# BICEPS-IV

## Emulator Manual

## Version 7.1

**BRENDES DATENTECHNIK GmbH Dresdener Str. 10 D-26160 Bad Zwischenahn**

**Tel.: +49 4403 / 81 68 38 eMail: Info@brendes.de**

Internet: http://www.brendes.de

# Contents

# 1 Introduction

The in-circuit-emulator **BICEPS** is a professional tool, which considerably ease the development of electronic circuits on the base of microcontrollers of the 8051 family. The hardware components of the controller as well as the programs executed by it can be emulated in real time, i.e. with the original speed.

For in-circuit emulation the processor will be simply removed from the existing circuit and replaced by an adapter, which is connected with the emulator and which works like the removed processor. Additionally to the normal functions of the processor there are however the following possibilities with the support of a normal PC, to which the emulator is connected via an serial or parallel interface:

- Presentation and altering of all internal registers and memories of the CPU, but also of external memories and assemblies, located on the circuit under test and reachable by the CPU
- Alterations of the program memory contents
- Start of a program to be tested, execution of a program in real time and stop of the program run at different free chooseable break conditions
- Tracing of states of buses and external signals in a real time trace memory.

As additional possibility an Universal adapter is available for the **BICEPS** emulators, which does not replaces the processor but the EPROM of the circuit under test. All debugging possibilities, i.e. especially the control about the processor and the access to all memory ranges, stay preserved and go therefore far beyond the possibilities of a simple EPROM simulator.

The reader of this manual should be familiar with the architecture of processors of the 8051 family as well as with the fundamental progression of a hardware and software development for microcontrollers. Suitable data books should be at hand, because an exact knowledge of the emulated processor is an supposition for a successful development work.

The performance characteristics of the **BICEPS** emulator are in detail:

- Adaption to different processors of 8051 family by adaptor boards (PODs)
- No restrictions with respect to memory space, ports and interrupts
- Support of Hook-mode and Bondout versions
- Universal adapter for connection of the emulator via EPROM socket with preversion of all debugging possibilities
- Emulation memory: 128k RAM
  - can be configured as: 64k program and 64k Extdata RAM
    128k program RAM (banked)
  - expandable to 512k program memory
  - emulator internal or external memory (mapping)
- 12 external inputs for break and trace
- Break with program stop at any state of data bus, address bus or signals at the external logic probes
- Break counter
- Conditional breaks for IF-THEN conditions
- Trigger of external devices
- 32 bit real time counter for time measurements in the range 100 ns up to ca. 12 h
- 32k x 72 bit real time trace memory with tracing of all states of buses, the external signals and the real time counter (time stamp)
- comfortable search and filter functions for the trace memory
- Performance analysis
- Internal and external CPU clock
- **BicWin** user interface for Windows95/98/NT
  - easy operation by means of mouse or keyboard
  - free relocatable and alterable windows
  - complete symbolic debugging
  - Sourcetext debugging
  - Line assembler, disassembler
  - comfortable Help function
  - Macros
  - Watch function for memory contents and variables
- Supports file formats of several compilers
- Connection to personal computer via serial or parallel interface

# EMC

Hint: The **BICEPS-**Emulator is a device for testing of electronic circuits. It is designed for the cooperation with such circuits and therefore no device to operate standalone according to §5 point 5 EMVG (German EMC Law). It is restricted to exclusive use by development laboratories or other enterprises with know-how in the field of EMC.

We explicitly indicate that according to §5 point 6 EMVG at development of devices, testing and installation precautions have to be taken to avoid interferences of third parties. Since under certain circumstances dejam measures must be eliminated (e.g. remove the cover) to execute the test, suitable measures for the complete test arrangement have in case to be provided.

## Attention !

The **BICEPS** hardware and the **BicWin**-software are protected by copyright. A copy of the software is only permitted for backup purposes and for own use; a transfer to others is explicitly prohibited. Software updates and support are given only to customers of **BRENDES DATENTECHNIK GmbH.**

## HINT !

You will find descriptions of current changes of software as well as additional user hints to this manual in the ”READ.ME” files on the enclosed system diskette. Kindly notice the contents of these files absolutly.

# 2 Function of the BICEPS-Emulator

## 2.1 Structure of the BICEPS-Emulator

The **BICEPS** emulator consists of seven essential components:

- The **Control Processor** takes over the control of the emulation process as well as the communication with the host computer (PC) via a serial or parallel interface.
- For the **Emulation Memory** 128k bytes are available, which can be configured as
  - 64k RAM program memory and 64k RAM external data memory or
  - 128k RAM program memory (banked)

  The emulation memory can be expanded to 512k RAM.
- The **External Adapter (POD)**, contains the emulation CPU. Using different PODs one can emulate various CPUs of the 8051 family. Bus signals of the CPU are transmitted over a flat cable to the main device. Real time program execution can be interrupted automatically at certain events or manually by user inputs.
- A **32-bit-Real Time Counter** allows exact time measurements of the program executing duration with a solution of 100 ns. A measurement range of up to 12 h is achieved by a switchable time base.
- The bus signals, the external signals, which can be fed-in from the circuit under test by means of test clipses (Logic Probes) as well as the state of the real time counter can be traced in the **Real Time Trace Memory** ❺ In this way the temporal progression of the program run or the access to the external data memory and the execution time of program parts can be represented (also during a running program).

- The **Address Range Selection** is used for marking either addresses or address ranges. For each address of the address space it can be defined whether a memory access of these address resp. this address range
  - shall lead to a program break
  - shall be traced in the real time trace memory
- Besides the possibility of defining address ranges for program breaks, using the **Break Logic** one can also determine conditions for the data bus and for external logic probes. Up to two break events can be defined, which are counted (break counter) and combined in real time. The break events can:
  - interrutp the real time program execution (break)
  - start or stop the real time trace
  - trigger an external device.

## 2.2 An Outline of the Emulation Process

After the initialization phase (the RESET command) the emulation processor transfers the contents of its internal registers to the control processor and is stopped by the latter. Now one can access the contents of the recovered register  as well as those of the emulation memory from the host computer by help of the control processor. Moreover, one can retrieve and alter memory contents, which are accessible only from the emulation CPU (externally mapped memory ranges). Should the emulation processor now execute a user's program, the (eventually changed) register contents are read and a jump into this program follows. After termination of the program's execution the new register contents are again transferred to the control processor.

The simplest form of program processing is execution in single steps. In this case the internal state of the emulation processor is presented after each program step. However, the normal procedure is the execution of a program in real time. Then the emulation processor is started at a certain position of the user's program. The program execution can be stopped either manually by the user or automatically by previously defined break conditions and the state of the emulation CPU is displayed. The break options are presented in detail in chapt. 5.

If the Universal adaptor of the **BICEPS** emulator is used, the emulator must be able to control the processor of the circuit under test. How is this possible without replacing the processor of the circuit as with usual emulators? If the processor executes all program memory accesses via the EPROM socket to the emulation memory of the emulator, then this device has the control, whether a user program to be tested is executed or not. If it is possible to switch at the right moment between user program and  hidden, emulator internal system routines, single step executions and breaks can be realized.

The **BICEPS** emulator possesses a complex monitoring logic, which surveys the program execution in real time and controls the switching to the system routines. The emulator maintains in this way the control about the processor also during program execution. A additional communiction logic permits the transfer of the internal register contents to the PC. No accesses to the external data memory are used for that purpose to avoid restrictions related to the functionallity of the ports.

The use of the **BICEPS** Universal adapter technique combines the advantages of EPROM simulators and in-circuit emulators in an ideal way. Even the support of applications with watch-dog timers are possible with this technique. The user has only to secure that the cernel of the circuit, i.e. the access of the processor to the EPROM socket is correct.

## 2.3 Start of the BICEPS Emulator

The **BICEPS** emulator is started by invoking the program **BicWin**. You will find a detailed description of the **BicWin** operation in the online help of the software. To adapt the **BICEPS** emulator optimal to ist working conditions several setting possibilities are provided.

The configuration is done on the hardware side via jumpers and on the software side via the “Options” dialog of the **BicWin** software. Before you start the emulator for the first time, you have to perform the working steps "Implementation" and "Configuration"; please read to this chapter 2.3.1 or 2.3.2.

When starting **BicWin** all necessary files are loaded. To that belongs moreover the configuration files (containing desktop and default settings information) and the controller derivates definition file BICREGS.TXT (containing the names and addresses of the Special function registers).

Thereafter the hardware components like break-, trace- and map-logic are initialized. The emulation memory stays unchanged, i.e. it contains a random content. The emulation processor performs a hardware reset and transfers the current contents of its register to the PC, which displays them on the screen.

### 2.3.1 Switching On and Start of the BICEPS-Emulator with Processor Adapter (POD)

For the start of the **BICEPS** emulator only a few points have to be noticed. It is recommended to proceed in the here given sequence:

1.) Start the installation program on the disc. It installs the **BicWin** software on your harddisc.

2.) Check, whether the main voltage and the input voltage of the enclosed power supply are in accordance. If this is the case, connect the power supply to the emulator.

3.) Connect emulator and PC by means of the enclosed serial or parallel interface cable (but not both cables). If you use the parallel interface, note that the PC must provide a bidirectional parallel interface in EPP mode (see BIOS setup).

4.) Control, whether the jumper settings of the CPU adapter (POD) conform with the desired mode, in case correct the setting (see appendix C)

5.) Connect emulator and adapter (POD) by means of both enclosed ribbon cable.

6.) Let the emulator switched off and start the **BicWin** software. After a short delay, you will be asked for configuring the emulator. Confirm this questions with “Yes”. You will get the **Bicwin** “Options” dialog. You must set the following options (in the register cards "Interface" and "Processor"). Further options settings as directories, compiler etc. can be done now, but also later after the start of the emulator.

7.) Set PC interface: Select the correct interface name. If you are working with a parallel interface, check wether the interface name (e.g. LPT1) and the hardware address (e.g. 03BC) are correct for your computer.

8.) Set Processor and POD type: this option is very important because the emulator initialization depends on this settings

9.) Switch on the main voltage of the **BICEPS** emulator. The existing main voltage will be indicated by the Power LED on the front of the emulator. Press the reset button on the rear of the **BICEPS** emulator.

10.) Click on the “Ok” button of the **BicWin** options dialog now. The emulator is started and **BicWin** presents the default desktop with Register-, Intdata- and Assembler window. The contents of the Special function registers must contain the reset values (e.g. PC=0000, SP=07). If there are error messages, see appendix D “Solving problems”.

11.) In case the emulator has reported no error, it can be connected now with your test board. For this purpose terminate **BicWin** (Close button or Alt+F4).

12.) Switch off the emulator and plug the adaptor in the socket on your test board.

13.) Switch on the power supply of the emulator and after that the voltage supply of the circuit under test. Start the program **BicWin** again. The emulator has to answer as described under 6.). Now it can be worked with the **BicWin** user interface.

14.) Much success for your work with the **BICEPS** emulator.

### 2.3.2 Switching On and Start of the BICEPS-Emulator Universal Adapter

For the start of the **BICEPS** emulator only a few points have to be noticed. It is recommended to proceed in the here given sequence:

1.) Start the installation program on the disc. It installs the **BicWin** software on your harddisc.

2.) Check, whether the main voltage and the input voltage of the enclosed power supply are in accordance. If this is the case, connect the power supply to the emulator.

3.) Connect emulator and PC by means of the enclosed serial or parallel interface cable (but not both cables). If you use the parallel interface, note that the PC must provide a bidirectional parallel interface in EPP mode (see BIOS setup).

4.) Control, whether the jumper settings of the CPU adapter (POD) conform with the desired mode, in case correct the setting (see appendix C)

5.) Please note, that the Universal adaptor can’t work without your test board. Remove the EPROM of your board. If you intent to emulate a ROM version, then replace the CPU of your board by a suitable piggy-back-CPU. Connect the board and the **BICEPS** emulator by means of the Universal adaptor. Connect the two additional clips cables ALE (input) and RESET (ouput). Please note, that the reset signal must be fed-in correctly (see also chapter C.1.3.4).

6.) Let the emulator switched off and start the **BicWin** software. After a short delay, you will be asked for configuring the emulator. Confirm this questions with “Yes”. You will get the **Bicwin** “Options” dialog. You must set the following options (in the register cards "Interface" and "Processor"). Further options settings as directories, compiler etc. can be done now, but also later after the start of the emulator.

7.) Set PC interface: Select the correct interface name. If you are working with a parallel interface, check wether the interface name (e.g. LPT1) and the hardware address (e.g. 03BC) are correct for your computer.

8.) Set Processor and POD type: this option is very important because the emulator initialization depends on this settings

9.) Switch on the main voltage of the **BICEPS** emulator. The existing main voltage will be indicated by the Power LED on the front of the emulator. Press the reset button on the rear of the **BICEPS** emulator. Switch on the power supply of your test board.

10.) Click on the “Ok” button of the **BicWin** options dialog now. The emulator is started and **BicWin** presents the default desktop with Register-, Intdata- and Assembler window. The contents of the Special function registers must contain the reset values (e.g. PC=0000, SP=07). If there are error messages, see appendix D “Solving problems”.

11.) In case the emulator has reported no error, it can be worked now with the **BicWin** user interface.

12.) Much success for your work with the **BICEPS** emulator.

# 3 The BICWIN User Interface

## 3.1 Survey of the Operation Facilities

In order to facilitate a comfortable operation of the **BICEPS** emulator, the user interface **BicWin** offers multiple functions. The representation is given with in a clear window technique with pull-down menus and status line. All windows can be opened and closed arbitraryly and changed in size and position. The operation is carried out with mouse or keyboard.

General informations to the possibilities of the **BicWin** user interface are presented in the following chapters; more detailed descriptions, especially of dialogs and menus are offered by the Online help of the **BicWin** software.

Following a short survey is given .

1.) All the memory cells accessible from the emulation processor can be represented and altered. For this the windows Extdata, Intdata and Program and the "Modify" functions. "Watch" variables can be defined and presented in special watch windows. With "Copy" memory ranges can be moved and with "Initialize" set to a certain value. With the "Map" function it is defined which memory ranges internally in the emulator and which ones externally on the user board are addressed. There is the possibility to copy the contents of a external EPROM directly into the emulation memory (see chapter 4).

2.) The special function registers of the CPU are displayed in the Register window. They can also be retrieved and changed directly by means of the "Modify" function, respectively (chapter 4).

3.) Execution of a program is initiated by function keys or by the "Debug" menue. In the single step mode there is a possibility of executing the subroutines in real time ("jump over calls").

4.) The program memory contents can be represented in disassembled form. The input of mnemonic instructions will be done with the **BicWin** line assembler.

5.) There is the possibility to set or clear breakpoints in the Assembler or Sourcetext window directly. More complex break conditions (e.g. combination of break events or break counters) are defined via the “Break” dialog.

6.) By means of the "Trace" functions one can either display the contents of the real time trace memory of the **BICEPS** emulator or select various options of tracing data in the trace memory (see chapter 6). Further functions are e.g. searching of certain contents or calculating of time differences.

7.) The memory contents of the emulator as well as the contents of the real time trace memory can be saved on or loaded from a disc. Several standard file formats are supported.

8.) Besides absolute addresses symbolic addresses can be used, which were created by an assembler or a compiler. If the use of symbols can be helpful, this is offered by BicWin with the “?” button.

9.) Sourcetext Debugging allows the test of a program on source text level. The high level language program can be executed in single steps, i.e. line by line. Breakpoints are set directly in the source text. (chapter 8).

10.) All debugging commands are recorded in the “Command” window and can altered and repeated easily.

11.) User-own macros provide the summarization of **BicWin** commands under a free chooseable name. These can be executed by entering the macro name but also automatically at the start of emulator and in case of a break.

12.) To handle several projects, the actual configuration of desktop, break and trace settings etc. can be loaded and saved. A context sensitive Help Function offers additional support at the work with the **BicWin** user interface

## 3.2 General Informations to BICWIN

The **BicWin** user interface consists of a screen divided into four parts:

- in the middle the working plane with windows and dialogs
- above that the selection menu
- below the status line
- the mouse panel with the most important actual functions.

In addition to that every window posseses a local context menu.

All functions can be invoked either with the keyboard or with the mouse.

Opened windows can be moved arbitrarily on the working area resp. enlarged or reduced. Only one window is always selected; this is recognizeable by the highlighted bar.

For all functions and windows short keys (function keys and ALT-key-combinations) are available. The most important are the ALT-keys, with which menu items and windows are selected and the function keys for releasing **BicWin** actions.

The context sensitive **BicWin** help system offers detailed descriptions of every function or windows. In the help window you find key words that can be selected by the mouse and lead you to more help information.

## 3.3 Windows and context menues

A window is a part of the screen, which you can shift, enlarge and reduce, cover with other windows , open and close. Several windows can be opened, but always only one window can be active (recognizeable at the highlighted bar at the head of the window).

Wih the ALT-key combinations a window can aimed to be opened or activated (e.g. ALT+A = assembler window, ALT+1 = watch window 1). With CTRL+F4 a window is closed, with F6 and Shift+F6 the next or the previous window is activated (see also overview short keys).

An own context menu is assigned to every window, which can be activated with the right mouse key or the context menu key. The context menu key is also designated as PopUp key. For this reason the offered key combinations are called "Pop+letter" so e.g. **Pop+B.** Incomparison with the CTRL key the PopUp key has the advantage that it need not be activated synchronously with the letter key and it can therefore better operated with one hand.

The following windows are available:

| Label | Function | ALT key |
|---|---|---|
| Assembler | Disassembled presentation of the program memory | ALT+A |
| Intdata | Hex presentation of the internal data memory | ALT+I |
| Extdata | Hex presentation of the external data memory | ALT+E |
| Program | Hex presentation of the program memory | ALT+P |
| Register | Special function registers of the CPU | ALT+R |
| Sourcetext | Presentation of source text or other text files | ALT+S |
| Trace | Trace memory content | ALT+T |
| Watch1..4 | Output of watched variables | ALT+1 ... ALT+4 |
| Command | Records **BicWin** commands (with possibility to repeat) | ALT+C |

## 3.4 Mouse palette and Status line

In the mouse palette the most important functions, which are just available, are offered. These can either be executed directly by mouse click or by input of the highlighted short key. The mouse palette can be oriented horizontal or vertical.

The mouse palette consists of a general part, which stays unchanged and a locally assigned part, where the offered functions change, depending on the just selected window. For instance the function **F7** (single step) is always available at the same position, while **Pop+B** (break point set/reset) is only offered in the assembler window and in the source text window.

The status line at the lower margin of the **BicWin** window presents the following information:

1. The runtime of the program executed in real-time or in single steps since the last reset of the real-time counter.
2. The status of the external digital inputs (logic probes)
3. A message when the program to be tested is executed.
4. A message about the just executed macro

## 3.5 Dialog boxes

Menu options with dots (...) open a dialog box, in which settings can be shown and altered. A dialog box must at first be closed, before another window can be activated.

In a diealogue window there are different operating elements:

- Action switches: release an action by activating with mouse or keyboard
- Marking and switching fields: switching options on and off
- Input fields: for input of text
- List window: Selection of elements with the mouse or arrow keys

# 4 Accessing the Memory

## 4.1 Basic Information

The **BICEPS** emulator is equipped with various emulation memories according to the achitecture of the emulated processor. The memory contents can be displayed (output) and changed in order to create specific test conditions. Besides this, we must naturally be able to enter or load the programs to be tested, and possibly also modify them.

Processors of the 8051 family possess five different memory ranges altogether:

- the Program Memory
- the external Data Memory
- the internal Data Memory
- the bit-addressable Memory
- the Special Function Registers;

the three latter are partially overlapping. A special **BicWin** window is allocated to each memory range, where the bit-addresable memory is displayed in the Intdata window.

In principle, we can distinguish the following access possibilities for various memory types:

- Loading and saving of memory contetnts
- Displaying and changing the Special Function Registers
- Displaying and changing memory cells (Extdata, Intdata, Program window)
- Modifying of memory cells, registers and variables
- Initializing and copying of memory blocks
- Mapping, i.e. determining whether the RAM internal to the emulator should be accessed or whether we should access the circuit to be tested
- Disassembled presentation of the Program Memory (Assembler window)

- Assembling 8051 Opcodes

## 4.2 Saving and loading files

The **BicWin** user interface offers various functions for loading and saving of memory contents with different formats.

We can load:

- Contents of the program and data memories
- Source text files for highlevel language debugging
- Symbol information

and we can save:

- Contents of the program and data memories
- Trace memory contents

For loading and saving of memory contents four different file formats are available.

**BicWin** can save memory contents of the emulator also as text files, e.g. to evaluate these files with other programsor to edit them with a text editor. Distinguished are

- hexadecimal representation of memory contents (Hexdump)
- disassembled representation of the program memory

It is to be noticed, that significant limits are stated for creating of text files, since the files canotherwise reach a considerable size (several tenthousend lines). At saving the program memory in disassembled form two formats are distinguished:

- in the list format (same contents as in the Assembler window)
- in the sourcetext format the columns with addresses and file contents are omitted, to make the text files usable as source for an assembler

## 4.3 Access to the Special Function Registers

The contents of the special function registers are displayed in the Register window. The names of the registers are preset, and depend on the CPU type selected in the “Options” dialog. In addition to the real special function registers, the display includes the program counter PC, and the register bank R0..R7.

The registers are separated in groups; each group has an own box with title bar (e.g. Timer). The groups can be moved inside the register window; so even if only a small register window is visible, the most interesting registers can be displayed. The names, addresses and groups are contained in the external file “BICREGS.TXT”, which can be edited if necessary.

The arrow keys or the mouse can be used to select a register and open the “Modify” dialog to enter a new content. Ports are set at once, i.e. the new content appears at the port pins immediately after entering the new value.

Attention:

> The internal data memory and the special function registers overlap in the address range 80H..0FFH. In the Intdata window no special function registers can be accessed, but only the internal RAM cells of the processor (if there are any).

## 4.4 Access to the Program Memory, to the internal and external Data Memories

Memory contents are represented in hexadecimal form in the Extdata, Intdata and Program windows. By means of the arrow keys we can select specific addresses and set at a new value, with the “Got to” function it can be jumped to a new address.

Altogether three fields can be distinguished: the address field, the hexadecimal field and the ASCII field. The cursor can be moved to any position of the hexadecimal and the ASCII field. New memory contents in hexadecimal form or as ASCII character can be entered.

For the individual functions one should keep in mind:

Program:

Entering data is possible only for internal mapped memory ranges.

Extdata:

Entering data i spossible always. If the test board should be directly accessed, then the considered memory range must be mapped externally (default setting).

Intdata:

In the address range 80H..0FFH the internal Data Memory overlaps with the Special Function Registers. The Special Function Registers cannot be accessed in the Intdata window.

## 4.5 Output and Change of single Memory Cells and Registers (Modify)

If a single memory cell or variable shall be aimed accessed without hexadecimal representation of a memory block, the Modify function must be used. This enables also other formats for input and the output, i.e. memory cells can be entered or represented in binary or decimal form, as ASCII characters or as highlevel language variable. Which variable type can be selected depends on the compiler defined under "Options Highlevel".

The function Modify can be invoked directly from the Register, Assembler and Sourcetext window. For that purpose the requested variable must be only selected by a double click with the mouse. If it is a valid variable then one gets the current value immediately and can enter a new value.

As a speciality the function "Modify write only" is still available, which enables the writing to memory cells, without accessing by previous reading the corresponding address. This is necessary e.g. with I/O components, which can change their state already by the reading access.

## 4.6 Watched variables

With the Watch function memory contents and variables can be displayed in up to four different watch windows. Which variable shall be represented is determined  by the “Watch list” dialog. Variable type (e.g. Unsigned) and format (e.g. binary) can be selected.

Up to four windows are available, into which watch variables can be issued (Watch windows 1..4). By the distinction of up to four windows it can be exchanged in an easy way between sets of variables to be issued without altering the definition in the list of watched variables. Only the windows with the desired variables had to be opened or the not relevant windows closed again. For this the key combinations ALT+1 until ALT+4 are provided.

The watch windows are actualized by choice either automatically after each program execution or only at selection of the watch window.

## 4.7 Initializing and Copying of Memory blocks

The function "Initialize" writes a constant value into a memory block, defined by start address and end address.

By means of "Memory Copy" we can move memory blocks, i.e. copy them. The source and target areas may overlap. As extension hereto there is the function "Copy EPROM", which transfers the program memory contents from the user's circuit into the Emulation RAM (see chapter 4.8 "Mapping").

## 4.8 Mapping

The memory types program memory and Extdata memory can be mapped. This means that the user can determine with help of the "Memory Map"-function wether

- the RAM internal to the emulator (emulation memory), or else
- a memory possibly present on the circuit to be tested (e.g. an EPROM)

should be accessed.

The default setting for map is an internal mapped program memory and an external mapped Extdata memory.

The **BICEPS** Universal adaptor offers as well a mapping possibility. This is true only for the program memory, the Extdata memory is mapped external always. To access the original EPROM, the EPROM of the test board must be removed, replaced by the adaptor and be plugged on the adaptor. This means, that an emulation is possible with the original CPU and the original EPROM.

## 4.9 Disassembled representation of the program memory

The Assembler window gives out the contents of the Program Memory in the form of executable commands, i.e. in disassembled form. The code undergoing this representation is always the code executed by the emulation processor, i.e. - depending on the relevant map configuration - the contents of either a memory internal to the emulator or an external one.

One can move through the memory forwards and backwards, using the cursor keys. The line with the current program counter position, i.e. the command to be executed next, is highlighted.

Besides the the representation of the program additional functions are possible:

- Execution of the program
- Set and clear of breakpoints and trace filter ranges
- Modifying of memory cells and variables

For more information see chapter 7.

The **BicWin** Line Assembler (integrated in the Assembler window) serves the comfortable input of 8051 program code into the program memory RAM. Before entering a mnemonic command, the command currently present in the program memory is displayed in  disassembled form. Entering code is only possible in internal mapped memory ranges.

For denoting the addresses of the program memory, the bit memory and the internal and external data memories one can use arbitrary symbols - provided that they are known, i.e. entered into the corresponding symbol table. The predefined symbols known to the **BicWin** Line Assembler are the names of the special function registers (bit and byte addresses). They correspond to the type of processor selected in the “Options” dialog. Furthermore complex arithmetic expressions can be formed at the declaration of addresses and constants, which can contain symbolic names as well as number constants.

# 5 Break possibilities

## 5.1 Basic Information

Breaks are defined in order to stop the program execution of the emulation CPU at certain conditions, interesting for the program test. For example, these condition may be the execution of a certain part of the program, or an access to some specific ranges of the Data Memory. The **BICEPS** emulator offers the following possibilities for defining a break condition:

- Access to one or several addresses or address ranges (64k breakpoint memory)
- State of the data bus by accesses to the data memory and program memory
- State of the external inputs (Logic Probes)
- Event counter for breaks (Break Counter)
- conditional breaks
- Overflow of the real time trace memory

The defined state of the address bus, the data bus and the logic probes forms a break event, where the combination is predetermined

(addresses AND data AND Cycle type)

An event is additionally connected with an event counter (break counter).

Altogether there are two break events “Break1” and “Break2”, which can generate a break alone or together. The following break actions are possible:

- interruption of the real time program execution (break)
- enabling the second break event for conditional breaks
- start or stop of the real time trace
- triggering external devices.

## 5.2 Break mode

“Break1” and “Break2” can be combined in several ways. Beside OR and AND combination there are conditional combinations, that allows to enable one break event by the otherone. Furthermore there are the tow events “Clear1” and “Clear2”, which can reset detetcted events “Break1” or “Break2”. In this way it is possible for example to define an event, which is only enabled in a special subroutine.

The events “Break1” and “Berak2” are available as signal outputs to trigger external devices (see chapter “Interfaces”).

In the **BicWin** help function you can find several examples of the break and controlling possibilities of the **BICEPS** emulator.

In addition to states of the address and data buses, we can also define a state of the external inputs as a break event. When the signals at the external inputs become effective and what qualities they must possess is described in chapter B.2. The definition is made as binary combination, i.e. as a sequence consisting of the symbols "0", "1" and "x", where to the external inputs 12 characters (corresponding to the 12 inputs) are assigned.. "x" stands for don't care, these bits are not rated.

## 5.3 Definition of Break Events

The most important function of the break logic of the **BICEPS** emulator is interrupting the program execution at a specific position of the program. This position is defined by its address.

For a break events “Break1”, “Break2”, “Clear1” and “Clear2” one can define as many addresses or address ranges as one likes. By the definition of the “Cycle type” it is possible to distinguish between Program and Extdata memory accesses; in addition a specific data bus state can be defined. A defined break address is only then valid when during the program execution the defined state of the data bus occurs at the same time.

For accesses to the program memory the options “Opcodes” (default), “Line numbers” and “All accesses” are available. "Opcodes" means, that a break is only then recognized when the first byte of a command is executed (independent of of the value of the read date). This is a very important option, since CPUs of the 8051 family access addresses, partly unnecessarily, and reject the read data afterwards. With “All accesses” this restriction is omitted; with “Line numbers” the opcodes cycles are restricted to sourcetext line beginnings.

The data bus conditions Read and Write contain the ranges 00-FF as presetting, i.e no restriction to a specific state. Each reading or writing access to an address of the external data memory leads to a valid break event, as far as the address is marked as break address.

Each of the possible break events is assigned an event counter (break counter). In order to effect a break, an event defined by the states of the address bus, the data bus and the logical probes must occur as often as it is predetermined in the break counter. Preset is a value of 1, i.e. the event is immediately valid at its first occurrence. For “Break1” a 24 bit counter is available, but this is reduced if break counter capacity is neede by other events.

## 5.4 The Onbreak Function

Normally, after a program execution has been aborted by a breakpoint, the **BicWin** user interface awaits a new command from the user. However, since the commands following a break are often very similar (e.g. output of the contents of a specific memory cell, execution of several single steps etc.), **BicWin** offers the possibility of automation of specific command sequences.

Any **BicWin** command or any macros can be defined as Onbreak command. In this case, such an Onbreak command will be executed immediately following a break or an execution of a single step. Only then new commands can be entered again.

If the Onbreak command is a macro containing the "Run" command as its last element, then the emulation processor is started up again. In this way we can obtain an automatic cyclic processing of the user's program fragments and **BicWin** commands. For example, each time when the emulation CPU executes a specific subroutine, a certain memory range can be saved on a diskette.

# 6 The Real Time Trace Memory

## 6.1 Overview

The **BICEPS** emulator can be provided with a Real-Time Trace Memory of a 32k x 72 bit capacity. It is used for recording certain signals during a real time program execution, similar to a logic analyser. Additionally existent is a 32 bit real time counter, which runs synchronously with each program execution and therefore enables exact time measurements.

The data recorded in the trace memory are:

- 16-bit address bus signals
- 8-bit data bus signals
- 12-bit external signals (logical probes)
- 32-bit position of the real time counter (time stamp)
- control signals
- break events “Break1” and “Break2”

On the basis of the traced information one can easily obtain an overview of the run and the time duration of a program or the external signals.

The real time trace memory of the **BICEPS** emulator can store up to 32766 entries. Each entry has a size of 72 bit and is named a "frame". The frame number is jointly represented in the trace window, where number 0 corresponds to the oldest entry and the largest frame number to the entry traced at last.

Traced are all program steps, i.e. single steps as well as program fragments executed in realtime.

The contents of the real time trace memory is displayed in the “Trace” window. This is also possible during the program execution, without influencing it ("on the fly" access).

The real time trace memory of the **BICEPS** emulator offers the following functions:

- representation of the traced data either as program or as sequences of cycles with time stamp and state of the external inputs
- show context in the Assembler or Sourcetext window while scrolling in the trace window
- calculating of time differences of the traced time stamp
- search functions for traced data
- Sourcetext trace mode (tracing of source text lines)
- selecting of address ranges and access modes for the tracing
- real time filter to enable/disable memory address ranges for tracing
- start and stop of tracing by break events
- Program break by overflow of the trace memory with a predetermined number of frames to be traced
- Performance analysis, i.e. calculating the execution time for different parts of the program

## 6.2 Time Measurements

The **BICEPS** emulator possesses a 32 bit real time counter, which counts timing marks during a program execution.The time base is switchable between

- 100 ns (measurement range up to ca. 7 min)
- 1 µs (measurement range up to ca. 70 min)
- 10 µs (measurement range up to ca. 12 h)

The current position of the real time counter is displayed in the status line. It can be set directly to 0.0 with the function "Reset real time counter" or together with a CPU reset with the function "Debug Reset all". The status line displays the time of program execution since the last reset of the real time counter.

Time measurements come in connection with the real time trace memory, since the position of the counter is traced as time stamp. The time of program execution yields to the difference of traced time stamps. Since their calculation from absolute values is too troublesome, the trace window offers several possibilities for forming of time differences:

1. The time stamp at the current cursor position in the trace window can be set to 0. All following time stamp entries are represented then relative to this zero mark.
2. The multiple manual setting to zero of the time stamp can be simplified with the time filter function. In the time filter dialogue conditions can be defined to which the continous time stamp representation starts again with 0.0. Is e.g. the address of a subroutine entered as condition, then the duration of execution of this subroutine is automatically represented in the trace memory window. Besides addresses also conditions for the data bus or the external inputs can be defined as time filter condition.

## 6.3 Search function

A complex search function allows to find those information in the large amount of traced data, which are interesting for the debugging. Default setting is no restriction, that means all traced frames met this condition. By restricting the search conditions (e.g. address range 1000-1020 instead of 0000-FFFF) the search function becomes more sensible.

In addition to address and data bus contents it is possible to search changes in break events “Break1” and “Break2”.

## 6.4 Sourcetext Trace Mode

In the sourcetext trace mode only the execution of lines of a loaded highlevel language program is traced instead of the complete program run. Assumption for this  is, that a source text as well as the corresponding symbol information were loaded (see chapter 8).

Each line of a highlevel language program is related to a memory range of the program memory. Since in the sourcetext language trace mode only the start of this address range is traced, the real time trace memory can trace up to 32766 sourcetext lines.

When scrolling in the Trace window, it is possible to display automatically the part of the program in the Assembler and Sourcetext window, which belongs to the actual selected trace address.

## 6.5 Controlling the real time trace

The tracing is controlled by trace filter conditions. Single addresses or address ranges can be defined; also symbolic names, in particular module names, are admissible there. Only accesses to the addresses defined in this way are traced in the real time trace memory of the **BICEPS** emulator.

Three real time filters are available. Default setting is one filter for each cycle type, i.e.

- accesses to the program memory
- writing accesses to the external data memory (WRITE)
- reading accesses to the external data memory (READ).

For the program memory filter “Opcodes”, “Line numbers” and “all accesses” are distinguished.

It is possible to switch from one filter to another during program execution, controlled by a break event.

The BICEPS emulator allows to start or stop the real time tracing at predefined events. For this purpose break events are used. It can be defined, which action should be released by break events. Possible are:

1. Start of tracing at program execution start
   Stop of tracing at program execution end
2. Start of tracing at program execution start
   Stop of tracing at an event
3. Start of tracing at an event
   Stop of tracing at program execution end
4. Start of tracing at an event
   Stop of tracing at another event
5. Start of tracing at program execution start
   Switch to another filter at an event
   Stop of tracing at program execution end

# 7 Program Execution

## 7.1 Overwiev

The **BICEPS** emulator offers various functions for executing a user's program.

- Execution of a program in real time (Run)
- Execution of a single step
- Execution of a single step, but with the condition that subroutines are executed in real time  (Jump over calls)
- Generation of a hardware reset of the emulation CPU (Reset processor)
- Execution of a program in real time until a break address is reached
- Set and clear of breakpoints

## 7.2 Execution of a program in real time, Reset

One of the essential functions of an in-circuit emulator is the execution of programs under real time conditions until the program's run is stopped. In the simplest case the break follows after executing a single step of the program (Single Step). However, the real time relation can be obtained only by executing several program steps.

By means of the "Run"-function we initialize the break logic and start the program.

The execution of a program in real time is indicated by a light diode in the field "Running" on the front panel of the emulator. The real time trace memory can be accessed, while the program is executed, and its contents displayed without disturbing the program execution. An interruption of the program can be performed

- by the user with the mouse or the keyboard or
- by occurrence of a break condition.

After a break, the actual values of the open windows and the status line are displayed. Very important for program debugging are the register window with the actual PC and registers and the actual part of the program in the Assembler window. After a break, an access to all memory ranges is possible in order that the current memory contents could be changed.

The function "Reset processor" resets the emulated CPU by executing a hardware reset.

Attention: The **BICEPS** emulator supports the feeding of an external reset signal. If an external reset signal is constantly active, no program execution is possible. Such a situation can be recognized by observing that the "Status" diode is not lighted, though the processor has been started up. (compare with section 19.2)

## 7.3 Execution of single Steps

The emulated CPU can execute a user's program either in real time or in single steps. In the first case, the program is started up and executed until a break occurs. In case of single steps, the internal state of the processor is displayed after each program step.

The "Jump over Calls" function allows to execute single steps with executing of subroutines in real time. This means: if the next command to be executed is a CALL opcode, then a break point is set immediately after this opcode, and the program is started up in real time; otherwise a normal single step is executed. If the processor fails to return from the subprogram, then the program execution must be stopped by the user.

Single steps are traced in the real time trace memory as programs executed in real time are.

## 7.4 Debugging with the Assembler and Sourcetext window

The most important debugging functions can be executed and monitored directly in the Assembler and Sourcetext window. This functions are:

- Reset of the processor
- Execution of a single step, i.e. an assembler opcode or a sourcetext line
- Single step with "Jump over Calls"
- Execution of the program in real time
- Program execution to the cursor, i.e. the command marked by the cursor is provided with a breakpoint and the program is started
- Setting the program counter PC on the command marked by the cursor
- Setting and clearing of breakpoints
- Marking of program ranges and setting/clearing of trace filters for this program ranges. By this feature it is possible to disable the tracing of program loops whose execution leads to an overflow of the trace memory
- Opening the “Modify” dialog by a double click of the mouse with the selected variable name.
- representation of another part of the program

The most of this functions can be executed directly by a click with the mouse in the mouse palette or by a function key. The remaining functions can be activated via the local context menue.

In combination with the Trace window it is possible to show the actual part of the program in the Assembler or Sourcetext window when scrolling in the Trace window. The address of the just selected trace frame is shown as highlighted bar in the Assembler or Sourcetext window. The user can watch the traced program flow in a representation with a moving bar just like the execution of single steps.

By deactivating the option “Follow program counter” it is possible to freeze a representation of the Assembler or Sourcetext window. The shown part of the program isn’t adapted when the program counter is changed by execution of the program.

The following lines are shown highlighted:

1. the actual program counter address
2. breakpoints
3. disabled trace filter ranges
4. the address of the just selected frame in the trace window
5. marked areas.

# 8 Sourcetext Debugging

## 8.1 Introduction

The **BicWin** user interface supports a program test on the base of highlevel language debugging. This  means that the source text of programs, which is written in a highlevel language (e.g. C51) or in assembler language, can be loaded by **BicWin** and is presented during the debugging process.

The sourcetext is shown in the Sourcetext window. All debugging features known from the Assembler window are available in the Sourcetext window, too (e.g. execution of the program line by line). In the real time trace memory sourcetext lines instead of assembler opcodes can be traced. Highlevel language variables are represented and altered corresponding to their type definition.

The switching between Assembler and Sourcetext window makes it possible to execute a program in different “resolution”.

The assumption is, that the relation between the line numbers of the original source text and the addresses of the executable code is known. The used compiler or assembler software must generate the necessary symbolic and line number informations.

## 8.2 Invoking the Compiler, Loading a Program

The **BicWin** user interface uses different files for handling sourcetext during debugging. They are:

- a file with the program code and the symbolic information
- one or more files containing the source text

For the cooperation with the **BicWin** software it is important to correctly select the various options of the compiler, so that the program and symbol files could be created in a form readable by the emulator. Very important are the line number symbols because they are needed for the relation of program code and source text line. To access highlevel language variables, symbolic names and typedef informations are needed. Check in y<our compiler manual, which options must be set to generate this informations (e.g. Keil-C51: objectextend). The function “Load status” of the **BicWin** software shows how many symbol names and line numbers were loaded.

After starting up the emulation software **BicWin**, a program to be tested is loaded by means of the function "File Load”. This function loads the program file as well as the symbol file and the source text, if source text informations are contained in the symbol file. More sourcetext modules are loaded by the function “File Open”. It should be noticed that the program file and the source text are loaded from the directories determined under "Options Directories".

## 8.3 Program execution

The loaded source text is represented in the Sourcetext window. Every line of the text can be selected by the mouse or the cursor keys. The line with the actual program counter, i.e. that line of the source text, which is to be executed next, is optically highlighted.

The Sourcetext window offers the same functions for executing a program and for debugging as the Assembler window (see chapter 7.4). The execution of a program in a high-level language does not differ in principle from that of an assembler program. The essential differences are the representation of the current program sector and the execution of single steps.

Breakpoints can be set only on the lines which have been translated into the corresponding program code by the compiler, so they cannot be set e.g. on empty lines, or on comment lines. If either the program counter of the CPU or a breakpoint cannot be set on some line, since no corresponding line number symbol for this line has been generated by the compiler, then an error message is given.

It can always be switched between Assembler window and Sourcetext window and therewith between the highlevel language representation and the assembler one. We can also determine in this way whether in case of single steps an assembler command is executed or whether the program will be executed line by line (i.e a single step corresponds to a line of the source program). Highlevel language single steps are only executed, when the Assembler window is not selected and the program counter points to a line number.

## 8.4 Handling of several modules

If the program is composed of several source text files (modules), more than one text files can be represented in the Sourcetext window. With the function “File Open” every text file can be loaded.

If the option “Follow program counter” is activated (default setting), more sourcetext files are reloaded automatically when the program counter points to another module after a break or a single step.

More than one Sourcetext window can be opened. In this windows the option “Follow PC” is switched off to enable one windows for the actual program section and other windows for fixed program representations.

Switching between different modules is possible only, if sourcetext file names and module names are contained in the symbol table.

# Appendix

# A BICWIN short keys

## A.1 Function keys

| | | ALT | STRG | UMSCHALT |
|---|---|---|---|---|
| F1 | Help | | | |
| F2 | CPU Reset | Reset all | Clear trace | Macro short key |
| F3 | Search Again | | | Macro short key |
| F4 | Run to Cursor | Terminate | Close window | Macro short key |
| F5 | Go to | | Go to symbol | Macro short key |
| F6 | Next window | | | Previous window |
| F7 | Single step | | | Macro short key |
| F8 | Jump over Call | | | Macro short key |
| F9 | Run | | | Macro short key |
| F10 | Menue | | | Context menue |

## A.2 ALT-keys

| Taste | Funktion | Typ |
|---|---|---|
| Alt+A | Assembler | (window) |
| Alt+B | Break | (dialog) |
| Alt+C | Command | (window) |
| Alt+D | Debug | (menue) |
| Alt+E | Trace | (menue) |
| Alt+F | File | (menue) |
| Alt+H | Help | (menue) |
| Alt+I | IntData | (window) |
| Alt+K | Break | (menue) |
| Alt+M | Macro | (menue) |
| Alt+N | Window | (menue) |
| Alt+O | Options | (menue) |
| Alt+P | Program | (window) |
| Alt+R | Register | (Fenster) |
| Alt+S | Sourcetext | (window) |
| Alt+T | Trace | (window) |
| Alt+V | Views | (menue) |
| Alt+W | Watch | (menue) |
| Alt+X | ExtData | (window) |
| Alt+Y | Memory | (menue) |
| Alt+1 | Watch 1 | (window) |
| Alt+2 | Watch 2 | (window) |
| Alt+3 | Watch 3 | (window) |
| Alt+4 | Watch 4 | (window) |

# B Interfaces

## B.1 Emulation POD Connector

Most signals of the Emulation CPU are connected directly with the Emulation Connector. The following lines are buffered:

- P0.0 ... P0.7 (Port 0)
- P2.0...P2.7 (Port 2)
- PSEN, ALE and P3.6 (only if it is used as WR signal).

These and the following information are true for processor adapters (PODs) only. If an **BICEPS** Universal adaptor is used see appendix C.1.

### B.1.1 External Reset

The **BICEPS** emulator supports the feeding of an external reset signal. This means that during a real time program execution the processor can be reset from outside. The break logic of the emulator stays nevertheless active.

If the reset signal is always active, no program execution is possible. This can be recognized if the “Running” LED does not illuminate though the processor was started. If in this case the program execution will be interrupted manually, then the message "Emulation CPU can't be interrupted. Hardware Reset performed" will appear.

### B.1.2 External Crystal, External Vcc

At XTAL1 and XTAL2 an external crystal may be connected, however, also a digital clock may be fed in. Via Vcc the adaptor can be supplied by the user's circuit, but in the normal case the supply voltage is obtained by the emulator over the flat cable .

## B.2 Connector “Break/Trace”

**BICEPS** emulators which are equipped with a real time trace memory offer 12 external digital inputs (logic probes) for tracing external signals or generating breaks by an external signal (see chapter 5 and 6).

All input signals are synchronized with the CPU clock. That means that the external inputs are latched by the falling edge of the signals /PSEN, /RD or /WR. A break can generated only, when the input signals were latched. The inputs are TTL and HCMOS compatible and are provided with a 4,7k PullUp resistor.

In addition to the 12 signal inputs two ouput signals for triggering external devices are available. These two signals correspond with the break events “Break1” and “Break2”. Please note the following specialities:

- the signal “Break1” drives also the “Trigger” LED at the front panel of the emulator
- the signal “Break2” is inverted to enable an activ low triggering of devices.

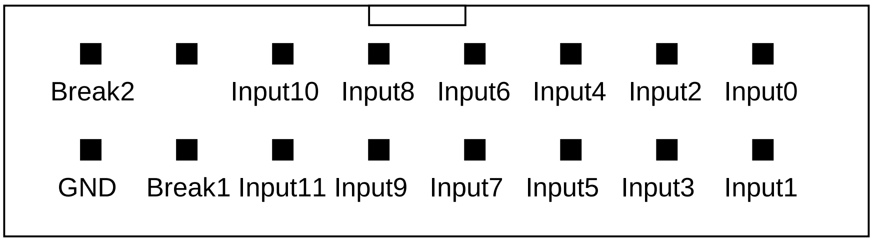


Fig. B.1 Connector „Break/Trace“

## B.3 RS232C Interface (9 pins)

Only three lines of the RS232C interface of the host PC are connected with the **BICEPS** emulator, namely RxD and TxD (data lines) and GND.

The communication format and baud rate are set automatically

## B.4 Parallel Interface (25 pins)

The parallel interface of the BICEPS emulator works bidirectional for a higher communication speed. The Personal Computer must provide a bidirectional parallel interface (EPP mode), too. This is true for modern computers (check BIOS setup); older computers an parallel interface card must be used.

## B.5 Power Supply

The BICEPS emulator is supplied over the power socket by the provided power supply unit. It generates a stabilized 5V voltage. Please take care of the correct mains voltage (in normal case 230V).

# C Specialities of the Adaptors for different CPUs of the 8051 family (PODs)

## C.1 The Universal Adapter of the BICEPS emulator

A speciality of the **BICEPS** emulator is realized with the Universal adapter. This is provided for two application areas:

1.) Connecting the bus signals of the controller without the necessity of replacing the processor (e.g. with PCBs in SMD technique). The Universal adapter has the same pin layout as a standard memory circuit. So it can be plugged in a socket of an external program memory (EPROM) for example.

2.) Emulation of ROM versions by means of a piggy-back CPU.

Besides the signals of the EPROM socket two further signals are of importance:

- The emulator get control about the processor on the circuit via a reset output. The RESET signal has to be fed into the circuit or connected to the RESET pin of the processor.
- The ALE signal of the processor must be taken off.

These two necessary connections are realized by means of two additional cables (with test clipses), which are connected to the Universal adapter.

<u>Please notice:</u>

- A system is assumed, where the data bus of the processor and the EPROM socket are connected directly, so that by means of the ALE signal the lower address bits can be produced out of the data bus signals.
- Since the Universal adapter is directly, i.e. without buffering, connected to the local data bus of the circuit under test, errors with noise loaded systems cannot be excluded. In such cases an adapter with emulation processor must be used.

On the upper side of the adapter a socket is provided, into which the EPROM originally existing on the pcb under test is plugged-in, but is now replaced by the Universal adapter. The Universal adapter can be use as adapter between pcb and original EPROM. With the map function the program memory can be switched over to this external EPROM; in this way the original processor and the original EPROM is emulated.

### C.1.1 Pin-Configuration

The address bus signals A0...A15 are taken via the EPROM adapter, the data bus signals D0...D7 as well as the /PSEN signal via the /CS or the /OE pin of the EPROM. For this a DIL plug with 28 pins is provided. PLCC versions can be adapted by means of a corresponding PLCC converter DIL28-PLCC32 or DIL32-PLCC32. The following pin configurationis based upon:

EPROM up to 64k, DIL28, PLCC32 (nc 1,12,17,26):

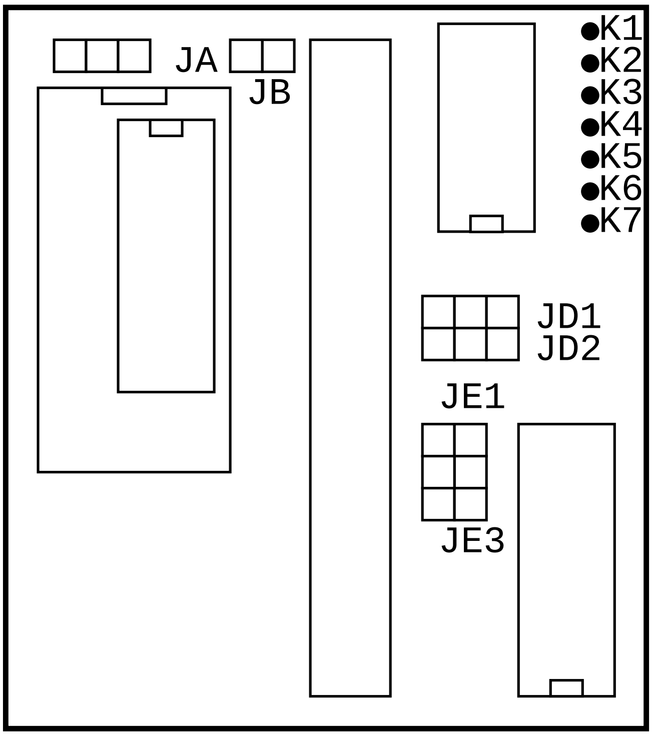
| Pin number | | Signal | | Pin number | Signal |
|---|---|---|---|---|---|
| DIL | PLCC | | DIL | PLCC | |
| 1 | 2 | A15 | 28 | 32 | |
| 2 | 3 | A12 | 27 | 31 | A14 |
| 3 | 4 | A7 | 26 | 30 | A13 |
| 4 | 5 | A6 | 25 | 29 | A8 |
| 5 | 6 | A5 | 24 | 28 | A9 |
| 6 | 7 | A4 | 23 | 27 | A11 |
| 7 | 8 | A3 | 22 | 25 | /OE |
| 8 | 9 | A2 | 21 | 24 | A10 |
| 9 | 10 | A1 | 20 | 23 | /CS |
| 10 | 11 | A0 | 19 | 22 | D7 |
| 11 | 13 | D0 | 18 | 21 | D6 |
| 12 | 14 | D1 | 17 | 20 | D5 |
| 13 | 15 | D2 | 16 | 19 | D4 |
| 14 | 16 | GND | 15 | 18 | D3 |

EPROM larger than 64k, DIL32, PLCC32:

| Pin number | | Signal | Pin number | | Signal |
|---|---|---|---|---|---|
| DIL | PLCC | | DIL | PLCC | |
| 1 | 1 | | 32 | 32 | |
| 2 | 2 | | 31 | 31 | |
| 3 | 3 | A15 | 30 | 30 | |
| 4 | 4 | A12 | 29 | 29 | A14 |
| 5 | 5 | A7 | 28 | 28 | A13 |
| 6 | 6 | A6 | 27 | 27 | A8 |
| 7 | 7 | A5 | 26 | 26 | A9 |
| 8 | 8 | A4 | 25 | 25 | A11 |
| 9 | 9 | A3 | 24 | 24 | /OE |
| 10 | 10 | A2 | 23 | 23 | A10 |
| 11 | 11 | A1 | 22 | 22 | /CS |
| 12 | 12 | A0 | 21 | 21 | D7 |
| 13 | 13 | D0 | 20 | 20 | D6 |
| 14 | 14 | D1 | 19 | 19 | D5 |
| 15 | 15 | D2 | 18 | 18 | D4 |
| 16 | 16 | GND | 17 | 17 | D3 |

The Universal adapter is supplied with voltage by the emulator, i.e. both the supply voltages are not connected via the Vcc pin of the EPROM socket.

## C.1.2 Jumper Settings



At the cable junction K the following signals can resp. must be connected:

K1: not used
K2: not used
**K3: ALE (necessary)**
K4: RESET-In (optional)
K5: WR (optional)
K6: RD (optional)
**K7: RESET-Out (necessary)**

### C.1.2.1 Emulation of ROM Versions (Jumper A)

Jumper A to the outside (to the PCB corner)

Use of a piggy-back CPU for the emulation of ROM versions

Jumper A to the inner side

Use of Romless CPU with external EPROM (Presetting)

### C.1.2.2 Polarity of the external Reset Signal (Jumper B)

This jumper is only necessary if an external reset signal shall be fed-in via the additional cable junction K4.

Jumper B set:

external reset signal RESET in positive logic (e.g. standard 80C51) shall be fed-in

Jumper B not set:

no external reset signal or an external reset signal /RESET in negative logic (e.g. 80C535) shall be fed-in (presetting)

### C.1.2.3 Connecting the PSEN Signal (Jumper D1)

Jumper D1 to the inner side (to the cable):

/PSEN signal is existing at pin 22 of the EPROM socket (/OE) (presetting)

Jumper D1 to the outside (to the edge of the PCB):

/PSEN signal is existing at pin 20 of the EPROM socket (/CS)

### C.1.2.4 Polarity of the Reset Output (Jumper D2)

Jumper D2 to the inner side (to the cable):

negative logic of the reset-out signal /RESET, i.e. LOW-level leads to reset of the CPU (e.g. for 80C535)

Jumper D2 to the outside (to the pcb edge):

positive logic of the reset-out signal RESET, i.e. HIGH level leads to reset of the CPU (e.g. Standard-80C51, presetting)

### C.1.2.5 EPROM-Type (Jumper Field E)

The EPROM type can be defined by means of the jumper field E, i.e. whether the upper address bits are led to the emulator or not. The address bits are evaluated if the corresponding jumpers are plugged-in., otherwise they are put on LOW level. In this way the Universal adapter can be adapted to different EPROM types.

At older versions of the Universal adapter this jumper field is not available, here the selection of the EPROM is made via the setup function of the **BICTOP** software. At newer adapters this is not necessary, here ”not used” is to set in the setup under Universal adapter.

The following jumpers are to place for the following EPROM types:

| | JE1 | JE2 | JE3 |
|---|---|---|---|
| 2764: | open | open | open |
| 27128: | open | open | placed |
| 27256: | open | placed | placed |
| 27512: | placed | placed | placed |

In case that not all jumpers are placed, it is to notice, that from the view of the emulator it can not be distinguished between accesses to the lower and the upper address ranges (e.g. an access to the address 8100H is seen as an access to the address 0100H).

### C.1.3 Connections via Clips Cable

The Universal adapter of the **BICEPS** emulator has six connections for signals, which cannot contacted directly via the EPROM socket. Two signals - ALE (input, K3) and RESET (output, K7) - have to be connected in any case the remaining signals are optional.

#### C.1.3.1 ALE signal (K3)

**Must be connected!**

#### C.1.3.2 Input for external Reset (K4)

The **BICEPS** emulator supports the feeding of an external reset signal. This means, that during a real time program execution the processor can be reset from outside. The break logic of the emulator remains active in spite of this.

If the reset signal is always active, no program execution is possible. One can recognize that the status LED is not illuminated though the processor was started.. If the ”GO” command is broken manually in this case , the message "Emulation CPU can't be interrupted. Hardware Reset performed" is given.

### C.1.3.3 Write and Read (K5, K6)

The connections K5 (/WR) and K6 (/RD) are provided for systems with external data memory. If the signals are connected, accesses to the external data memory can be defined as break condition and traced in the real time trace memory.

### C.1.3.4 Reset Output (K7)

The feeding of a reset signal into the circuit under test is mandatory necessary to control the processor of the circuit. It is to notice here that the Reset signal output is not disturbed by the user board. Admissible is e.g. a voltage monitoring and reset device with an open collector output. Problems can occur with an electrolytic capacitor parallel to the reset signal, it must be eliminated during the test phase.

### C.1.4 ROM Versions with Piggy-Back-CPUs

The emulation of ROM versions is done by means of a so called piggy-back-CPU. This is a special processor variant, which has a socket for the reception of an EPROM on the upper side of its case. The piggy-back-CPU behaves together with this EPROM exactly like a CPU with internal ROM.

The Universal adapter of the **BICEPS** emulator is plugged into the EPROM socket on the upper side of the piggy-back-CPU. In this way the following CPUs can be emulated:

80C51/52, 87C51/52, 89C51/52, 80C154, 87C154

Supported are up to 32 kByte internal ROM/EPROM.

For jumper settings please notice the following:

| | |
|---|---|
| Jumper A: | to the outside (to the pcb edge) |
| Jumper D1: | to the inner side (to the cable) |
| Jumper D2: | to the outside (to the pcb edge) |
| Jumper field E: | E1 eliminated, E2 und E3 placed |

The clips cable ALE und RESET have furthermore to be connected to the CPU (compare also chapter C.1)

In order that the piggy-back-CPU can operate, the same provisions must be fullfilled as with the original ROM version, i.e.

- the CPU must be supplied with 5V
- a clock must exist (external clock or quartz at pin 18 and 19 of the CPU)
- the pin /EA, which defines the operation as ROM version, must lie on HIGH potential

## C.2 BICEPS processor adapters (PODs)

The processor adapters of the **BICEPS** emulator is constructed out of three parts to enable simple adaption to controller type, operation mode and package. Important is the difference between port mode and bus mode emulation. The POD parts are plugged together vertically:

The upper part (adapter part A) defines the operation mode of ports 0 and 2. If this ports are operating in bus mode, e.g. if the application uses an external program or data memory, standard bus drivers can be used on the POD (A1). If Port0 and Port2 are used as I/O pins, the controller must be switched in the so called "Hook" mode. In the hook mode, bus and port informations are multiplexed; so the port contents can be recreated by a special circuit from the Port0- and Port2 signals (A2).

The middle part (adapter part B) consists of the microcontroller, jumpers for setting th eoperating mode and connectors to parts A and C. If the controller is a bondout version, part A1 must be used as upper part, because multiplex between I/O and bus informations is necessary.

The lower part (adapter part C) adapts the POD to the package type of the emulated controller, e.g. DIL, PLCC or QFP.

All jumpers which can be set by the user, are located on the CPU adapter (POD), so the emulator's case need not be opened. The place of the jumpers and their specialities are described in the following subchapters. Information about the Universal adapter you find in appendix C.1.

### C.2.1 Adapter part A1

Standard part for all applications with Port0 and Port2 in bus mode. All circuits have sockets and can be exchanged by the user in the case of a defect. This adapter part is used with bondout versions, too.

### C.2.2 Adapter part A2

POD for applications with Port0 and/or Port2 in port mode. The controller is switched in the "Hook" mode by the emulator. The I/O-signals of Port0 and Port2 are generated by a circuit that has the same quasi-bidirectional Port structure as an 80C51 controller. Pullup resistors, an additional pullup driver for low/high transitions (Port2) and tri-state-drivers if used as external bus (Port0) are provided.

Two operating modes are possible (menu "Options"):
Port0 and Port2 are operating as ports
Port0 is operating as 8 bit bus, Port 2 as I/O-port

### C.2.3 Adapter parts B...

The adapter part B contains the microcontroller and jumpers to define the operating mode

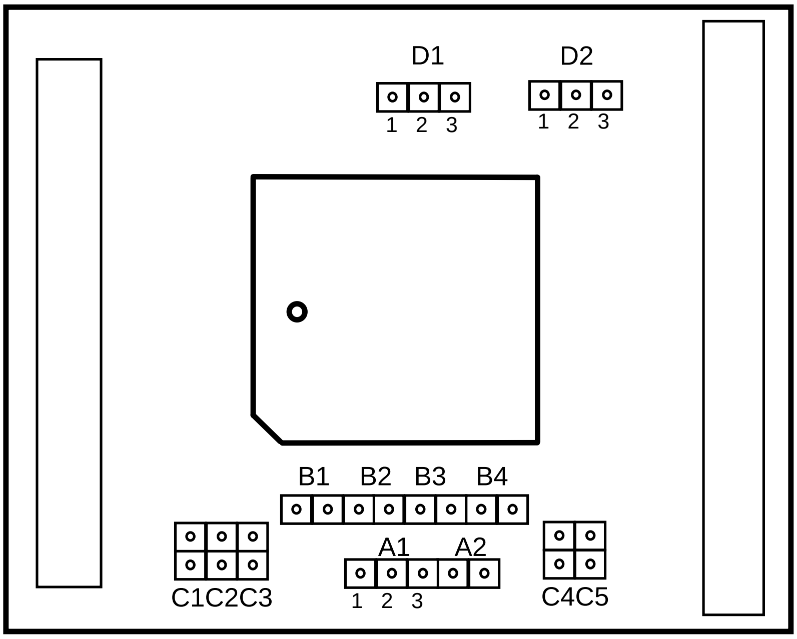
**CPU clock (Jumper A):**
Selection between emulator clock (Oscillator of adpter part A) and external clock. This selection is possible only by jumper and not by software. If bondout- or hook-mode-versions are used, the internal clock must be used.

**RD- and WR-signals (Jumper B):**
For the ports P3.7 and P3.6 of the CPU it must be definedc, wether they are working as I/O-pins or as RD- and WR-signals. In the first case the port pins are connected directly with the board under test without any connection to the emulator logic.

**Power supply of the adapter (Jumper C):**
The external adapter with the emulation CPU (POD) can receive its power supply by the emulator as well as by the user's circuit. In a normal case the former solution is chosen, since in the opposite case one always needs an external power source for starting the emulator.

**Reset signal high or low activ (Jumper D):**
This jumper is only available, if controllers with different reset signals can be plugged in the CPU socket.

## C.3 POD B32-44 for CPUs in a 40/44 pin package (80C31/32, 80C51/52, 80C320, 80C51FA ...)



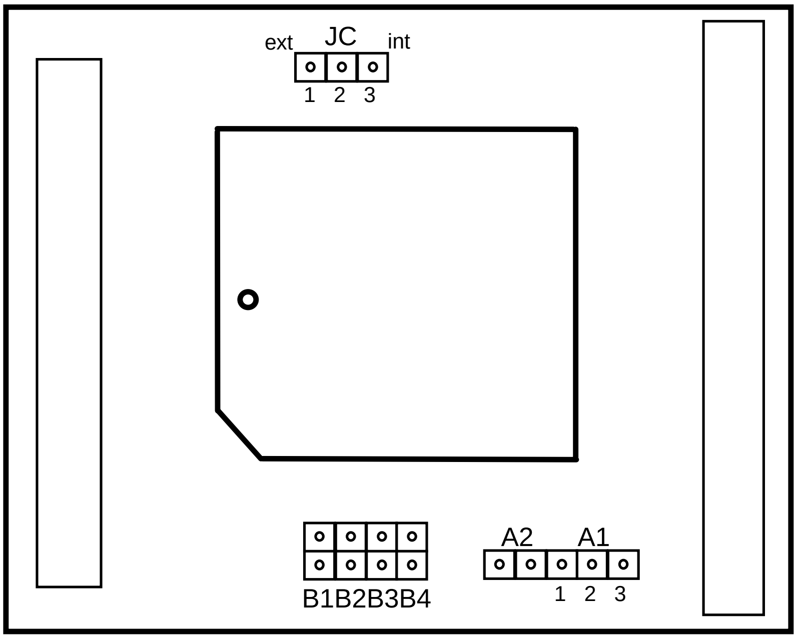
**CPU clock (Jumper A)**
A1=2-3: Emulator clock (oscillator of partA, default)
A1=1-2, A2: external clock or crystal (XTAL pins are connected directly with the board)

**RD- and WR signals (Jumper B)**
B1, B2, B4: P3.7 = RD, P3.6 = WR (default)
B3: P3.7 = Port, P3.6 = Port

**Power supply of the adapter (Jumper C)**
C2, C3, C4: Standard-80C51, internal power supply (default)
C1, C3, C4: Standard-80C51, exteral power supply by the user board
C1, C2, C5: Philips 8x591, internal power supply
C1, C4, C5: Philips 8x591, external power supply by the user board

**Reset logic (Jumper D)**
D1=2-3, D2=2-3: Standard-80C51-RESET, high activ (default)
D1=1-2, D2=1-2: Inverted reset: /RESET, low activ (for example 8xC591)

## C.4 POD B552 for Philips 8xC552

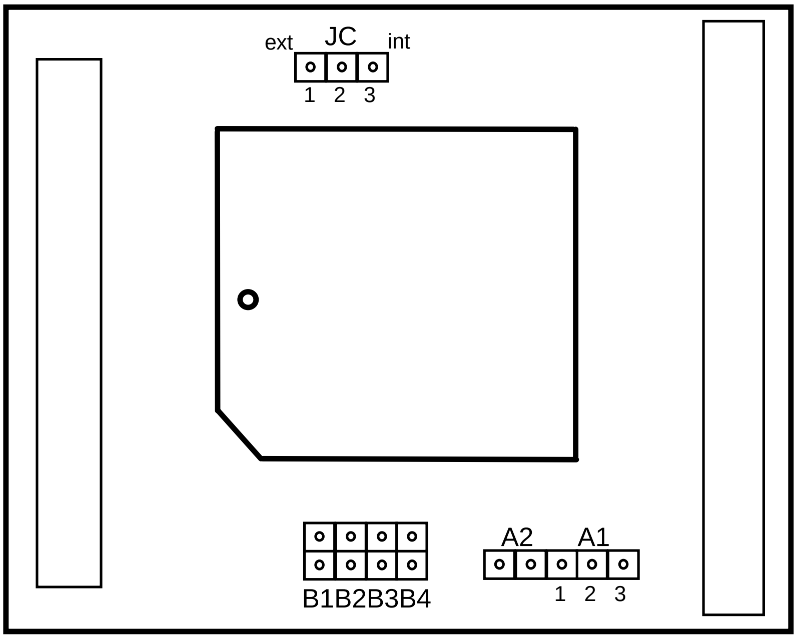


**CPU clock (Jumper A)**
A1=2-3: Emulator clock (oscillator of partA, default)
A1=1-2, A2: external clock or crystal (XTAL pins are connected directly with the board)

**RD- and WR signals (Jumper B)**
B1, B2, B4: P3.7 = RD, P3.6 = WR (default)
B3: P3.7 = Port, P3.6 = Port

**Power supply of the adapter (Jumper C)**
JC=2-3: internal power supply (default)
JC=1-2: exteral power supply by the user board

## C.5 POD B592 for Philips 8xC592

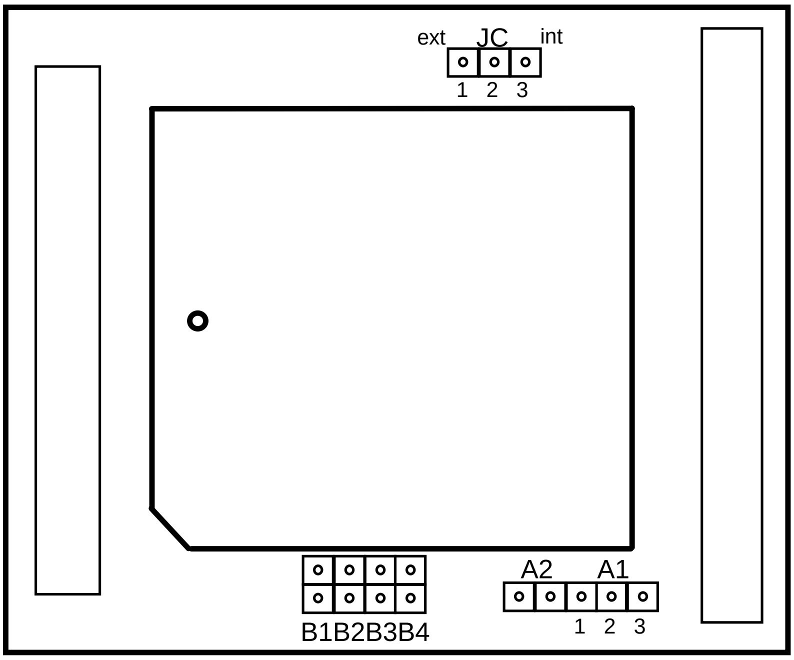


**CPU clock (Jumper A)**
A1=2-3: Emulator clock (oscillator of partA, default)
A1=1-2, A2: external clock or crystal (XTAL pins are connected directly with the board)

**RD- and WR signals (Jumper B)**
B1, B2, B4: P3.7 = RD, P3.6 = WR (default)
B3: P3.7 = Port, P3.6 = Port

**Power supply of the adapter (Jumper C)**
JC=2-3: internal power supply (default)
JC=1-2: exteral power supply by the user board

## C.6 POD B537 for Siemens/Infineon 80C517A/80C537 (with converter also for 80C515A/80C535)



**CPU clock (Jumper A)**

| | |
|---|---|
| A1=2-3: | Emulator clock (oscillator of partA, default) |
| A1=1-2, A2: | external clock or crystal (XTAL pins are connected directly with the board) |

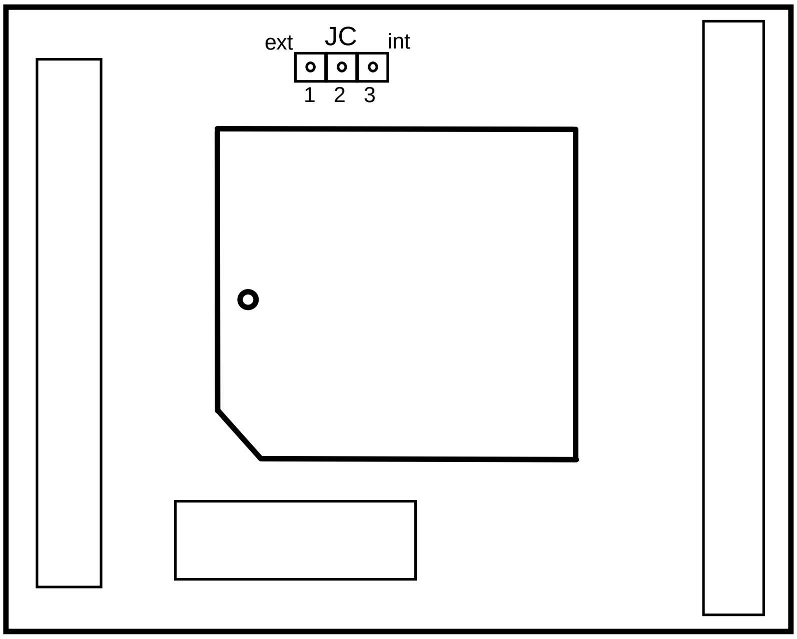
**RD- and WR signals (Jumper B)**

| | | | |
|---|---|---|---|
| B1, B2, B4: | P3.7 = RD, | P3.6 = WR | (default) |
| B3: | P3.7 = Port, | P3.6 = Port | |

**Power supply of the adapter (Jumper C)**

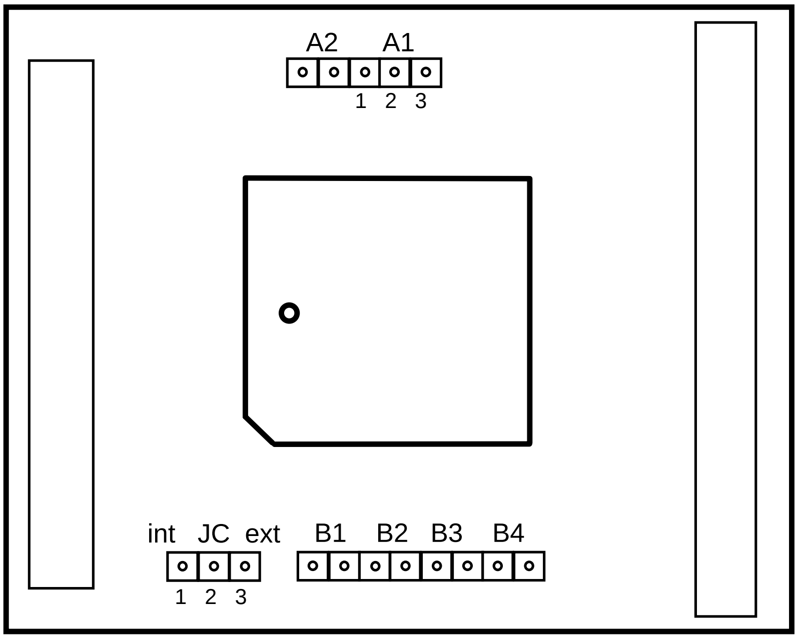
| | |
|---|---|
| JC=2-3: | internal power supply (default) |
| JC=1-2: | exteral power supply by the user board |

## C.7 POD B520 for Dallas 87C520/530



**Power supply of the adapter (Jumper C)**
JC=2-3: internal power supply (default)
JC=1-2: exteral power supply by the user board

## C.8 POD B89C51CC for Temic89C51CC



**CPU clock (Jumper A)**
A1=2-3: Emulator clock (oscillator of partA, default)
A1=1-2, A2: external clock or crystal (XTAL pins are connected directly with the board)

**RD- and WR signals (Jumper B)**
B1, B2, B4: P3.7 = RD, P3.6 = WR (default)
B3: P3.7 = Port, P3.6 = Port

**Power supply of the adapter (Jumper C)**
JC=1-2: internal power supply (default)
JC=2-3: exteral power supply by the user board

# D Solving problems

When the emulator is started or during operation several error messages can be displayed.

**„Emulator not ready“**
Check, wether the power supply is switch on and wether the serial or parallel interface is set correctly. More information to the interfaces you find in appendix B.

**„Processor is not responding“**
The emulation CPU doesn’t answer after a hardware reset. Reasons may be:
If you use a processor adaptor (POD):

- The adaptor isn’t connected correct to the emulator
- There is no CPU on the adaptor
- The power supply is jumpered external (jumper D), but the external power is not connected

If you use an Universal adapter:

- Thea /PSEN signal is not connected correctly (jumper D1)
- The logic of the reset signal is jumpered wrong (jumper D2)
- The reset signal is not connected correctly (see appendix C)
- The power supply of the test board is switched off.
- The necessary conditions for the operation of the piggy-back-CPU are not fulfilled (compare appendix C.2)