PEPS-III

Pro-gram-ma-ble
E-prom-Simula-tor

Manual

PEPS-III **Manual**

State: **08/12/1998**

Software: Bernhard Emese

Andreas May

Manual: Johannes C. Lotter
Andreas May

Contents

1 Introduction 1

PEPS is a tool for designing control software ("firmwa-re") for micro- and monoboard computers respectively. During that process the PEPS (**P**rogrammable **EP**ROM-**S**i-mulator) is lodged in the EPROM-socket, takes over the EPROM's functions and similar to an CPU emulator allows to quickly test the software. Additionaly some RAM types may be simulated, so that the memory of the micro- or monoboard computer may be read into the the PC. Please note that the CE signal is not evaluated during RAM simulations.

PEPS is being loaded with the program that is to be tested from an IBM-compatible PC via the printer port. As only the EPROM is being simulated, the type of CPU does not concern the PEPS. The software supports systems with a 16-Bit- as well as with a 32-Bit-data -bus.

Technical Data:

Data transmission rate up to 32.000 byte/sec. (on Intel 386)
Current intake (during access) below 60 mA
Access time below 100 ns

Obviously the technical specifications are by all means superior to those of the EPROM to be simulated. The higher data transmission rate saves you involuntary breaks. PEPS is programmed in seconds.

During data transmission and during testing the PEPS remains linked to the PC. The data is not lost even after switching off the system or disconnecting the cable. Power supply is effected through the target system; amperage is only about a third of that of the EPROM to be simulated.

PEPS does have a reset terminal. This terminal will be held to GND during data transmission. In simulation mode, the reset terminal has a pull up resistor to VCC. The software supports a debug mode outputting pulse chains via the reset terminal to test the start-up behaviour of the target system (s.b.).

PEPS contains 512 KByte internal battery buffered RAM. This allows you to simulate EPROMs (2716, 2732, 2764, -27128-, 27256, 27512, 27513) and RAMs (6116, 6264, 62256, 511001, 628512) between 2 and 512 KByte. The device type is selected with the DIL switch.

2 Quickstart 2

To allow you to start working with the EPROM simulator as fast as possible, we would like to take the opportunity to familiarize you with the most important steps allowing you to operate PEPS.

a) On the diskette you will find the programm PEPS.EXE. That is all you will need in terms of software. Copy it to your hard disk or your scratch diskette. If there is also a READ.ME file on the diskette, do please read it, as it will contain updated information not yet included in this manual.

b) Select the desired EPROM or RAM type respectively with the switches S1 to S6. The display for the necessary setting of the DIL switches is part of the control software and is fetched by calling up the program without file names or paramter.

PEPS

For a 64 kB EPROM of the type 27512 the following must be set:

EPROM size 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 27512 64 kB X X - X
X - - - - - - - - - - X -

X: DIL-Switch On

d) Insert the conductor flat cable into the EPROM socket of your target system (**Note polarity! PEPS is destroyed when improperly poled!**) and connect the PEPS via the interface cable to your PC´s parallel port **LPT1** (if you wish to use a different parallel port, please read the chapter 'Software' beforehand). If you wish to simulate an device with less than 32 pins, you will have to use an intermediate socket that has the same number of pins as the device. Switch on the target system.

e) Create an EPROM file that can run on the target system (let's assume that it is called **TARGET.COM**) and then initiate the transmission by calling up:

PEPS TARGET.COM

During data transmission the LED on the PEPS will light and the reset terminal is on GND. When data transmission is finished, the LED will go off and the reset terminal takes on high impendance. PEPS now behaves like an EPROM programmed by **TARGET.COM**.

3 The Software3

The software is running under DOS, or in a Windows DOS box. For Windows NT, you need a special driver, available from the web server www.engelmann-schrader.de. You can also get a LINUX version.

The PEPS-loading program is mainly used to transfer EPROM data from and to the simulator. Apart from that it offers features like file conversion, manual "patching" in the simulated EPROM as well as an interactive debug-mode and a few other options to be described in the following.

Most compilers, assemblers, and linkers create codes either as an EXE file, a binary file, or an ASCII file in either the INTEL-HEX or MOTOROLA--S formats. EXE-files may be converted to binary files with the DOS instruction EXE2BIN; the other formats are recognized by the PEPS loader, converted into binary and transmitted. The transfer is carried out with the speed of about 10 - 32 KByte/sec, depending on the computer.

Output may be directed to one of the parallel ports LPT1 to LPT4. Target systems with a 16-Bit or 32-Bit data bus may at the same time transmit data to 2 and 4 PEPS devices simultaneously using only one parallel port and the loader. Apart from that you may also instead of writing the data to the PEPS simulator write the data to a file with identical content.

Error messages (e.g. wrong file format, checksum error or transmission failure) as well as check messages like number of bytes transmitted and the file format are displayed on screen.

Call:

PEPS [?] returns a list of available options as well as the DIL switch table.

PEPS [<Name>] [Options] transmits data to the PEPS.

<Name>: Path name of the file containing EPROM-data to be loaded (source file, must be given before options).

Options: are introduced by minus sign '-', followed by the option character and a parameter if needed, usually a number or a file name.

Important: between the options character and the parameter there may be no blank. Blanks on the other hand have to be used to separate options from one another.

-ennnnnn loads PEPS only from EPROM-adress nnnnnn (hex, default 000000).

-w<Name> Output not to PEPS, but as binary to file <Name> or reading out the PEPS and saving to file <Name>, if no source file was given.

-bn sets the number of PEPS--devices linked up: 1, 2 or 4 (default 1).

-pn sets the number of the parallel port used: LPT1 (default), LPT2, LPT3 or LPT4.

-fn file format (0 = binary, 1 = INTEL--HEX, 2 = MOTO-RO-LA--S).

If the -f option is left out, PEPS will recognize the file format from the file's extensions. .HEX = INTEL, .SHX = MO-TOROLA, all others: binary.

-cn Check/Verify-Option: checks the data transfer to PEPS.

0 = No check.

1 = execute hardware-test only
(default).

2 = Read back and check all transmitted data.

-on Debug-Mode: 0 = don't activate (default), 1 = activate after loading.

-inn nn.. Patching: Directly loads bytes (hex) to the EPROM-adress specified by **-ennnnnn**.

All integers (nnnnnn) are hexadecimal entries of up to 6 digits (numerics from 0-F). Leading zeros may be left out. With the option -i the bytes following the first one must each be separated by a blank. The sequence of the options is arbitrary, as long as the file name is entered first thing before the options.

Thus an example for an input:

```
PEPS c:\source\eprom.HEX  -wc:\source\eprom.BIN
```

The INTEL-HEX-file C:\source\eprom.HEX is converted into a binary file and written to C:\source\eprom.BIN.

Or:

```
PEPS eprom.BIN -e0f00 -c2 -p2
```

The file eprom.bin from the current table of contents is transmitted to PEPS, beginning with adress 0f00 (hex). PEPS during this is connected to the printer port LPT2 (-p2). Each byte is read back and checked during transmission (-c2).

Or:

```
PEPS -wD:\read.bin -l20000
```

Reads 128 kB out of the PEPS and saves it to the file D:\READ.BIN.

Or:

```
PEPS -i50 45 50 53 0d 0a -e1000
```

Transmits bytes 'P','E','P','S',<CR>,<LF> to the target device starting with adress 1000 (hex).

3.1 Note concerning INTEL-HEX files

PEPS will automatically recognize INTEL-HEX file format for a file with the extension .HEX. Using the segment address of the start record, the extended address record, and the address offset of each data record these data may be transmitted to the respective position in the PEPS. During this process only those addresses in PEPS will be overwritten, that occur in the data records. Other areas, including gaps between data records, will remain unchanged.

Should the address read (AdressOffset + Segment/16) be larger than 512k, a warning will be issued, as PEPS can only simulate address segments up to 512k.

Should a record contain an incorrect checksum or byte number or be incomplete, the transmission will be terminated and a corresponding error message will be issued.

3.2 Note concerning MOTOROLA-S files 32

PEPS load program will automatically recognize MOTOROLA-S-file format for a file with the extension .SHX. Similarily to the procedure described for the INTEL-HEX format, load addresses will be automatically calculated and data will be transmitted to these addresses. The same error conditions will be tested for and displayed if necessary as well.

3.3 Note concerning Binary Files 33

If PEPS does not recognize any of the two formats, the PEPS loader will assume the file to be an unformatted binary file and will transmit the file unchanged starting at the target address set with -ennnnnn.

The option -f0 allows you to define every file as a binary file. This allows you to transmit files correctly that conicidentally start with a answer code but actually are binary files. The options -f1 and -f2 attempt to read a file in the INTEL-HEX or MOTORO-LA-S format even though they are missing the respective extensions.

3.4 Outputting Data to a File

With the option -w chosen, no transmission to PEPS will take place. Instead the data will be written to a binary file on diskette. During outputting INTEL-HEX and MOTOROLA-S files to diskette gaps in the data between data records will be filled up with ZERO bytes (00 hex). Large jumps between addresses within the file may easily cause the creation of huge files. Prior to the offset address of the first data record no ZERO bytes will be added, the output file will always start with the data of the first record.

Should a data record with an offset address below the one in the previous record occur, the data will be appended in the file and a corresponding warning will be issued.

3.5 Patching

The option -i allows you to send one or any number of bytes to PEPS without having to specify a file. It can be used to change single bytes at any address without having to reload the entire program. Transmission will always be effected to the PEPS address set with -ennnnnn and contains as many consecutive bytes as is allowed by the input.

Please note: specifying a read file causes the consecutive transmission of the single bytes and the file, probably causing the former to be overwritten by the latter.

3.6 Data Security

The PEPS load program allows you to check each byte transmitted for transmission errors. The option -c2 activates this check causing increased transmission times. On the other hand it does allow you to exclude the possiblity that errors in transmission cause the target device to malfunction.

Yet the default setting -c1, where only the first byte sent will be read back while all others will only be written, does offer a sufficient check of the hardware.

-c0 disables both checks and simply transmits all data without any check whatsoever - even if no PEPS is connected to it.

3.7 Debug-Mode <u>37</u>

This mode offers support when implementing a target system with previously un-tested hardware. It is especially valuable if initially nothing works. -o1 acitvates the mode. Corresponding instructions will appear on the screen after loading.

In the debug mode you may initiate a reset of the target system from the PC's keyboard, either resetting or 'clocked' with a selected frequency. This latter mode allows you to check on the startup behaviour of the target system using an oscilloscope. Most hardware errors may be quickly tracked down this way.

The red reset clamp of PEPS for this purpose needs to be connected to the reset inlet of the target system.The clamp is connected to an open-collector outlet switched towards GND with reset active. In the reset mode the state may be discerned from PEPS' LED display. If the LED is glowing, RESET is active.

The following keys are active after loading in the debug mode:

^C,esc,cr	=> Return to operating system.
R	=> Permanent-Reset on.
I	=> Clocked Reset on.
<,>	=> Reduce/increase clock frequency.
others	=> Short Reset-Impuls; Clocked-Reset off.

When returning to the operating system the last state of the reset connection will be kept.

3.8 16- and 32-Bit-Target-Systems

With target devices having a data bus larger than 8-Bit the parallel use of 2 (for 16-Bit) or 4 (for 32-Bit) PEPS is required to cover the data bus. This configuration you can chose using -b2 or -b4 respectively. It causes only every 2nd or 4th byte by turns to be loaded to the same PEPS to ensure that a 16-Bit or 32-Bit EPROM-data file be correctly distributed between the different simlulators.

This operating mode requires a 14-pole stepped cable for parallel switching of the PEPS devices via the extension terminal. The stepped cable is a conductor flat cable with four post-connectors where slot numbers are allotted by ommiting conductors. It is available as an optional extra. The extension terminal has 14 pins and is led out of the PEPS' right side. Connect the stepped cable in such a way as to insure that the corresponding poles of the PEPS' are connected (bus principle) and pin 1 of the stepped cable (marked pin) points away from the PC connection.

By connecting two or four PEPS' with a stepped cable your automatically allot device numbers used by the load program to allocate the data to the PEPS devices. The PEPS with the number one is the one connected to the first slot of the stepped cable, which can be identified by the presence of all conductors of the tape cable. From one slot to the next the device number increases by one.

Only the following configurations are possible:

a) one PEPS, no stepped cable. 8-Bit mode.

b) one PEPS on slot 1 and 2 respectively. 16-Bit mode.

c) one PEPS on slots1 to 4 respectively. 32-Bit mode.

Only the first PEPS may be connected to the PC. All other PEPS devices must have their probably connected D-SUB-cables disconnected. Otherwise the load program is unable to address the PEPS devices correctly.

If during loading a PEPS device addressing problems should occur you will be issued the following message:

Peps n not responsive .

Should a loading malfunction occur even though there is a correct connection between the PEPS devices and to the PC, you should check whether

a) all PEPS devices have power.

b) the DIL-switches of all PEPS devices were set correctly.

c) the pins 18 - 25 of your PC's printer port are all connected to GND.

4 Pin Assignment4

PEPS can simulate EPROM with less then 32 pins. In this case you must interpolate one or several 24-or 28-pin sockets; pins 1,2,31,32 or 1,2,3,4,29,30,31,32 respectively remain free. (With 32-pin EPROM--types it is also advisable to put a socket under the conductor flat cable to avoid damaging the pins).

The trunk with the D-SUB-plug is provided for the connection to the parallel port of the development system (PC). It has the following assignement:

Pin #	Signal	Meaning	Printer Port
2	M0	Mode-Flag	DATA0
3	M1	Mode-Flag	DATA1
4	M2	Mode-Flag	DATA2
5	DATA	Data Line	DATA3
6	CLK1	Clock Line 1	DATA4
7	CLK2	Clock Line 2	DATA5
8	CLK3	Clock Line 3	DATA6
9	CLK4	Clock Line 4	DATA7
11	DOUT	Returning of Read Data.	BUSY
18-25	GND	Earth Terminal	

ATTENTION - the GND-Potential of the target system is linked to the PC's GND via PEPS. If the target system is earthed the PC must be operating from the same mains network. Differences of potential caused by network grounding on both sides may destroy PEPS, the target system and your PC!

5 PEPS Hardware-Programming 5

This chapter should only be of interest to you if you wish to write your own PEPS load program. Programming PEPS is done using the 8 control bits CLOCK1 - 4, /DATA, M0 - M2, that have to be connected to eight output lines of the PC, e.g. to a PIO or a centronics-interface.

The operating mode of PEPS is set via the inlets M0, M1 and M2. Five modes are available:

M2-1-0:	000	Count up addresse (Mod. 0)
	001	Send or Receive Data Byte (Mod1)
	010	Load Byte to RAM (Mod 2)
	011	Simulate EPROM or RAM (Mod 3)
	100	Load Byte from RAM (Mod 4)

Modes 0 to 2 and 4 have their RESET terminal on GND, the target system is reset. Only in mode 3 does the RESET become inactive.

The CLOCK-Line triggers the action preselcted by M0-2. CLOCK is HIGH-active and must assume a LOW-level in idle state or during mode change.

The following meanings are assigned: CLOCK1=> Device1 is being accessed; CLOCK2=> Device2 etc. (The device number is set via jumper on the PEPS)

During mode 1 the data byte is written via the /DATA-Line or read out via the /DOUT-Line serially inverted.

In order to write a data byte to any PEPS address the following steps have to be undertaken:

5.1 Set Address: In mode 3 a HIGH-level on the CLOCK-inlet sets the internal address counter to zero. In mode 0 the address set will be increased by one during each HIGH--edge of CLOCK. By resetting and repeated counting up any address 0..FFFF (hex) may be set this way. Addresses set once are kept even when modes are changed.

The following timing--dia-gram shows the setting of address 0004:

```
                 Resetting          Counting Up
M0       --------------______________________________________-
M1       --------------______________________________________-
M2                     ________________________________________________
CLOCK    ___----______----______----______----______----_____
Address            0000                   0001      0002      0003       0004
```

5.2 Send Data Byte: Mode 1 must be selected for this. Now the inverted byte will be transmitted serially via the DATA--line. At each LOW-edge of CLOCK here the inverted level at the DATA-inlet will be accepted as bit. Bit 7 will be transmitted first, bit 0 last. A byte once pushed through will be kept even when changing modes. The impulse diagram below shows the writing in of bytes 35 (hex) = 00110101 (binary).

```
M0       _----------------------------------------------------------
M1       _______________________________________
M2       _______________________________________
/DATA    __--------------_________------_____------_____
CLOCK    ___--___--___--___--___--___--___--___--__
Bit          0     0     1    1     0    1     0     1
```

Note: The data written in appear offset by 8 bit on the /DOUT-line. This feature is used to check for correct data transmission.

5.3 Load Data Byte: The byte sled in in mode 1 initially is located in the shift register and still needs to be loaded to the RAM. This is done in mode 2. The byte is written with the HIGH--edge at the CLOCK-inlet to the address set in mode 0.

```
M0        ------____________________________________
M1        ____--------------------_______________________
M2        ________________________________________
CLK       _________--------__________________________
Byte at 0004h
```

This procedure repeats until all data is transferred. After that, the simulation mode can be entered with Mode -3.

Mode-3 will be activated with a high level on the clock input:

```
M0        __------------------------------------------------------------
M1        __------------------------------------------------------------
M2        _________________________________________
CLK       _______--------___________________________
                     From this point simulation occurs.
```

In order to accelerate transmission some steps may be shorten. As data will usually be transmitted in blocks setting the address may be reduced to a simple counting up after each byte transmitted. Thus the sequence is Send - Load - Count Up etc. If identical data - i.e. zeros or FF's - are to be written to consecutive storage cells it is sufficient to send the data once, followed by repeated loading and counting up.

5.4 RAM-Simulation

In order to be able to read data from the memory of the target system into the PC as well, RAM simulation is possible.

To do this after setting the DIL switches for the respective RAM types the simulation mode - mode 3 - is set first.

Now the target system may read and write the PEPS like a normal RAM.

In order to read the RAM data into the PC the address to be read is set as describe in 5.1.

In mode 4 - loading Byte from RAM - a data byte will be accepted into the shift register.

```
M0        _________________________________________
M1        _________________________________________
M2        ________-----------______________________
CLK       __________----___________________________
                         Byte Accepted
```

The inverted byte will be transmitted serially to the PC via the /DOUT-line in mode 1 (cf. 5.2). During that the inverted level will be handed over at the /DOUT-terminal (e.g. 35 H) at each High-edge of CLOCK.