirely in this space. Larger programs of up to 32 kByte using a 32 kByte ROM flow over into the home page 01.

To facilitate programs requiring larger ROMs without causing addressing conflict by going outside of the 32K Byte user program area the memory bank controller (MBC) provides a means of mapping further contiguous 16kByte pages into the home page 01 area.

1.3. MEMORY MAPPING

The basic Game Boy design foresees only 32 kByte of user program area. This is divided into two 16 kByte pages, a *fixed home* page and a *home page*. To allow the use of program ROMs larger than the physical address space a controller is used which "maps" each additional 16 kByte page into the home address space as shown in Figure 5. This process is called 'memory mapping' (or bank switching). It is similar to the method used under DOS when accessing expanded memory. The Z80 has only 16 address bits whereas the ROM has more, depending on its capacity (1 for 64kB, 2 for 128kB and so on). To access a page the page number is loaded into the control register of the MBC which is mapped into the home address space. The MBC will then blend the selected page into the home-area by providing the required address mapping. The memory bank controllers used in Game Boy cartridges also support RAM mapping to the external work RAM location (see Figure 5). The quantity of RAM and ROM pages mapped into the specific address space depend on the MBC used and the RAM or ROM on the cartridge itself.

1.4. CARTRIDGE TYPES

Figure 1 shows the inside of a cartridge. It seems to be a type 10h-cartridge with RAM on the lower left, ROM on the lower right, a battery above, MBC3 in the upper region and a timer in the upper left. The pins of the connector start at the left with VCC (pin 1) and end on the right with the GND (pin 32). For more detail see section 3.1 'How to connect an Eprom'.



Figure 1: Inside a cartridge
 (Type 10h)

There are different types of Game Boy cartridges available. These types vary mainly in their RAM/ROM addressability and may be classified into several categories. The information on the cartridge type will be found (or must be placed) at address 0147h of the ROM; ROM and RAM size at 0148h and 0149h. Table 2 shows the different setup available (01/99). For further information see gbspec.txt [\[1\]](#): "Cartridge type".

147h	Cartridge type	147h	Cartridge type
00h	ROM ONLY	12h	ROM+MBC3+RAM
01h	ROM+MBC1	13h	ROM+MBC3+RAM+BATT
02h	ROM+MBC1+RAM	19h	ROM+MBC5
03h	ROM+MBC1+RAM+BATT	1Ah	ROM+MBC5+RAM
05h	ROM+MBC2	1Bh	ROM+MBC5+RAM+BATT
06h	ROM+MBC2+BATTERY	1Ch	ROM+MBC5+RUMBLE
08h	ROM+RAM	1Dh	ROM+MBC5+RUMBLE+SRAM
09h	ROM+RAM+BATTERY	1Eh	ROM+MBC5+RUMBLE+SRAM+BT
0Bh	ROM+MMM01	1Fh	Pocket Camera
0Ch	ROM+MMM01+SRAM	FDh	Bandai TAMA5
0Dh	ROM+MMM01+SRAM+BATT	FEh	Hudson HuC-3
0Fh	ROM+MBC3+TIMER+BATT	FFh	Hudson HuC-1
10h	ROM+MBC3+TIMER+RAM+BATT		
11h	ROM+MBC3		

Table 2: Different possible cartridge configurations

2. THE PROJECT IDEA

The idea is to connect other devices to the Game Boy. Usually this can be accomplished by using standard input/output interfaces called ports. These ports are 'pins' to connect external devices to data-, address- and control-buses. The parallel port should be accessed as easily as a RAM or ROM, to keep programming simple.

2.1. WHAT IS MISSING IN THE GB-HARDWARE ?

The Game Boy already has a serial port which could be used if there weren't two major problems. The first is that the data flow-rate on serial ports is slow. The second is that the 'game-link port' does not seem to support any standard protocol for data interchange. There is little documentation on this 'link port' (see gbspec.txt [\[1\]](#)). If it shall be used custom protocols must be developed. (Note: for programming it might be handy that the link port is the only port in the GB hardware having it's own interrupt)

This best performance for connecting external hardware is achieved with an 8 bit parallel port. This is the basic idea which will be discussed in paragraph 3.0 Building the hardware.



Figure 2: Game Boy with adapter & ROM

3. BUILDING THE HARDWARE

This section covers the design and build of the parallel port. It includes general details on the required ROM interfacing to the Game Boy. The GB-Adapter is a cartridge supporting the addition of 32k*8 and 64k*8⁶ EPROMs⁷.

The capability of the standard programmable I/O chip 82C55A will be considered and the GB-I/O parallel port project circuit diagram described. A ready-to-etch layout for the printed circuit board is included as part of the design.

3.1. HOW TO CONNECT AN EPROM

The very first step in developing software for the Game Boy is to connect the required program memory to the 'Gamepak Edge Connector' (GEC). This can be done in different ways. One possibility is to take an old cartridge (maybe with an MBC1), solder the ROM out and connect all of the data and address to an EPROM.

A more professional way (if you can build/etch boards) is to make an adapter yourself. For this purpose you will need to know how to connect an EPROM to the Game Boy.

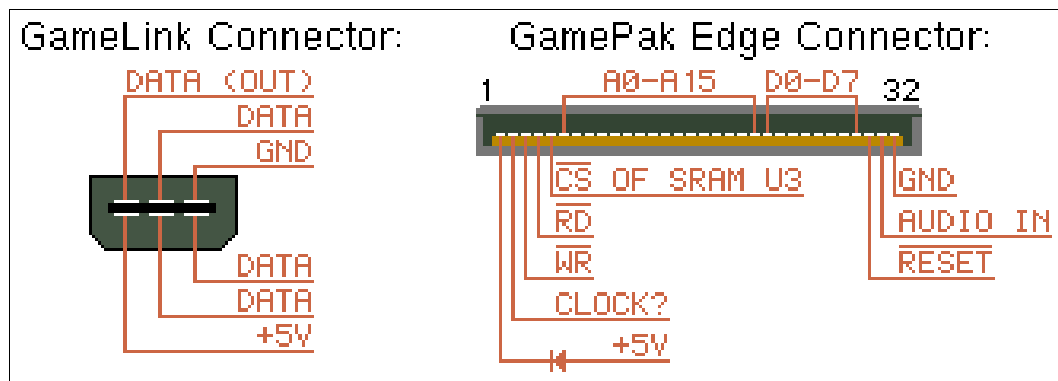


Figure 3: GameLink- and GamePak-connector pinout

On the right of Figure 3 is a view of a cartridge connector as seen from the Game Boy slot (the text on the cartridge is on the upper side). The picture shows the pinout of the cartridge and should be used for further reference.

⁶ In this special case there is no MBC used. The switching is performed by a hardware switch.
⁷ Only the term EPROM will be used. PROMs EPROMs OTP-EPROMs having the same pinout may also be used. Flashes and EEPROMs have nearly the same pinout whereby they require a different programming method.

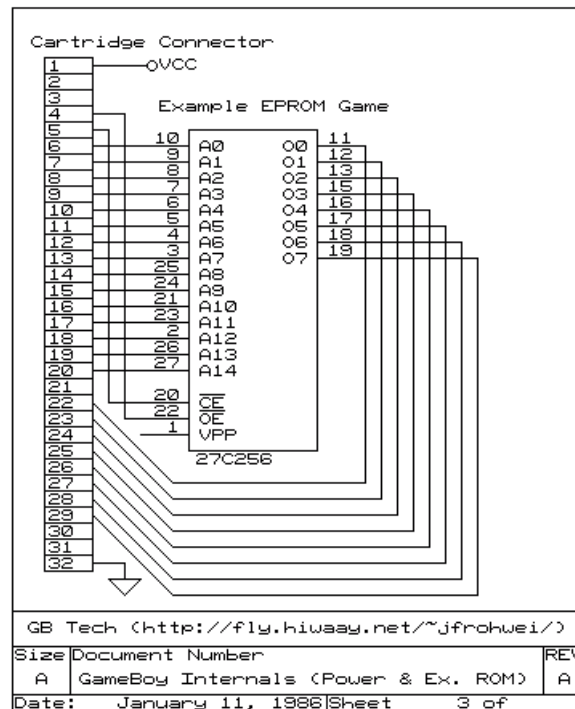


Figure 4: Wiring diagram for a 32kB EPROM

Figure 4 shows the wiring diagram for an 32k*8 EPROM. Address and data lines are connected directly. The power for the EPROM (Vpp) can be obtained from pin 1 (Vcc) of the GEC. Ground (GND) is connected to pin 32 (GND) of the GEC. A bit more tricky are the chip enable (/CE) and output enable (/OE). They are connected to read (/RD) and write (/WR) respectively. Notice for further development that all these signals are low active. To access the lower or upper half of a 64k*8 EPROM simply use a jumper or switch to connect A15 of the EPROM either with GND or Vcc. Any CMOS EPROM witch an access time of less than 120ns may be used.

3.1.1. THE GB-ADAPTER

The GB-Adapter is a simple adapter from the ‚Gamepak Edge Connector‘ (GEC) to a 28 pin DIP socket or an 34pin header. It supports 256Mbit and 512Mbit EPROMs. When using a 512Mbit (64k*8) EPROM J3 is used for selecting the upper or lower 32k of the EPROM. This way two different programs can be stored in one EPROM, accessed by turning off and switching.

All 32 pins of the GEC are also available on a standard 2x17-pin connector on the adapter board. The idea of the adapter was to create an easy way to connect further circuitry to the Game Boy . An EPROM can also be mounted on the board if required. The capacitor provides for ripple and spike supression and should

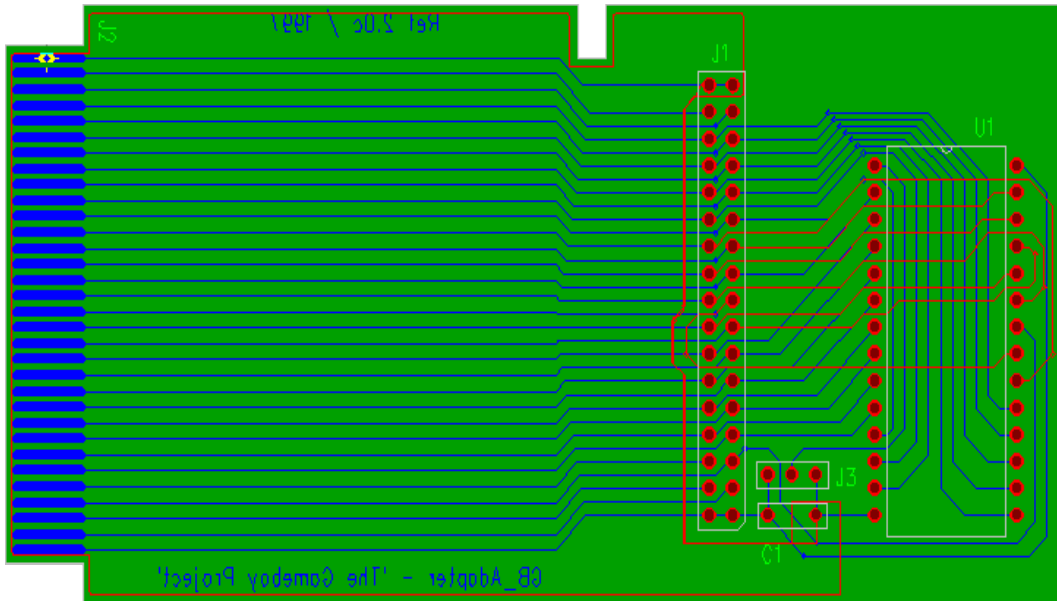


Figure 5: Board layout for the GB-Adapter

have a value of around 100nF (C1). The board used is a single sided 60mm*110mm board. Figure 5 shows the board layout. To download the ready to etch layout plans see appendix [5.2.1 'Layouts'](#). Note that the view is from the component side of the board through to the solder side.

The connector J2 may be used as an expansion slot for further applications. Pin 1 & 2 of J1 (=J1.1 & J1.2) are connected to pin 1 of the GEC (= J2.1). Pin J1.3 goes to J2.2, J1.4 to J2.3 and so on (See Figure 4). The /WR of the GEC is not routed to the EPROM but to A15 to disconnect the EPROM from the data bus while not in use.

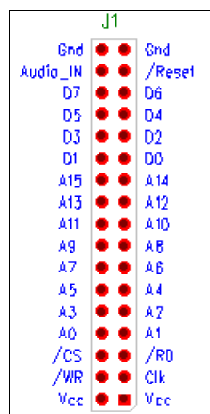


Figure 6: GB Adapter J1 pinout

3.2. THE 8255A PI/O - CHIP

"The 82C55A is a high performance CMOS version of the industry standard 8255A. It is a general purpose programmable I/O device which may be used with many different microprocessors. There are 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The high performance and industry standard configuration of the 82C55A make it compatible with the 80C86, 80C88 and other microprocessors. Static CMOS circuit design ensures low operating power. TTL compatibility over the full military temperature range and bus hold circuitry eliminate the need for pull-up resistors"⁸.

A block diagram of the 8255 is shown in Figure 7. The device comprises three 8 bit ports whereby port c can be subdivided into two 4 bit groups. Each of these three ports is addressed by A0 and A1. With its read, write, the chip select and data signals it looks and behaves like a tiny ROM or a RAM with only three bytes. To switch between 'RAM-' and 'ROM-mode' the 8255 has a mode register where three working modes can be selected. The mode register is selected when A0 and A1 are set to high (+5V).

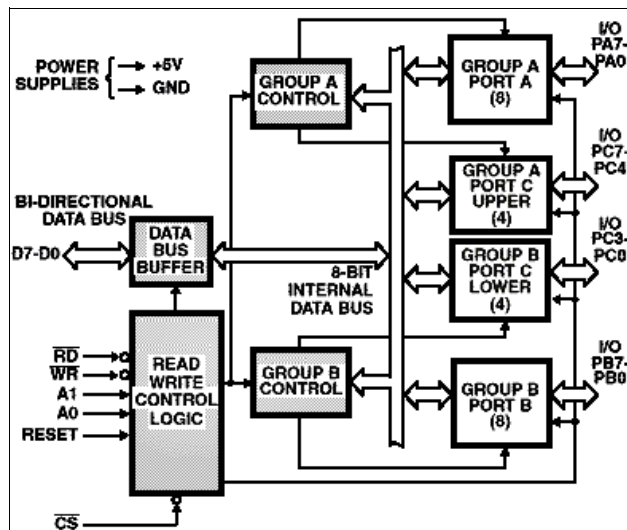


Figure 7: Block diagram of a 82C55A

The three modes are:

- Mode 0: basic input / output
- Mode 1: strobed input / output
- Mode 2: bi-directional bus

For our purpose only mode 0 was interesting, but the device may be switched into each with the setup supplied in this document. To work with the three ports they must first be initialized. This is done by writing the proper control word into the control register. Figure 8 shows the definition of the control word. While bit D2,

⁸ From: HARRIS Semiconductor, data sheet on 82C55A [3]

D5, D6 and D7 define mode 0..2
 the bits D0, D1, D3 and D4 define the settings of the ports to input- or output-mode. In other words you need to 'program' or switch the 82C55 every time you want to change a port data flow direction. If 'all input mode' in mode 0 is requested the control word would be $10011011_{bin} = 9B_{hex} = 155_{dec}$. If 'all output mode' in mode 0 shall be selected the control word would be $10000000_{bin} = 80_{hex} = 128_{dec}$.

For further information on the 82C55A and it's capabilities see the PDF-document (on local server).

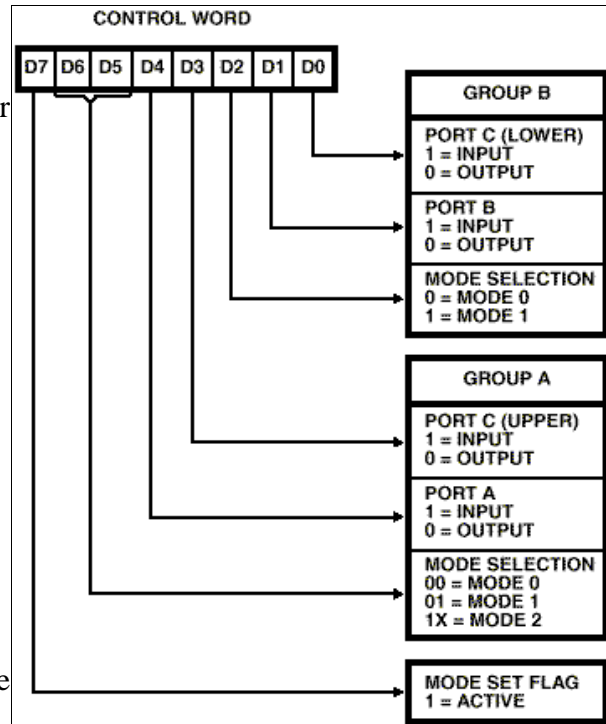


Figure 8: The control word for port definition

3.4. THE GB-I/O INTERFACE BOARD

In principal the GB-I/O interface board is functionally the same as the circuit diagram shown in Figure 9. We just realized the logic in full-NOR-technology to reduce the IC count. In doing this we eliminated the 74HC04 and the 74HC32 and put in a 74HC04. Components used:

<i>Part</i>	<i>Description</i>	<i>Part</i>	<i>Description</i>
U2	Harris 82C55A (DIP)	J1	Header 34 pin (2.54mm grid)
U3	74HC00 (DIP)	J3	Jumper
U4	74HC138 (DIP)	C1	Capacitor (100nF)
U5	74HC30 (DIP)	Cx	5 * Capacitor for each IC between Vcc/Gnd
U6	NM27C512Q120 (DIP) (='EPROM 64k+8)	board	double sided 100 mm * 160 mm

Table 3: Parts for the GB-I/O interface board

Figure 10 shows the pinout of the 34pin header J1 of the GB PI/O interface board. The purpose of the header is to allow expansion for further devices. Vcc, ground, clock, U3 chip select and the three ports are available on the header.

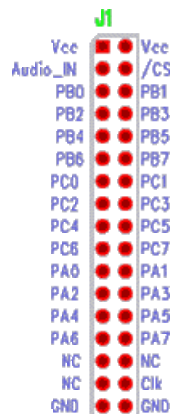


Figure 10: PI/O-Board J1 pinout

Figure 17 displays the board layout. To download the ready to etch layout plans see appendix 5.2.1 '[Layouts](#)'. Note that the board is viewed from the component side through the board to the solder side (see next page).

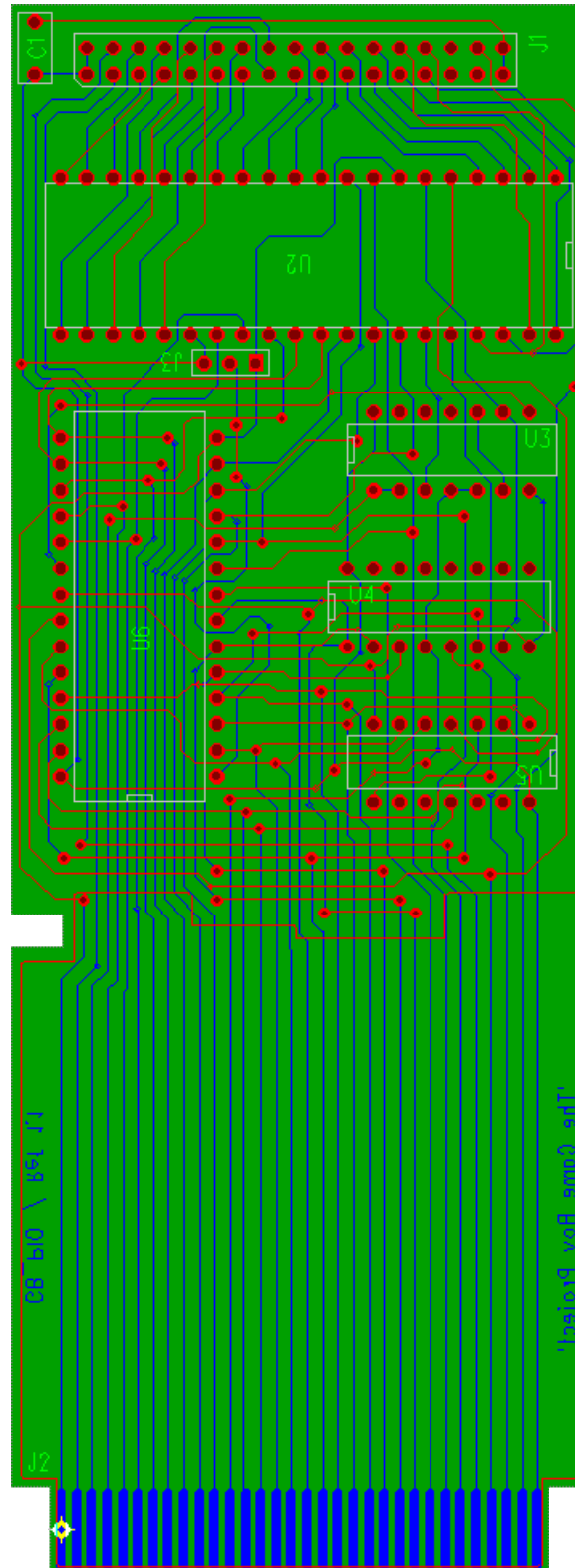


Figure 17: GB-I/O interface board layout

4. CREATING THE SOFTWARE

There are several tools to write programs in different languages for the Game Boy and all are freeware for noncommercial use. There are a Basic-interpreter, a C-compiler and different assemblers online. A set of software tools for a development kit shall be described in this section. These are only some of the tools available, and they might not be the best to work with. However, the internet grows and there are more to come.

4.1. SOFTWARE (DEVELOPMENT) TOOLS

This chapter contains information on software development tools for emulating and testing purpose. The tools listed here can be understood as a basic set of tools to develop applications for the GB-PI/O board, but also for general GB programs.

4.1.1. GBBASIC

'*Game Boy Basic*' is an online basic interpreter written by Jeff Frohwein [1]. This makes it a very handy tool when testing the GB-PI/O board or any other hardware setup. You can use it for example to set the 8255 command register or to read or from or write to the ports. The software comes as ready compiled binary (gbbasic.gb) and can be downloaded from the local server (see 5.2.2'[Software tools](#)').

4.1.2. GBDK

The '*Game Boy Development Kit*' [4] includes an ANSI C based and a TASM based compiler, linker and builder. It is available for DOS and Unix (Linux). GBDK consists of a command line accessed compiler which compiles .c and *.s files to GB ROM images (.gb). As usual for C, it is possible to include assembler routines in the C source code. GBDK was written by Pascal Felber and has it's own web page [4]. Although there are other Z80 based assemblers optimized for the Game Boy, GBDK is one of the most powerful packages available and will let you easily write your own complex programs in C or ASM.

4.1.3. GBDEV STUDIO

The *,Game Boy Development Studio'* [\[5\]](#) is a Windows 9x frontend for GBDK. It is mainly a text editor with a C syntax checker to make your source code better readable. The buttons in the editor can be set to special command line functions. By this it is possible for example that the *,compile'-button* lets GBDK compile your source or the *,execute'-button* passes the compiled ROM image to an emulator.

4.1.4. NO\$GMB

Although there are a couple of Game Boy emulators available in the internet, the *,Nocash Game Boy Emulator'* [\[6\]](#) is the mightiest emulator we've come across when it goes to software development. NO\$GMB (by Martin Korth) features executing code, tracing through code, setting breakpoints and more. It works together with the *.sym file which GBDK automatically produces (GBDK version 2.0.15 and higher). This debugging features will be a good help when making your own software.

4.1.5. GBTD & GBMB

These are two also very handy tools when you are building you own screen data for the Game Boy. *,Game boy Tile Designer'* and *,Game Boy Map Builder'* [\[7\]](#) by Harry Mulder are Windows 9x programs that help you designing your own maps and tiles. The output can be included in either C or ASM images.

4.2. FIRST STEPS USING GB-BASIC

After the hardware had been built the functionality should be checked. An easy way of doing this is to write a short GB-Basic program. This requires an EPROM with the latest version of Jeff's GB-Basic [2] in it. After plugging in the EPROM and switching on the GB the ,Nintendo' start screen should appear. If the screen is blank or ,snow', stripes appear on the screen the hardware should be checked.

Begin with a check of the EPROM (Programmers can read out the ROM's and cross check it with the source file), then re-check the layout. If GB-Basic is up and running you're more than half way home!



Figure 18: GBBasic accessing the 82C55A

To check the 82C55 requires access to the 82C55 at the hardware addresses 7FF8h to 7FFFh. This can be achieved by reading and writing to these addresses using the Basic commands peek and poke.

To output a bit combination at a port simply write it with 'poke <port-adr> <value-to-write>'. To read a bit combination from a port use 'peek <port-adr>'. TO display the returned value on screen use 'print peek <value>'. The address and value can be in decimal or hex (put a 'h' behind the value then). Don't forget that the ports of the 82C55 have to be set to either input or output mode before reading or writing data. This is done by writing the control word to 7FFFh (the control register) - see paragraph 3.2.

To generate a bit combination for testing use pull up or pull down resistors of approximately 5kΩ. Note that the driver capacity of the 82C55A is limited. To drive an LED for example requires an additional transistor.

4.3. THE PROGRAM

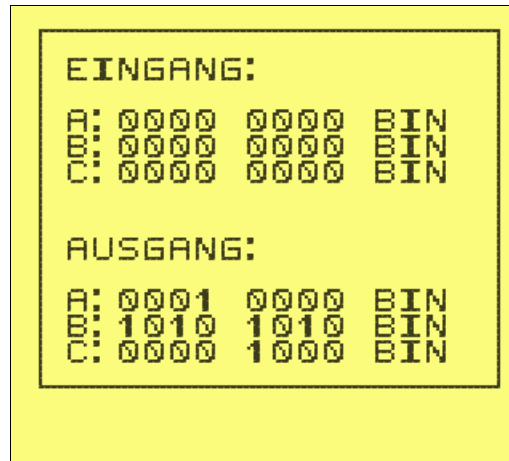


Figure 19: Screen shot of the testing program

A testing program was written in assembler using the GBDK [4] assembler. The functionality is limited to setting and resetting single bits on the different ports. A port read option with screen output is integrated. A simple basic program would have done the same, but the use of assembler offered the opportunity of learning how to use the compiler. The source code is well documented on a line by line basis. Figure 19 shows a screen shot of the program. ‚EINGANG‘ shows the actual settings of port a,b,c. ‚AUSGANG‘ shows the output status of each port. Move the cursor up down to switch between input and output mode. In output mode (‘AUSGANG‘) move the cursor left to increase the output on a binary basis; move the cursor right to decrease. Port b does a ‚invert‘ of the bits every cycle, while port c does a shift right of one bit. The GB ROM image of the program (steffen.gb) can be found under appendix 5.2.2‘[Software tools](#)‘. Use the NO\$GMB emulator [6] to trace through the program and see it’s documented source.



5. APPENDIX

5.1. REFERENCES

- [1] gbspec.txt, also called the ‚PAN Document‘.
<http://www.fh-karlsruhe.de/fbnw/html/Gameboy/Docs/gbspec.txt>
- [2] GBBasic, Jeff Frohwein’s ‚Game Boy Technikal Central‘.
<http://fly.hiwaay.net/~jfrohwei/gameboy/>
- [3] HARRIS™ 82C55A CMOS Programmable Peripheral Interface.
http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/hardware/Harris_82C55.pdf
- [4] GBDK, ‚Game Boy Development Kit‘ by Pascal Felber.
<http://lsewww.epfl.ch/~felber/GBDK/>
- [5] Gbdevkit, ‚Game Boy Development Studio‘ by ‚AXIS DESIGN‘
<http://www.geocities.com/Eureka/9827/>
- [6] NO\$GMB emulator by Martin Korth.
<http://www.work.de/nocash/>
- [7] GBTD, GBMB, ‚Game boy Tile Designer‘ and ‚Game Boy Map Builder‘ by Harry Mulder.
<http://www.casema.net/~hpmulder>
- [8] PADS Inc., Software company with a high end PCB layouting program called ‚PADS‘, <http://www.pads.com/>
- [9] CADSOFT, Software company with a high end PCB layouting program called ‚EAGLE‘, <http://www.cadsoft.de>

5.2. DOWNLOADS

5.2.1. LAYOUTS

In this chapter you will find everything you can find on the site itself to download. You will find different file formats for every project part. The files declared as Pads files are the files generated with the Pads 2.0 demo of [Pads Inc. \[8\]](#). Unfortunately it is not possible to download the demo but you will find a site at your local Pads distributor to get a demo CD shipped to you. The ‚ps‘ files are postscript files - best accessed with GhostView. The ‚pdf‘ files are files for Adobes Acrobat’s reader.

The SMD Version is only a **development preview**. It was designed with the Eagle

3.55 [9] freeware which can be downloaded at the distributors site. The freeware is fully functional for one schematic sheet and a board size of 80mm x 100mm. I strongly recommend this for your own development, because of the easy handling and the integrated autorouter. The libraries can also be found here.

<i>Files</i>	<i>Description</i>
gb_adap.zip	Pads files for the Game Boy Adapter (only layout yet) http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/gb_adap.zip
adap_ps.zip	layout for the GB-Adapter in postscript format http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/adap_ps.zip
adap_pdf.zip	layout for the GB-Adapter in Adobe Acrobat format http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/adap_pdf.zip
gb_pio.zip	PADS [8] files for the PIO board (layout) http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/gb_pio.zip
pio_ps.zip	layout for the PIO board in postscript format http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/pio_ps.zip
pio_pdf.zip	layout for the PIO board in Adobe Acrobat format http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/pio_pdf.zip
pio_smd.zip	Preview of the PIO-SMD version designed with EAGLE [9] http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/pio_smd.zip
lib.zip	PADS [8] 'user' library with the parts used in the PCBs (e.g. the Gamepack Edge Connector GEC) http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/lib.zip
prj_all.zip	all the PADS [8] files above http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/prj_all.zip
gb_lbr.zip	EAGLE 3.55 [9] library for Game Boy parts (GEC, etc) http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/prj_all.zip

5.2.2. SOFTWARE TOOLS

These files can be found on the local server but might not be up to date.

<i>Files</i>	<i>Description</i>
bbb122.zip	Online basic interpreter to test your board www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/software/gbb122.zip
Steffen.zip	The asm program Steffen wrote for testing the board http://www.fh-karlsruhe.de/fbnw/html/Gameboy/downloads/steffen.zip