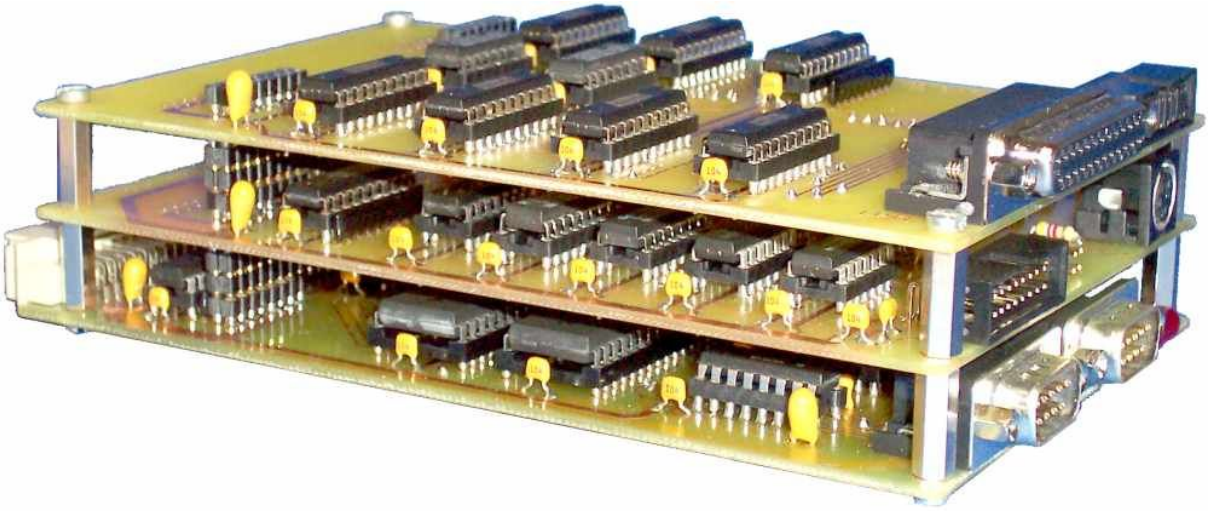
Multi - I/O Unit

For the MyCPU

- Selfbuild Guide -

© 2015 / Dennis Kuschel



uses `/IOEN1:2200h-23FFh, /HALT` and `IRQ1 - IRQ3`

Dennis Kuschel · Corintostraße 21 · 28279 Bremen · Germany · www.mycpu.eu · dennis_k@freenet.de 2015-07-14

Multi-I/O Unit Technical Details

➢ Multi-I/O Base Board

- Provides two standard serial RS232 ports (with 16C550 UART)
- Full hardware handshake supported
- Allows the use of line-powered RS232 devices
- Four LEDs for status information
- Does not require further hardware. A minimal MyCPU System requires only the MyCPU itself, the Memory Base Board and this Multi-I/O Base Board.

➢ Keyboard- and LCD- Interface Board

- Optional board, adds two more interfaces to the Multi-I/O-Unit
- Provides a standard PS2- keyboard connector. A minimum keyboard controller hardware off-loads the CPU. The keyboard controller is implemented ½ in hardware and ½ in software.
- Provides an interface for standard LC-Displays. The display chips HD44780 and KS0073 are supported. Supported display sizes: 2x20 / 2x40 / 4x20 / 4x40

➢ Parallel Ports Interface Board

- Optional board, adds two more interfaces to the Multi-I/O-Unit
- Provides a standard line printer port (LPT). Allows the use of printers with the old style centronics interface. Only the unidirectional SPP mode is supported.
- Provides an 8 bit multi-I/O interface (8 digital inputs plus 8 digital outputs)

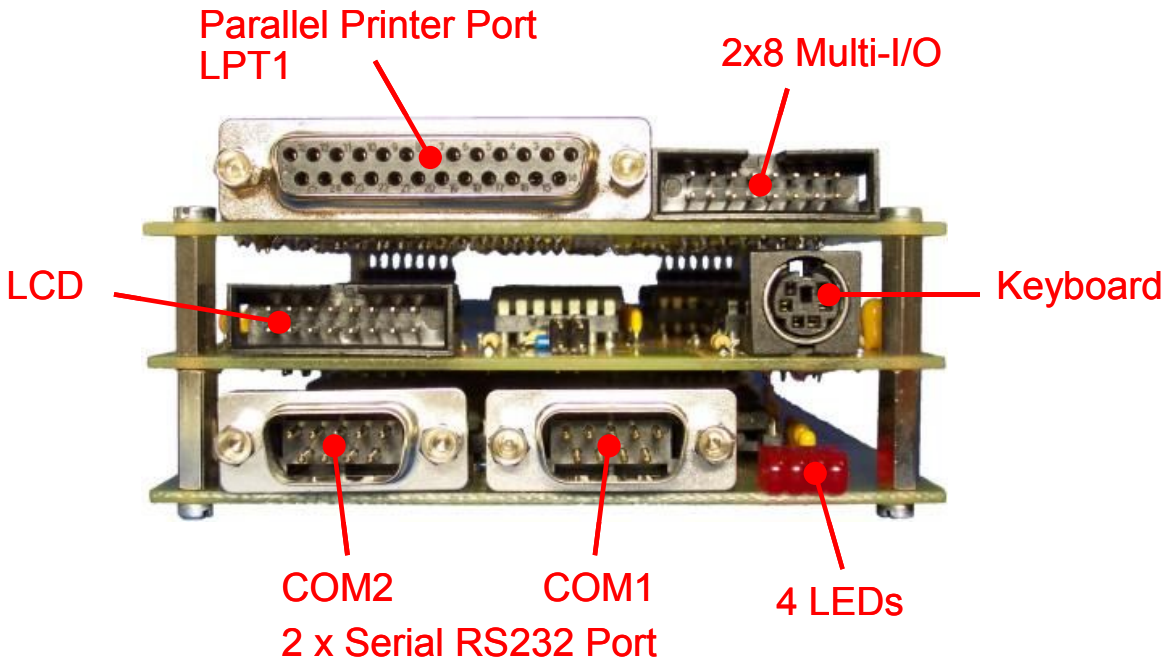


Fig. 1: Multi-I/O Unit: Connectors

Content

Figures

Tables

Author:

Dennis Kuschel
Corintostraße 21
28279 Bremen
Germany

web : http://www.mycpu.eu
email: dennis_k@freenet.de

1 Boards

1.1 Multi-I/O Base Board

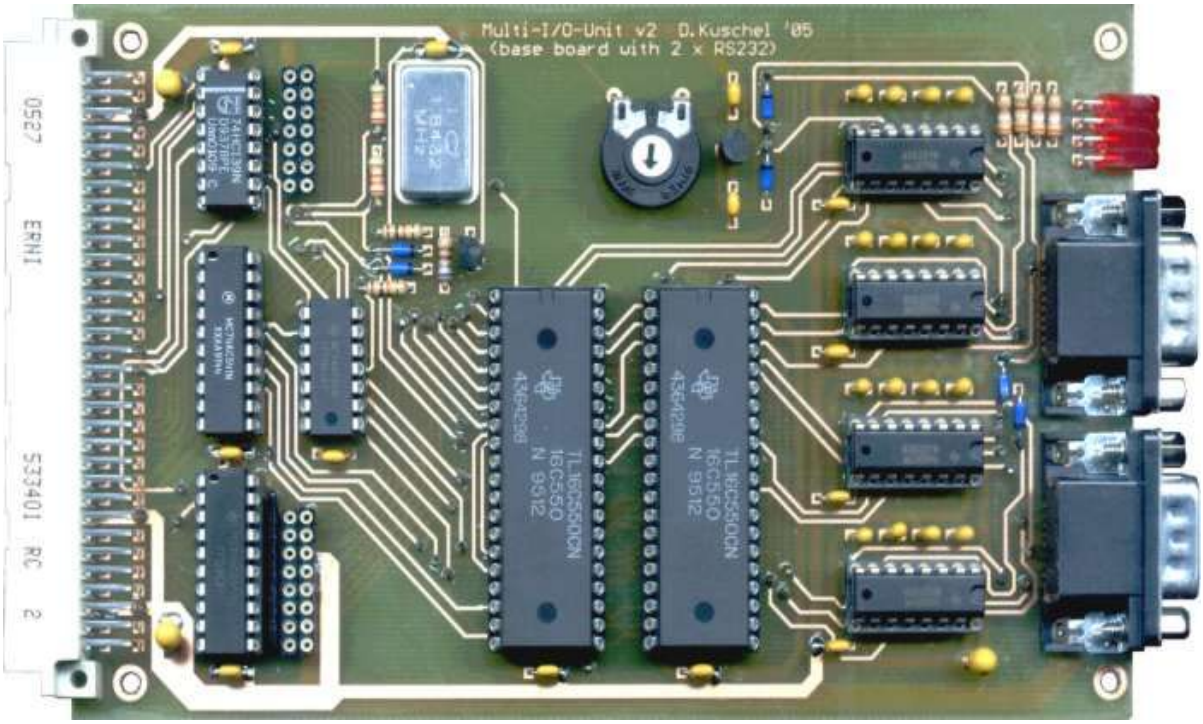


Fig. 2: Multi-I/O Base Board

1.1.1 Description

The Multi-I/O Base Board provides two standard serial RS232 ports. This allows to setup a minimal MyCPU computer system that is controlled through a terminal that is connected to the first serial port (COM1). COM2 can be used for network access (loading and executing software from a fileserver).
The board provides standard RS232 hardware handshake and standard conform signal levels by using the MAX232 line drivers. Because all RS232 signals are supported (including DTR / DSR and DCD), it is possible to use RS232 devices that are powered through the RS232 line.

1.1.2 Selection of Components

There are some 74ACxxx types listed in the partlist. The AC-type parts are required if you want to run the computer system on frequencies above 4 MHz. Because the Advanced CMOS (AC) parts are hard to get and very expensive, you could also use the cheaper HC types. Dependent on the quality of the components, you can also reach clock rates up to 6 MHz by using only the cheaper 74HCxxx types.

1.1.3 Placement of Components

After you have soldered all Via's, you can continue with the integrated circuits. I suggest you not to use sockets for the IC's, except for the UARTs and the 74ACxxx parts. If you wish to use sockets, you must use precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I strongly recommend you to follow the placement order I have noted in the placeplan below (see blue numbers, and start with the IC that has the blue number 1). When all IC's are placed and soldered, you can continue to place the capacitors. In the last round the resistors, diodes and board connectors are placed and soldered.

ATTENTION!
Please be careful, and don't forget to solder a pad on the top side of the board. I have marked all critical pads with red paint in the placeplan below. Please check if you have really soldered these pads!

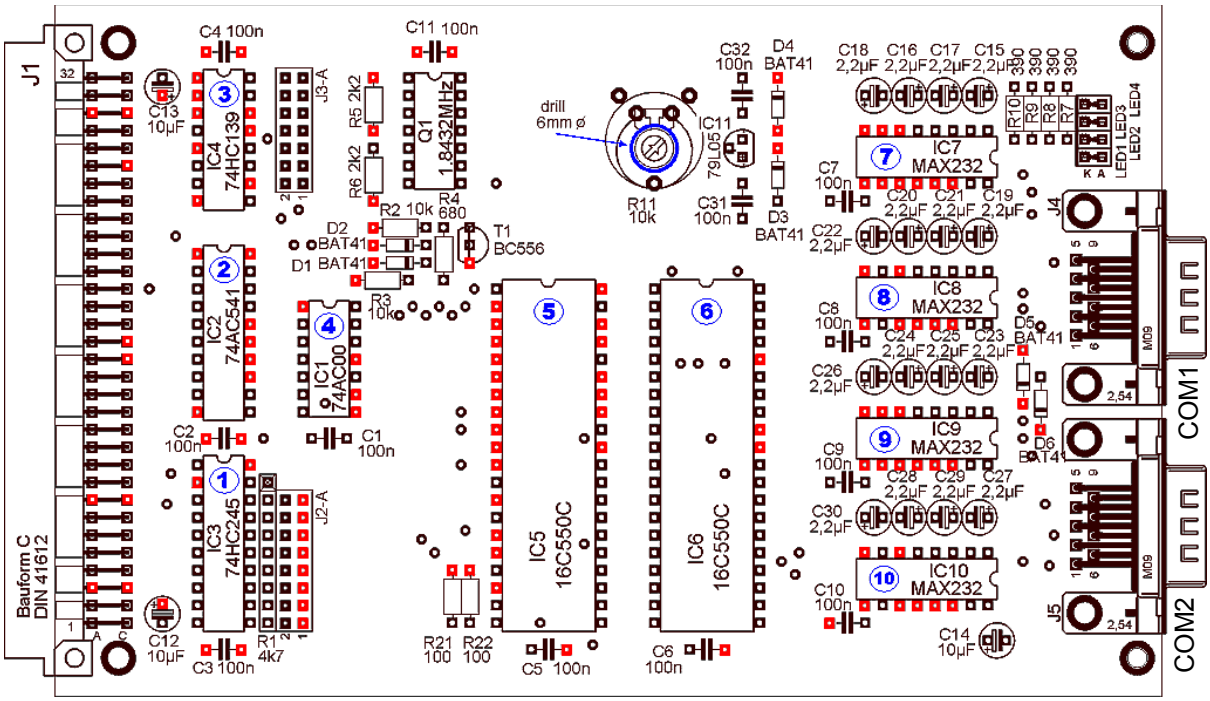
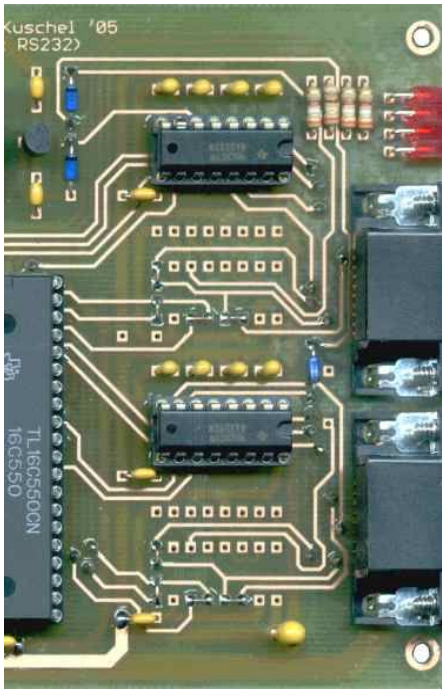
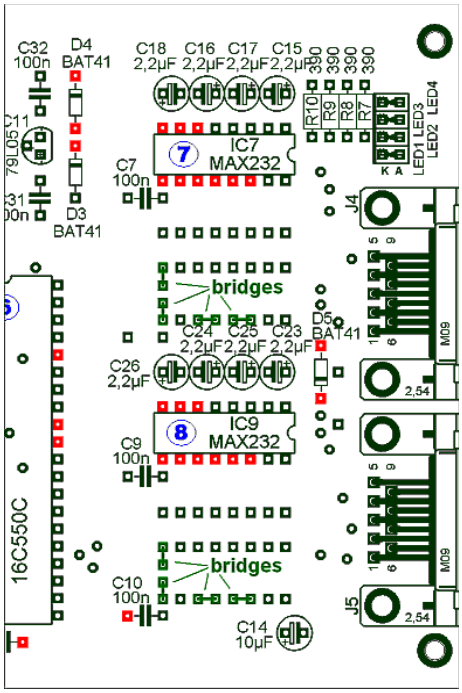


Fig. 3: Memory Base Board - Placeplan

1.1.4 Option: 2 instead of 4 MAX232

It is possible to use only two instead of the four MAX232 line driver devices. This saves money, but both RS232 interfaces will not support the DTR, DSR and DCD lines. Furthermore, this may result in problems if you wish to use line-powered RS232 devices that get their power from the DTR line.





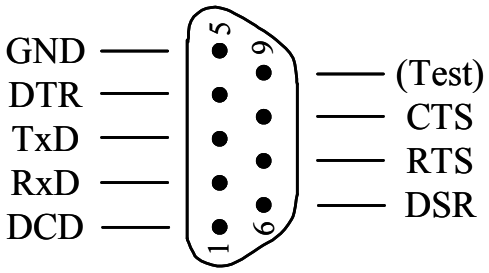
When IC8 and IC10 are not placed, also C19 - C22, C27 - C30, D6 and C8 need not to be populated. Note that some wire-bridges must be soldered.

Fig. 4: Bridges instead of IC8 and IC10.

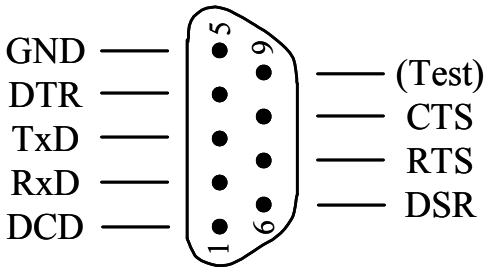
1.1.5 Partlist

74AC00	IC1	
74HC139	IC4	
74HC245	IC3	
74AC541	IC2	
16C550 -or- 85C550	IC5, IC6	***xxC550 required!***
MAX232	IC7, IC8, IC9, IC10	
79L05	IC11	
BC556	T1	
BAT41	D1, D2, D3, D4, D5, D6	
LED, flat package, 2.5x5mm	LED1, LED2, LED3, LED4	
100 Ohm	R21, R22	
390 Ohm	R7, R8, R9, R10	
680 Ohm	R4	
2.2 kOhm	R5, R6	
10 kOhm	R2, R3	
SIL 8 x 4.7 kOhm	R1	
Potentiometer, 10 kOhm	R11	
100nF ceramic capacitor	C1 - C11, C31, C32	
2.2µF / 16V, Tantal	C15 - C30	
10µF / 16V, Tantal	C12, C13, C14	
Oscillator 1.8432 MHz	Q1	
SUB D-9 Connector (male)	J4, J5	
DIN 41612 Connector	J1	
14 pin header	J3-A	
16 pin header	J2-A	

1.1.6 RS232 Connectors



J4 / COM1



J5 / COM2

Fig. 5: RS232 Connectors (male - top view)

1.1.7 Test

For the first tests it is required that no LCD is connected to the Multi-I/O Unit! The test-environment consists of the MyCPU, the Memory Unit and the Multi-I/O Unit.

1.1.7.1 LEDs

There are 4 LEDs on the base board. After powering on the MyCPU, the LEDs should show a moving light after a couple of seconds. If the LEDs do not light, you have probably old software installed. You need at least Kernel Version 2.3. This kernel is part of the EPROM release 1.03 or newer. You can check your ROM version by entering the "`ver`" -command at the MyCPU command prompt.

The state of the LEDs can be changed by the user. For example, they can be used to show the program status. The Kernel exports the function `$0306` (`KERN_IOCHANGELED`) for this purpose. You can change the state of the LEDs with the following shell commands:

`call ($0306) $00`	switch all LEDs off
`call ($0306) $01`	switch only the first LED on, other LEDs are off
`call ($0306) $02`	switch only the second LED on, other LEDs are off
`call ($0306) $04`	switch only the third LED on, other LEDs are off
`call ($0306) $08`	switch only the fourth LED on, other LEDs are off
`call ($0306) $03`	switch first and second LED on, other LEDs off
`call ($0306) $0B`	switch LEDs one / two / three on, fourth LED off
`call ($0306) $0x`	(other LED combinations possible: x = bitmask)
`call ($0306) $0F`	switch all LEDs on
`call ($0306) $10`	switch off first LED, other LEDs remain unchanged
`call ($0306) $11`	switch off second LED, other LEDs remain unchanged
`call ($0306) $12`	switch off third LED, other LEDs remain unchanged
`call ($0306) $13`	switch off fourth LED, other LEDs remain unchanged
`call ($0306) $20`	switch on first LED, other LEDs remain unchanged
`call ($0306) $21`	switch on second LED, other LEDs remain unchanged
`call ($0306) $22`	switch on third LED, other LEDs remain unchanged
`call ($0306) $23`	switch on fourth LED, other LEDs remain unchanged
`call ($0306) $FF`	switch on the automatic moving light (like after reset)
`call ($0306) $03 >nul`	(example) The "`>nul`" suppresses the status line that is printed by the call-command.

To change a LED from within a basic program, you can use this command:

```
shell "call ($0306) $03 >nul"
```

The following line outputs the decimal value 3 on the LEDs:

```
V = 3
shell "call ($0306) "+str$(v)+" >nul"
```

Note: This method of setting the LEDs from within a basic program is not very fast, because every time a new shell instance is opened.

1.1.7.2 COM1

The first serial port (COM1) is used as terminal port. With a terminal attached to this port (e.g. a Windows PC running "HyperTerminal") you will get access to the MyCPU operating system. The port settings are as follows:

```
115200 Baud, 8 Databits, 1 Stopbit, no parity, hardware handshake
```

Please connect COM1 of the MyCPU to COM1 of your computer and open and configure (for example) the "HyperTerminal" program. After switching on the MyCPU, you should see some text on the terminal. After a while, you will see the command prompt `8:/>`. Now you are ready to enter some commands. Here are some useful shell commands you should try:

```
8:/> basic     - starts the basic interpreter
8:/> mem -a    - displays information about the system memory
8:/> edit      - text editor, use [F1] for menu and [TAB] to navigate in file dialogs
8:/> ver       - displays software version numbers
8:/> set       - displays or sets environment variables
8:/> dir       - displays the content of the current directory
8:/> 15:       - change current drive and change to drive 15 (ROM drive)
```

Note: If you see some garbage in the terminal window, or the terminal behaves different after each reset of the MyCPU, you may have a problem with the UART chips. In my tests I found that the 16C550 - chips from Texas Instruments behave a bit unpredictable. If you have such chips, you may try to replace IC2 (74AC541) by its HC counterpart. If this does not help, you can also try to replace IC1. But the safest way is to replace the 16C550. I have some chips from "LGS" (GM16C550) that work properly.

IMPORTANT! The MyCPU requires hardware handshaking. Else the computer will not boot. When no cable is connected to COM1, at least the pins 7 and 8 of J4 must be bridged!

1.1.7.3 COM2

Testing COM2 is a bit more difficult. When COM1 works, you must connect COM2 also to your personal computer. Open a second HyperTerminal and configure it like the first one (see settings above). Then you must enter some commands in the first Hyper Terminal window:

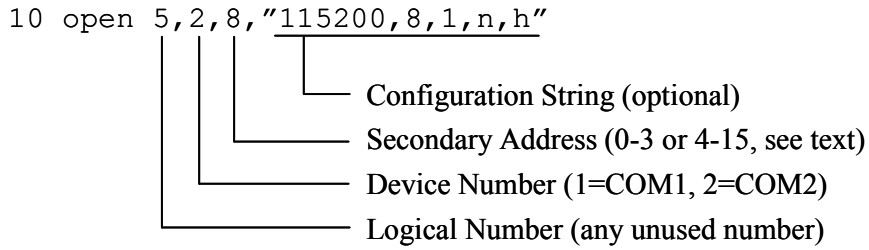
```
8:/> kill remotefs
8:/> basic

new
10 open2,2,8
20 get#2,a$:ifa$=""then20
30 print#2,a$;
40 print"input: ";a$
50 goto20
run
```

The first command ("kill") removes the network driver "remotefs" from memory, so COM2 gets usable. The second command starts the basic interpreter. The small basic program opens COM2 and waits for input on that port. When you press some keys in the second Hyper Terminal, you should immediately get the echo in the same terminal, and additionally the text "input: x" should be printed to the console terminal window.

1.1.7.4 Using Serial Interfaces

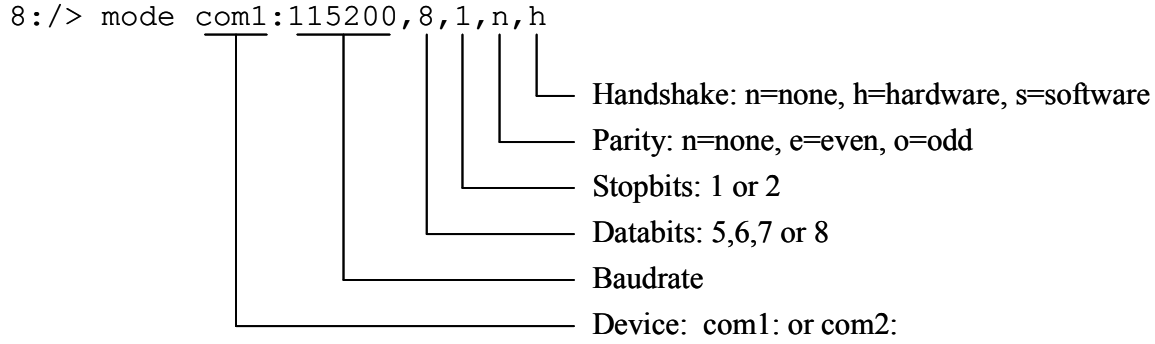
The operating system supports two serial interfaces ("UARTs"). The both interfaces are named com1: and com2: (like in the PC environment). Additionally, you can use device-numbers to access the ports. The first port has the device number 1 and the second one the number 2. For example, the following command will open the second serial port from within a basic program:



The secondary address selects if the com-port is opened in binary mode (secondary addresses 0 - 3) or in text-conversion mode (secondary addresses 4 - 15). In text-conversion mode, incoming ASCII-text is converted into PETSI-code and outgoing PETSI-code is converted to ASCII-text. Example:

`open 5,2,8` - converts between PETSI / ASCII code
`open 5,2,0` - does no character conversion

It is also possible to change the settings of the COM-ports from the MyCPU command shell. The appropriated command is "mode" :



A hint on device numbers: Devices are numbered from `0:` to `15:`. The serial ports have the numbers `1:` and `2:`, or alternative on the shell you can write `com1:` and `com2:`. Here is an example:

"`copy welcome.txt com1: /a`" is the same like "`copy welcome.txt 1: /a`"

NOTE:
In the standard configuration (directly after powering on the MyCPU) the remote file system driver is installed on com2:. So if you wish to use com2: for your own purpose, you must first uninstall the driver by typing "`kill remotefs`" at the command prompt.

1.1.7.5 X/Y-Modem File Transfer

You can use the X- or Y-Modem Protocol to transfer files to or from the MyCPU computer system. My suggestion is to use the more powerful Y-Modem protocol if your terminal supports it. Z-Modem is not supported.

Use this command to transfer a file from the terminal sever to the MyCPU:

```
8:/> xget com1:
```

Now use your terminal program to upload the file. With HyperTerminal, click “Transfer” and then “Send File”, select “Ymodem” and choose the file to upload. The file will now be stored in the actual directory on the MyCPU, that is 8:/ in the example above.

To transfer a file from the MyCPU to your personal computer, use this command:

```
8:/> xput com1: file
```

Note that “`file`” is the name of the file you wish to transfer. After you have entered this command on the MyCPU, use your terminal to download the file. With HyperTerminal, click “Transfer” and then “Receive File”, select “Ymodem” and choose the destination directory.

As mentioned above, the `xget` / `xput` - commands can also do the X-Modem protocol. To use X-Modem, the commands change as follows:

```
8:/> xget com1: filename filesize

8:/> xput x com1: file
```

1.1.7.6 Remote File System Service

By default, the remote file system service is installed on com2:. This enables the MyCPU-system to access files on a fileserver (a PC workstation configured as server for MyCPU). This is useful for a minimalistic MyCPU System consisting of just the MyCPU, memory board and Multi-I/O Unit. The hardware setup is really simple: Connect `com1:` of the MyCPU with `com1:` of your PC, and connect `com2:` of the MyCPU with `com2:` of your PC. At the PC side, run a HyperTerminal on `com1:` and the MyCPU fileserver service on `com2:`.

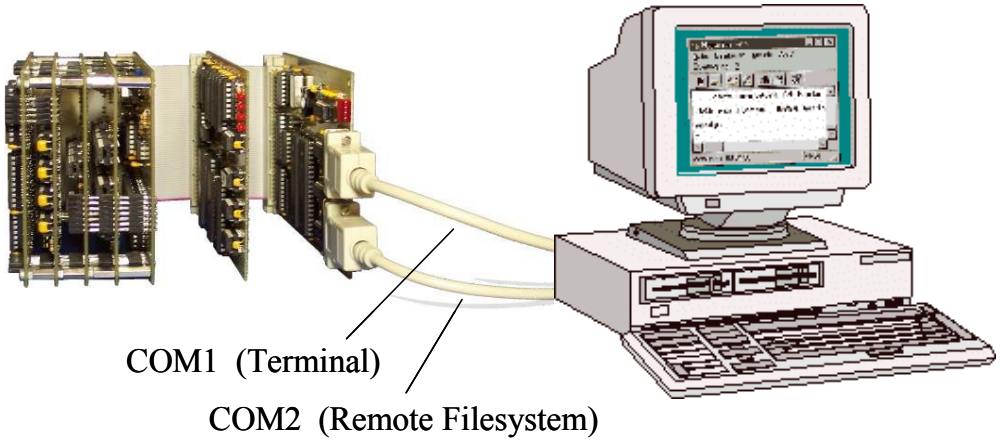


Fig. 6: Connection MyCPU (client) - PC (server)

You must run the remote file system server software on the PC. Please download the full MyCPU software package from the MyCPU homepage (currently http://mycpu.microcontroller.net). When you unpack the zip-file, you will find a subdirectory named "server". This is the server program for the remote filesystem service. To run the server on PC `com2:` please start the program by typing "`rfss /p2`" in a DOS-box. When you power on the MyCPU, you should see the following text in your terminal window:

```
FOUND: RS232 Interface (UART)

WARNING: No interrupt controller found.

Romdrive installed (drive 15)
starting 15:/init

Remote Filesystem Driver V2.2
installed on drive 14:


  ---===###  BOOT-MENU  ###===---

  [1]  boot a minimal system
  [2]  with assembler environment
  [3]  with network support
  [4]  work on remote disk
```

Fig. 7: Booting the MyCPU from the PC server

1.1.8 Important Note on the RS232 Cable

Please make sure that you are using a correct wired NULL-Modem cable. Unfortunately, some cables that can be bought in local electronic stores nowadays, do have a wrong wiring and are NOT REAL NULL-MODEM CABLES. Please check the wiring of your cable set, and make sure the wiring of the RTS/CTS-lines is correct. The diagram below shows the correct wiring:

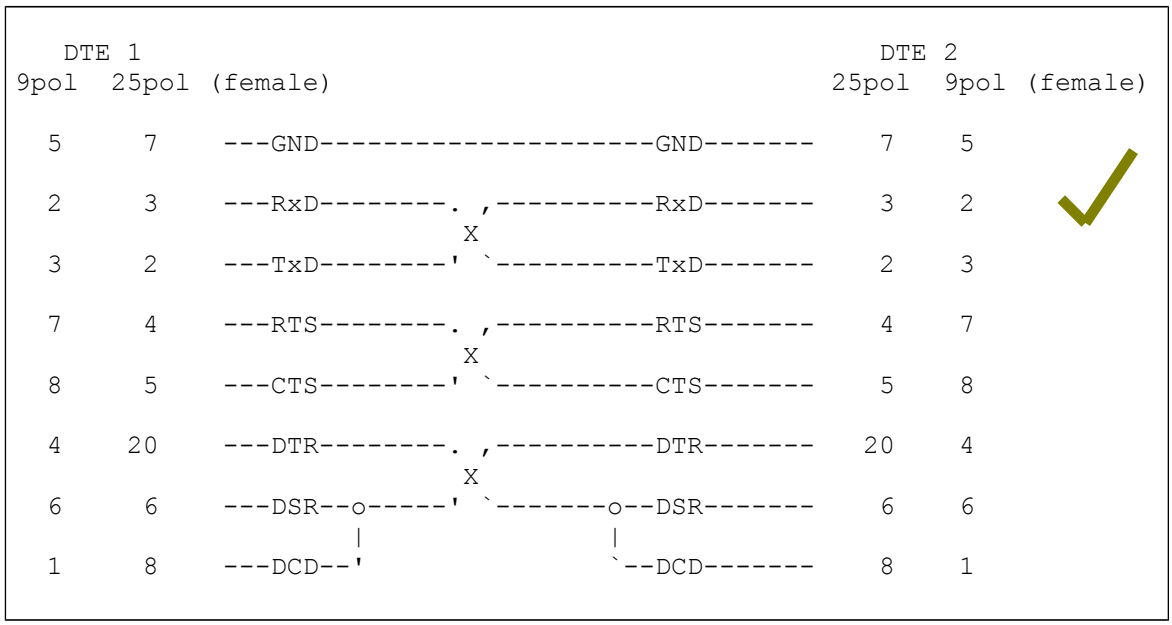


Fig. 8: Correct Wiring of a RS232 NULL-Modem Cable

Example for a WRONG wiring (e.g. of a cable bought at “Conrad Elektronik” in Germany):

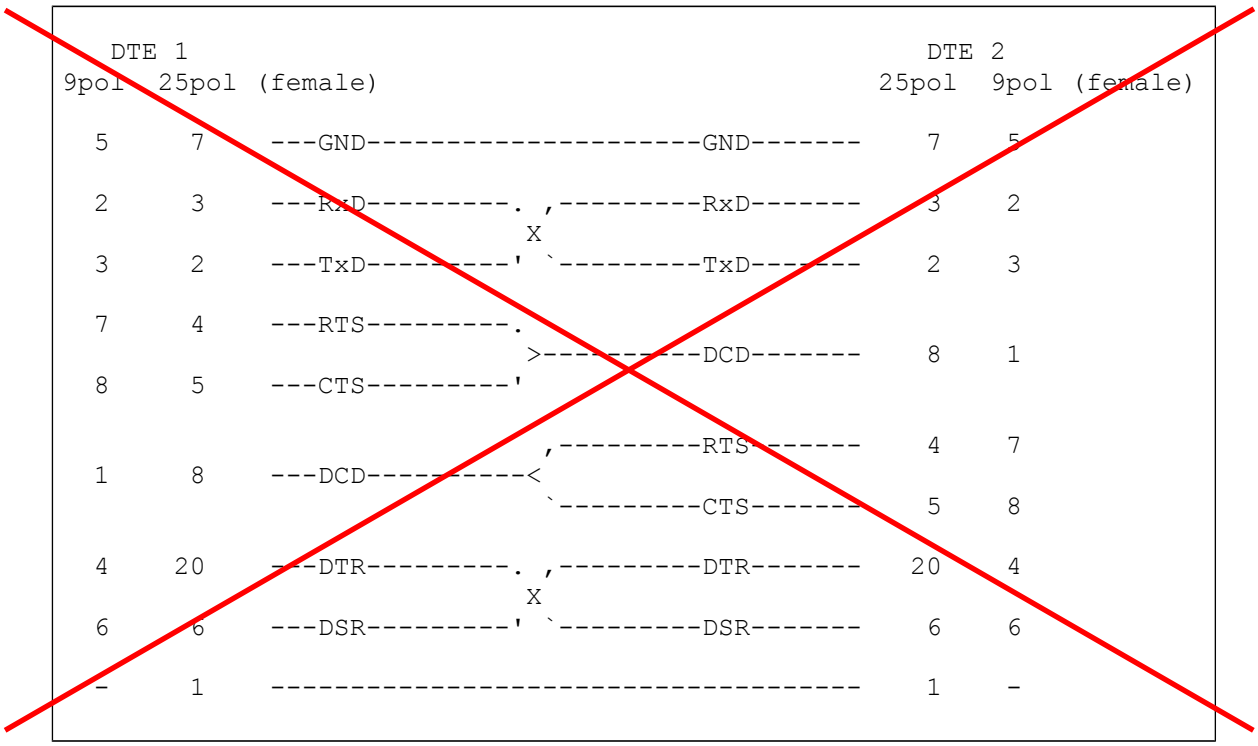


Fig. 9: Wrong Wiring of a RS232 NULL-Modem Cable

1.2 Keyboard- and LCD- Interface Board

Please Note:

The Keyboard- and LCD- Interface Board is ***not required*** *for a minimal working MyCPU-System. It is* ***not a must****, but* ***just an useful option****.*
To be able to use the Keyboard Interface, you ***must also have*** *the* ***Interrupt Controller Unit****.*

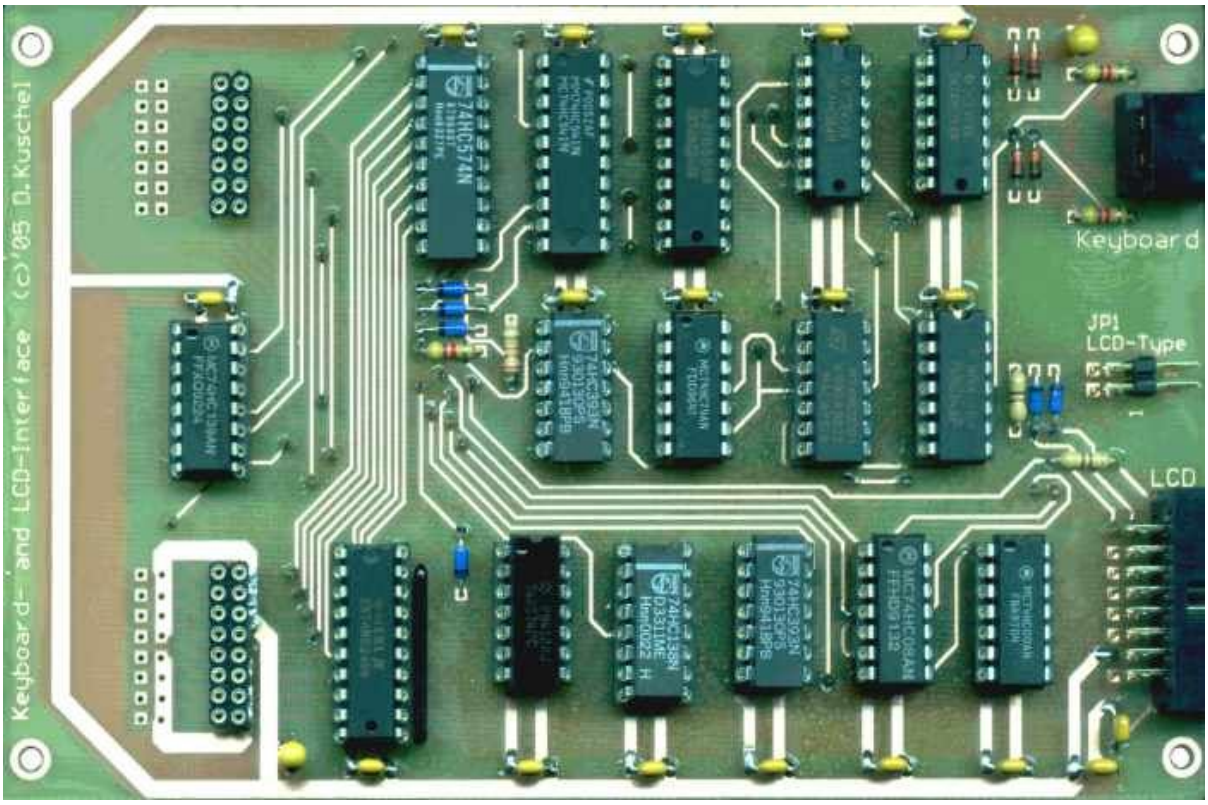


Fig. 10: Multi-I/O Keyboard- and LCD- Interface Board

1.2.1 Description

The Keyboard- and LCD-Interface Board allows you to connect standard AT-type keyboards with PS/2 connector to the MyCPU system. Furthermore, several types of LCD displays can be interfaced with this board. Currently LCD displays with the Hitachi chipset HD44780 (and compatible) are supported by the operating system.

1.2.2 Selection of Components

The 74AC74 integrated circuit (IC23) is required for proper operation, when also 74AC-types are used on the base board. Otherwise, also 74HC74 will do, but this results in reducing the maximum operating frequency.

1.2.3 Placement of Components

After you have soldered all Via's, you can continue with the integrated circuits. I suggest you not to use sockets for the IC's. If you wish to use sockets for all IC's, you must use precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I strongly recommend you to follow the placement order I have noted in the placeplan below (see blue and green numbers, and start with the IC that has the blue number 1). When all IC's are placed and soldered, you can continue to place the capacitors and resistors. As last you can place the board connectors.

ATTENTION!
Please be careful, and don't forget to solder a pad on the top side of the board. I have marked all critical pads with red paint in the placeplan below. Please check if you have really soldered these pads!

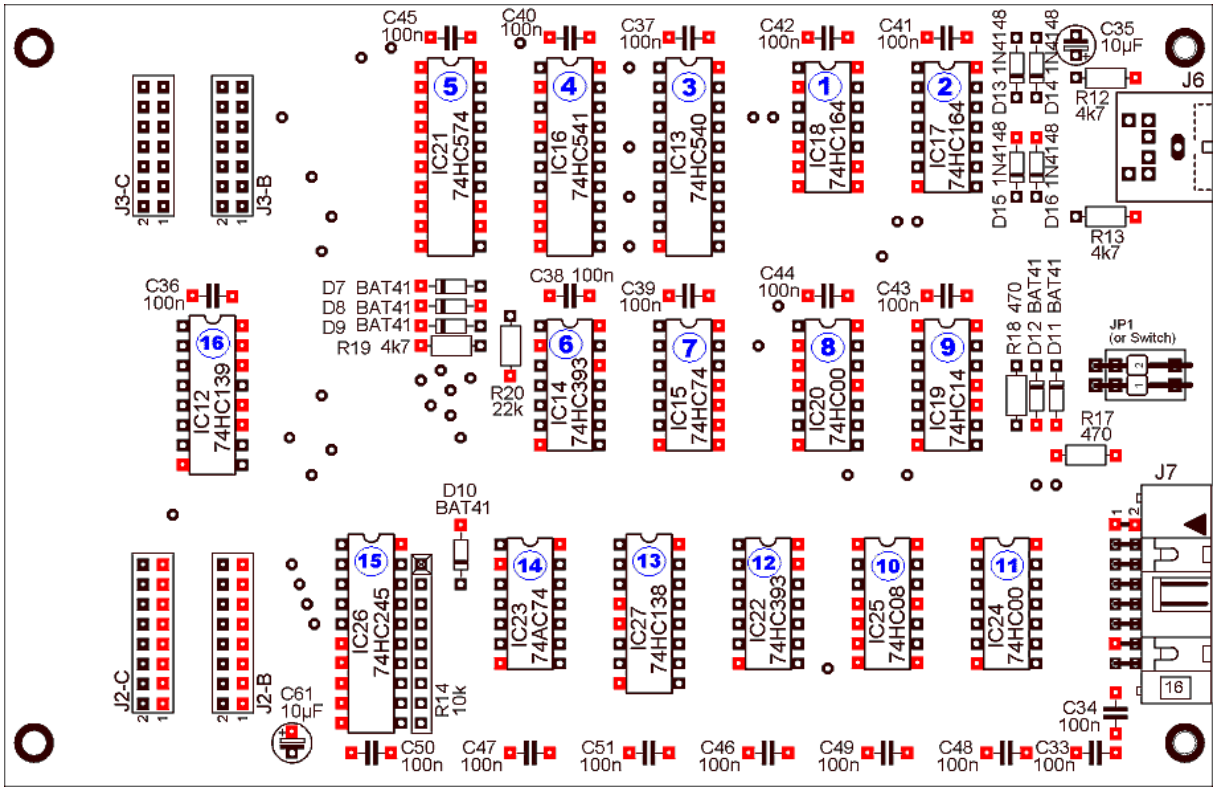


Fig. 11: Multi-I/O Keyboard- and LCD- Interface Board - Placeplan

1.2.4 Partlist

74HC00	IC20, IC24
74HC08	IC25
74HC14	IC19
74AC74	IC23
74HC74	IC15
74HC138	IC27
74HC139	IC12
74HC164	IC17, IC18
74HC393	IC14, IC22
74HC245	IC26
74HC540	IC13
74HC541	IC16
74HC574	IC21
BAT41	D7, D8, D9, D10, D11, D12
1N4148	D13, D14, D15, D16
470 Ohm	R17, R18
4.7 kOhm	R12, R13, R19
22 kOhm	R20
SIL 8 x 10 kOhm	R14
100nF ceramic capacitor	C33, C34, C36 - C51
10µF / 16V, Tantal	C35, C61
2 pin Jumper or DIP-Switch	JP1
PS/2 Keyboard Connector	J6

16 pin header for cable	J7
14 pin header	J3-B, J3-C
16 pin header	J2-B, J2-C

1.2.5 Keyboard- and LCD- Connector

The keyboard-connector supports standard AT-type keyboards. The LCD connector can be used for all kinds of HD44780 driven displays. There are two enable lines (E1 and E2) to address two Hitachi chips. This is required for e.g. big LCDs with 4x40 characters that use two controller chips.

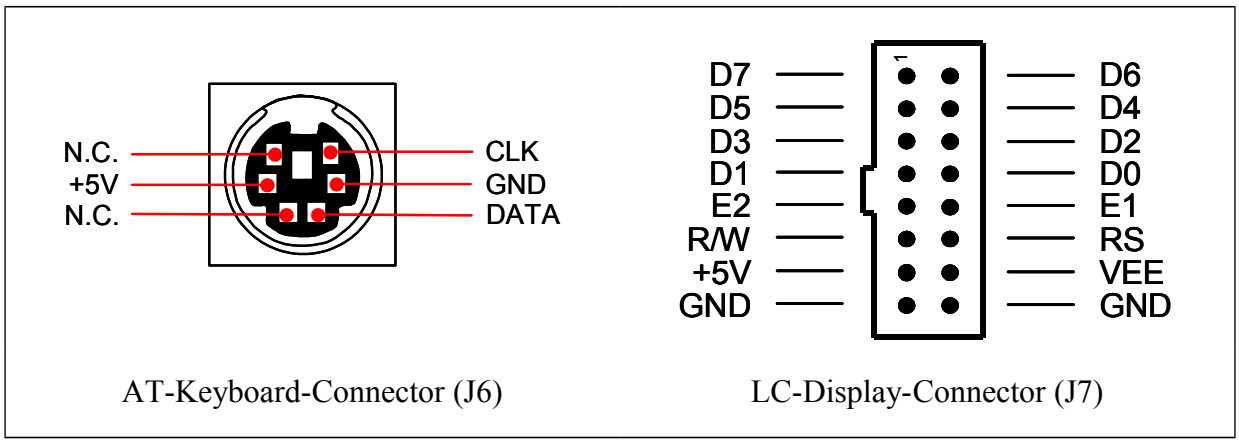


Fig. 12: Keyboard- and LCD- Connector

1.2.5.1 Keyboard Signals

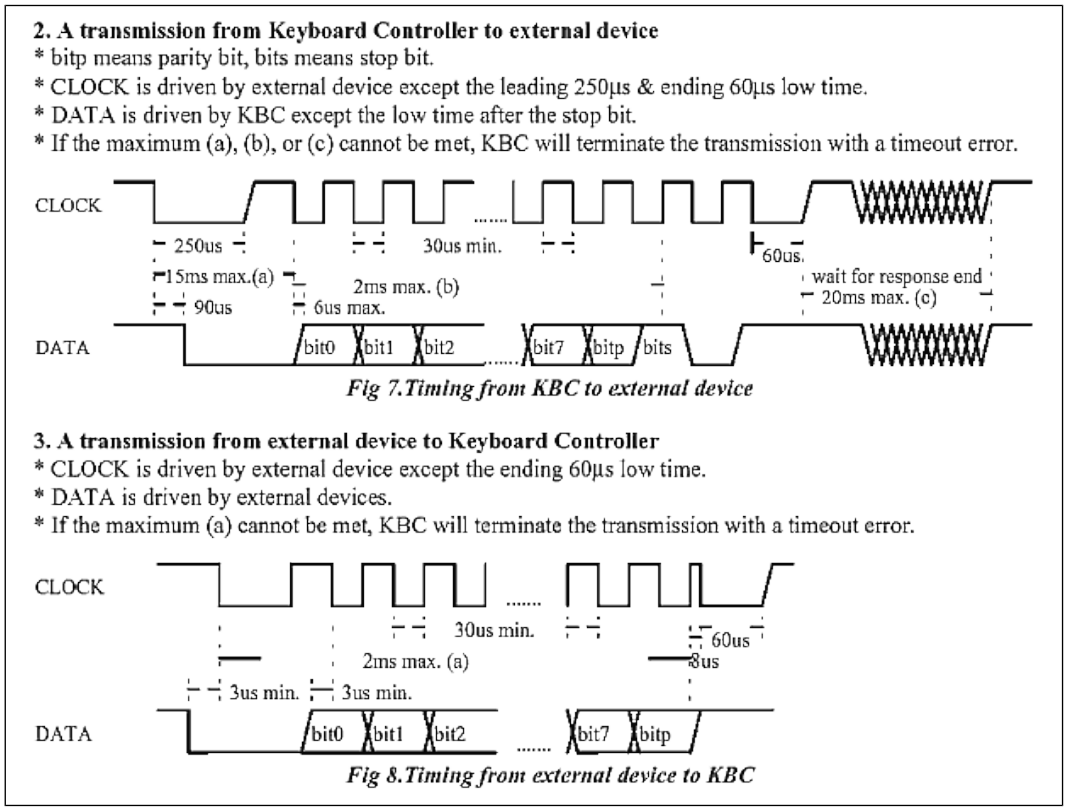


Fig. 13: Keyboard Data Timing Diagram

1.2.5.2 LCD Signals

The following table shows how a LC-Display is connected to J7 and how the jumper block JP1 must be configured. The voltage on the VEE-pin of J7 is adjustable between -5V and +5V and is used to drive the LCD plane. The adjustment is done with a potentiometer on the base-board (R11). To reduce the voltage range you can put a resistor of 1k between VEE and GND.

Note: The length of the cable between J7 and the LCD should not exceed 20 cm !

Size of LCD	RAM-Addresses	Number of Chips	used Enable -signals		Settings of JP1	
			E1	E2	1-3	2-4
2 x 20	00h, 40h	1	used	n.c.	short	Short
2 x 40	00h, 40h	1	used	n.c.	open	Short
4 x 20	00h, 40h, 14h, 54h	1	used	n.c.	short	Open
4 x 20	00h, 20h, 40h, 60h	1	used	n.c.	open	Open
4 x 40	1: 00h, 40h 2: 00h, 40h	2	chip 1	chip 2	open	Open

Tab. 1: LCD connection and configuration

1.2.5.3 LCD sliding window

The MyCPU operating system software is configured to work either with a remote terminal of 24x80 characters or with a LCD with a size of 4x40 characters. But anyway the LCD driver software allows you to connect smaller displays. The system sees a 4x40 display internally, but the driver writes only a part of this internal screen out to the display. So you will see only a part of the internal screen buffer on your LCD, and that is the section where the cursor is currently placed.

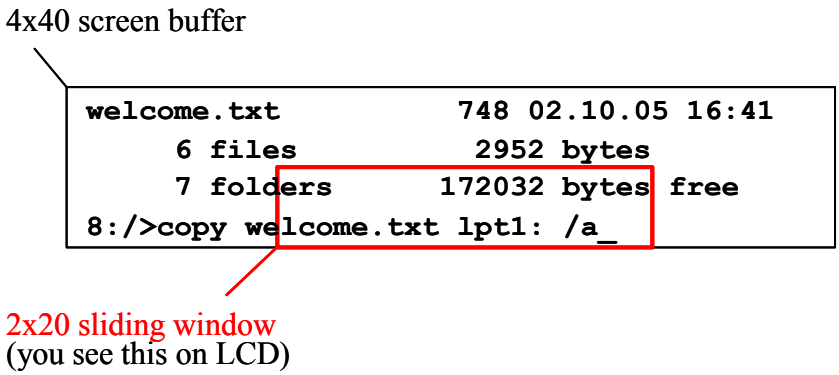


Fig. 14: LCD sliding window

1.2.6 Test

For the following tests it is required that you have the correct software version installed. You need at least Kernel Version 2.3. This kernel is part of the EPROM release 1.03 or newer. You can check your ROM version by entering the "`ver`" -command at the MyCPU command prompt.

1.2.6.1 Test of the Keyboard Controller

For this test it is required that no LCD module is connected to the Multi-I/O Unit! The test-environment consists of the MyCPU, the Memory Unit, the **Interrupt Controller** and the Multi-I/O Unit.

Please do NOT connect the keyboard to J6 for the initial test. When you switch on the MyCPU-System, you will get the message "`AT Keyboard error. Use TTY (COM1) instead`" in the terminal window. The MyCPU has detected the presence of the keyboard controller but has no access to the keyboard itself. When you now press ENTER in the terminal window, the system should continue to boot as usual. All is fine until now.

Now switch off the CPU, connect the keyboard to J6 and switch on the MyCPU-system again. The error message will no more appear, and you will see the line "`FOUND: AT Keyboard Controller`". The keyboard should work as expected: When you enter some text on the keyboard, the text will appear in the terminal window.

You can use the "`console`" command at the shell prompt to change the keyboard settings:

Command	Function
`console ttykeyb`	use terminal for keyboard input
`console keyb`	use AT-Keyboard for input
`console en`	set english keyboard layout
`console gr`	set german keyboard layout

1.2.6.2 Test of the LCD interface

Attach a LCD to J7 and set JP1 appropriated. When you power on the MyCPU-System, you should get some output on the LCD. The terminal-window remains clean (no text output). If you do not see anything on the LCD, please try first to change the LCD voltage by adjusting R11 on the base board (the resistor has the label "LCD contrast" at the bottom side of the base board).

When you believe that the lines you see on the display are in randomized order, please try to change the setting of JP1. In the worst case the operating system does not support your LCD. If so, you can change the LCD driver by yourself (see file sources/kernel/char2lcd.asm in the MyCPU software package).

The "`console`" command can be used to change the standard output:

Command	Function
`console lcd`	use the LCD for display
`console screen`	use the VGA-unit for display
`console ttydisp`	use the terminal for display
`console tty`	use the terminal for input and output

1.3 Parallel I/O - Board

Please Note:

*The Parallel I/O - Board is **not required** for a minimal working MyCPU-System. It is **not a must**, but **just an useful option**.*

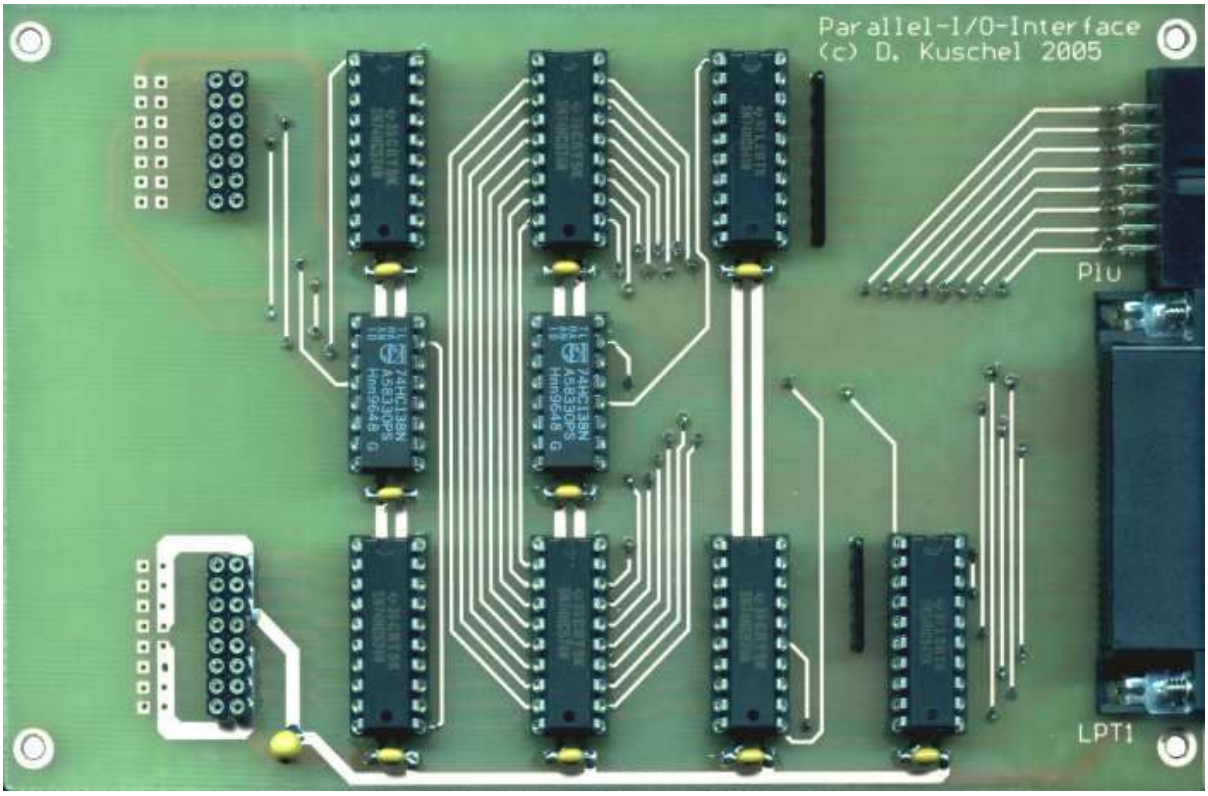


Fig. 15: Parallel I/O Board

1.3.1 Description

This board adds a parallel line printer interface and a 16 pin digital I/O port to the MyCPU system. The line printer can be accessed by using the devices "`lpt1:`" and "`prn:`" at the operating system level. The digital I/O-port has 8 inputs and 8 outputs that can be accessed by direct memory writes to the appropriated register address.

1.3.2 Selection of Components

No special parts are required.

1.3.3 Placement of Components

After you have soldered all Via's, you can continue with the integrated circuits. I strongly suggest you to use sockets for the IC's, because it can easily happen that an IC is destroyed through a voltage error on one of the connectors. And please follow the placement order I have noted in the placeplan below (see blue and green numbers, and start with the IC that has the blue number 1). When all IC's are placed and soldered, you can continue to place the capacitors, resistors and headers.

ATTENTION!
Please be careful, and don't forget to solder a pad on the top side of the board. I have marked all critical pads with red paint in the placeplan below. Please check if you have really soldered these pads!

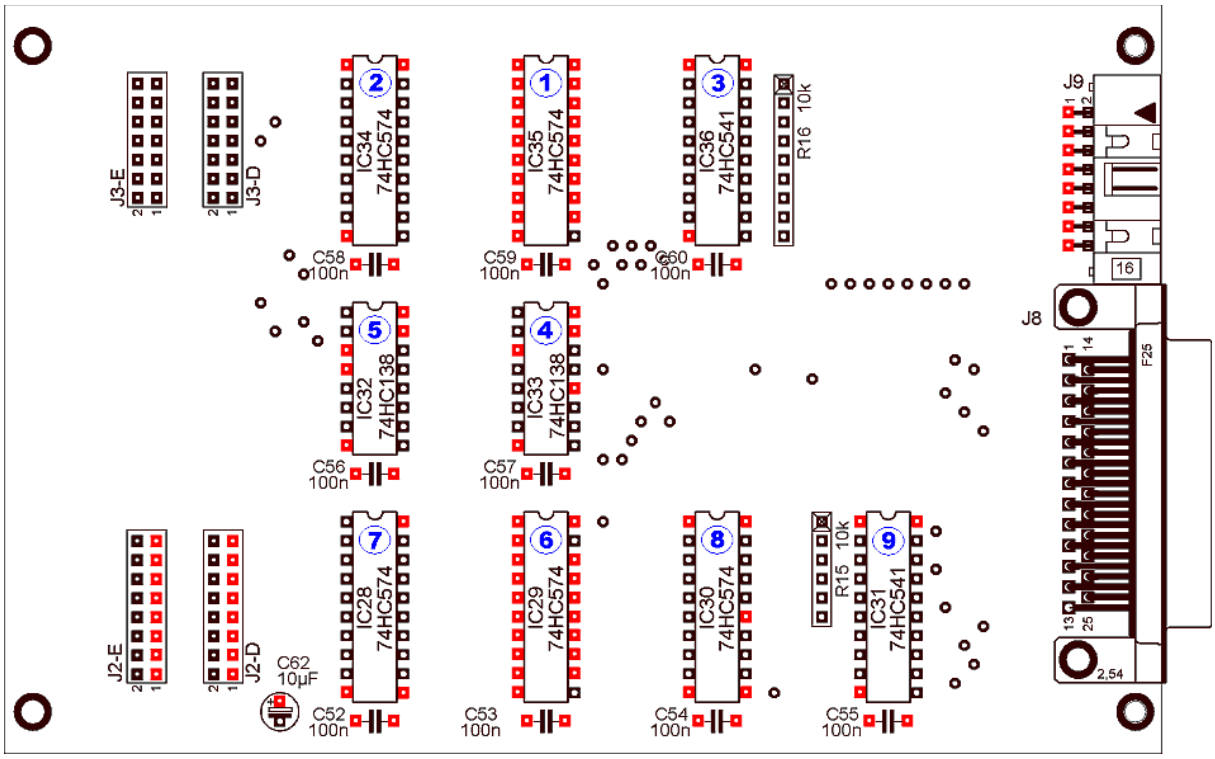


Fig. 16: Parallel I/O Board - Placeplan

1.3.4 Partlist

74HC138	IC32, IC33
74HC541	IC31, IC36
74HC574	IC28, IC29, IC30, IC34, IC35
SIL 5 x 10 kOhm	R15
SIL 8 x 10 kOhm	R16
100nF ceramic capacitor	C52 - C60
10µF / 16V, Tantal	C62
SUB D-25 Connector (female)	J8
16 pin header for cable	J9
14 pin header	J3-D, J3-E
16 pin header	J2-D, J2-E

1.3.5 Test

For the following tests it is required that you have the correct software version installed. You need at least Kernel Version 2.3. This kernel is part of the EPROM release 1.03 or newer. You can check your ROM version by entering the "`ver`" -command at the MyCPU command prompt.

1.3.5.1 Test of the Printer Port

Test-environment: MyCPU, memory board, Multi-IO Unit (COM1 attached, COM2 unused).

To test the printer port, please attach a parallel line printer to J8. Note that the printer must be able to print raw ASCII text from a DOS environment. Cheap printers sometimes do not work without having a printer driver installed. The MyCPU parallel printer port does only support the SPP mode.

Switch on the printer. Then switch on the MyCPU. Type the following line at the shell prompt:

```
copy 15:/init prn:
```

Alternatively, you can type also:

```
copy 15:/init lpt1: /a
```

This command will print out the content of the init-script from drive 15. The parameter '`/a`' is required because it tells the copy command to convert the content of the script file into ASCII character format (the MyCPU usually stores texts in PETSI code). The `prn:` device does the code conversion automatically. You can also try to print some text from basic. The device number of the printer is `4` (`prn:` device, whereas the `lpt1:` device has the number `5`). Here is an example basic program you should try:

```
8:/> basic

new
10 open 1,4,0
20 for i=1 to 30
30 print#1,"Hello World! ";
40 next
50 print#1
60 print#1,"-END-OF-TEST-"
70 print#1,chr$(12);
80 close 1
run
```

Fig. 17: Basic Program for Printer Test

Summary of devices:

`prn: / 4:` - logical printer device, does code conversion automatically
`lpt1: / 5:` - physical printer device, transfers bytes unchanged

1.3.5.2 Test of the Parallel I/O-Port

The multi-purpose parallel I/O-Port provides 8 digital inputs and 8 digital outputs. The digital inputs have 10k pull-up resistors installed, so it is easily possible to connect switches to the inputs (you need to connect the switch between the input-pin and ground). The digital outputs have TTL level and can drive a LED directly (a resistor of 390 Ohm is required). The pin assignment of J9 is as follows:

Pin Number	Function	Pin Number	Function
1	Output 0	2	Input 0
3	Output 1	4	Input 1
5	Output 2	6	Input 2
7	Output 3	8	Input 3
9	Output 4	10	Input 4
11	Output 5	12	Input 5
13	Output 6	14	Input 6
15	Output7	16	Input 7

Tab. 2: Pin-Assignment of J9

The simplest way to test the port is to connect inputs and outputs with jumpers directly in the connector. Please see the figure below:

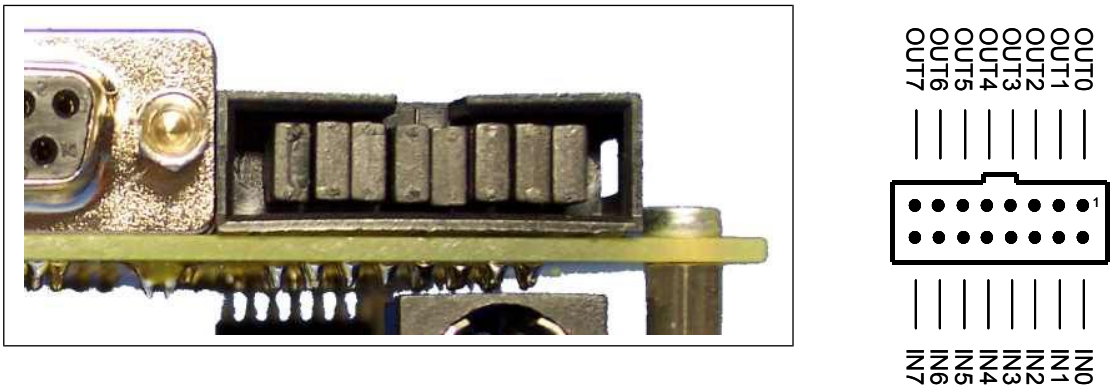


Fig. 18: Test-Configuration of J9

At the command shell prompt you can directly write data to the outputs and read it back. The “eb” - command writes a byte to a register, whereas the “db” - command reads a register back. Here is an example (of course, the values written and read back must be equal in this test):

```
8:/>eb $2382 $57

8:/>db $2382
2382: 57

8:/>db $2383
2383: 57
```

- write byte $57 to output register
- read back the output register
- read back the input register

Fig. 19: Testing the Parallel I/O-Port

2 Board Stack

2.1 Stacking the Boards

The Multi-I/O Unit consists of 3 printed boards. If wire-wrap connectors with long pins are used for J2 and J3, the boards can simply put together (please see the picture below for details). But you are free to use other connectors. On the Keyboard-/LCD-Interface-Board and on the Parallel-Ports-Interface-Board is space for a second connector. For example, you can decide to populate the connectors as follows:

- J2-A / J3-A on the Base-Board connect to J2-B / J3-B on the LCD/Keyboard-Board.
- J2-C / J3-C on the LCD/Keyb.-Board connect to J2-E / J3-E on the Parallel-I/O-Board.

– Or different board order –

- J2-A / J3-A on the Base-Board connect to J2-D / J3-D on the Parallel-I/O-Board.
- J2-E / J3-E on the LCD/Keyboard-Board connect to J2-C / J3-C on the LCD/Keyb.-Board.

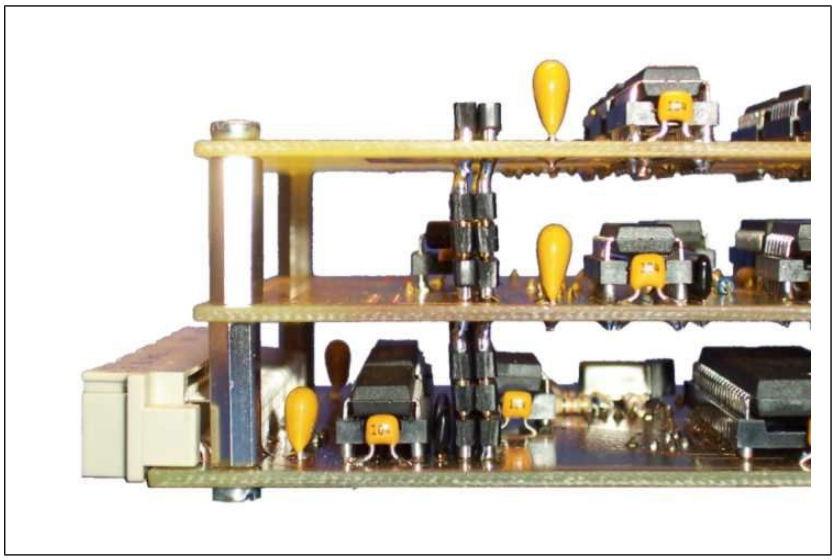


Fig. 20: Board Stack – Position of Connectors (example)

3 Overall Part List

3.1 Detailed list of required parts

1 x	74AC00	IC1
2 x	74HC00	IC20, IC24
1 x	74HC08	IC25
1 x	74HC14	IC19
1 x	74AC74	IC23
1 x	74HC74	IC15
3 x	74HC138	IC27, IC32, IC33
2 x	74HC139	IC4, IC12
2 x	74HC164	IC17, IC18
2 x	74HC245	IC3, IC26
2 x	74HC393	IC14, IC22
1 x	74HC540	IC13
4 x	74AC541	IC2, IC16, IC31, IC36
6 x	74HC574	IC21, IC28, IC29, IC30, IC34, IC35
2 x	16C550 -or- 85C550	IC5, IC6
4 x	MAX232	IC7, IC8, IC9, IC10
1 x	79L05	IC11
1 x	BC556	T1
4 x	1N4148	D13, D14, D15, D16
12 x	BAT41	D1 - D12
4 x	LED, flat package, 2.5x5mm	LED1, LED2, LED3, LED4
2 x	100 Ohm	R21, R22
4 x	390 Ohm	R7, R8, R9, R10
2 x	470 Ohm	R17, R18
1 x	680 Ohm	R4
2 x	2.2 kOhm	R5, R6
3 x	4.7 kOhm	R12, R13, R19
2 x	10 kOhm	R2, R3
1 x	22 kOhm	R20
1 x	SIL 8 x 4.7 kOhm	R1
1 x	SIL 5 x 10 kOhm	R15
2 x	SIL 8 x 10 kOhm	R14, R16
1 x	Potentiometer, 10 kOhm	R11
40 x	100nF ceramic capacitor	C1 - C11, C31 - C34, C36 - C60
16 x	2.2µF / 16V, Tantal	C15 - C30
5 x	10µF / 16V, Tantal	C12, C13, C14, C35, C61, C62
1 x	Oscillator 1.8432 MHz	Q1
1 x	2 pin Jumper or DIP-Switch	JP1
2 x	SUB D-9 Connector (male)	J4, J5
1 x	SUB D-25 Connector (female)	J8
1 x	DIN 41612 Connector	J1
1 x	PS/2 Keyboard Connector	J6
2 x	16 pin header for cable	J7, J9
5 x	14 pin header	J3-A, J3-B, J3-C, J3-D, J3-E
5 x	16 pin header	J2-A, J2-B, J2-C, J2-D, J2-E

4 Registers

4.1 Overview – All Registers of the Multi-I/O-Unit

Address	Register Name	Read/Write	Function
2200h	REG_KEYB_CTRL	R/W	Keyboard control register
2201h	REG_KEYB_DATA	R	Keyboard data
2201h	REG_KEYB_IRES	W	Reset keyboard interrupt condition
2280h	REG_LCD1_CTRL	R/W	LC-Display 1 – Control Register
2281h	REG_LCD1_DATA	R/W	LC-Display 1 – Data Register
2282h	REG_LCD2_CTRL	R/W	LC-Display 2 – Control Register
2283h	REG_LCD2_DATA	R/W	LC-Display 2 – Data Register
2300h	REG_UART1_DATA	R/W	UART1 – Data Register for send and receive
2301h	REG_UART1_IER	R/W	UART1 – Interrupt Enable Register
2302h	REG_UART1_IIR	R	UART1 – Interrupt Identify Register
2302h	REG_UART1_FCR	W	UART1 – FIFO Control Register
2303h	REG_UART1_LCR	R/W	UART1 – Line Control Register
2304h	REG_UART1_MCR	R/W	UART1 – Modem Control Register
2305h	REG_UART1_LSR	R	UART1 – Line Status Register
2306h	REG_UART1_MSR	R	UART1 – Modem Status Register
2307h	REG_UART1_SCR	R/W	UART1 – Scratch Register
2308h	REG_UART2_DATA	R/W	UART2 – Data Register for send and receive
2309h	REG_UART2_IER	R/W	UART2 – Interrupt Enable Register
230Ah	REG_UART2_IIR	R	UART2 – Interrupt Identify Register
230Ah	REG_UART2_FCR	W	UART2 – FIFO Control Register
230Bh	REG_UART2_LCR	R/W	UART2 – Line Control Register
230Ch	REG_UART2_MCR	R/W	UART2 – Modem Control Register
230Dh	REG_UART2_LSR	R	UART2 – Line Status Register
230Eh	REG_UART2_MSR	R	UART2 – Modem Status Register
230Fh	REG_UART2_SCR	R/W	UART2 – Scratch Register
2380h	REG_PTR_DATA	R/W	Printer Port – Data Register
2381h	REG_PTR_CTRL	W	Printer Port – Control Register
2381h	REG_PTR_STATUS	R	Printer Port – Status Register
2382h	REG_MPIO_OUT	R/W	Multi-Purpose-I/O – Output Register
2383h	REG_MPIO_IN	R	Multi-Purpose-I/O – Input Register

Tab. 3: Multi-I/O Unit Registers

4.2 RS232-Interfaces (UARTs)

The both UARTs are mapped into the address range 2300h - 2307h (UART1 / COM1) and 2308h - 230Fh (UART2 / COM2).

Register table of one UART:

Bit No.	REGISTER ADDRESS											
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	2	3	4	5	6	7	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	Modem Control Register	Line Status Register	Modem Status Register	Scratch Register	Divisor Latch (LSB)	Latch (MSB)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0†	Data Bit 0	Enable Received Data Available Interrupt (ERB)	"0" If Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (ΔCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit 0	Receiver FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (ΔDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Transmitter FIFO Reset	Number of Stop Bits (STB)	Out1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable Modem Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 4)	DMA Mode Select	Parity Enable (PEN)	Out2	Framing Error (FE)	Delta Data Carrier Detect (ΔDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 4)	Receiver Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 4)	Receiver Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 4)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

Tab. 4: UART Registers

Please see the datasheet of the 16C550 UART chip for details.

4.3 Keyboard Controller

4.3.1 REG_KEYB_CTRL

The Keyboard Control Register allows direct access to the keyboard serial interface. Also the shift registers are controlled by this register.

Address: 2200h (read/writeable)

MSB(D7)							LSB(D0)
CK2	CK1	PAB	STB	RES	SRR	CLK	DAT

DAT — This bit shows the state of the serial keyboard data line when read. When this bit is written to zero, the data line is tied to GND. When this bit is written to one, the state of the data line is controlled by the keyboard.

CLK — This bit shows the state of the serial keyboard clock line when read. When this bit is written to zero, the clock line is tied to GND. When this bit is written to one, the state of the clock line is controlled by the keyboard.

SRR — Shift register reset. When this bit is set to 1, all bits in the shift registers will be set to their initial state.

RES — Keyboard Reset. This bit was connected with pin 3 of the old 5 pin DIN connector. In the new design with the mini-DIN (PS/2) connector this bit is unused.

STB — This bit is read only. It shows the state of the stop bit of the last received transmission. This bit can be used for error detection.

PAB — This bit is read only. It shows the state of the parity bit of the last received transmission. This bit can be used for error detection.

CK1 — Check Bit 1 (read only). When read, it shows the state of Check Bit 2.

CK2 — Check Bit 2. This bit is zero when read. When this bit is written, its state appears in Check Bit 1. These check-bits are used by the processor to detect the keyboard controller.

4.3.2 REG_KEYB_DATA

Address: 2201h (read only)

This register is read-only. It contains the last data byte that was received from the keyboard.

4.3.3 REG_KEYB_IRES

Address: 2201h (write only)

This is a write-only pseudo register. If something is written to this register, the interrupt logic of the keyboard controller will be reset, and the keyboard clock line is released enabling the keyboard to send more data.

4.4 LC-Display

Note: Please see the HD44780 chip documentation for details.

4.4.1 REG_LCD1_CTRL

Address: 2280h (read/writeable)

First LCD port / signal E1: LCD Control Register

4.4.2 REG_LCD1_DATA

Address: 2281h (read/writeable)

First LCD port / signal E1: LCD Data Register

4.4.3 REG_LCD2_CTRL

Address: 2282h (read/writeable)

Second LCD port / signal E2: LCD Control Register

4.4.4 REG_LCD2_DATA

Address: 2283h (read/writeable)

Second LCD port / signal E2: LCD Data Register

Instruction	Code										Description	Execute Time (max.)
	RS	R/W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	1.64ms
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the Cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the Cursor move direction and specifies or not to shift the display. These operation are performed during data write and read.	40µs
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D) cursor ON/OFF (C), and blink of cursor position character (B).	40µs
Cursor / Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the Cursor and shifts the display without changing DD RAM contents.	40µs
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (N) and character font (F).	40µs
CG RAM Address Set	0	0	0	1	ACG						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40µs
DD RAM Address Set	0	0	1	ADD							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40µs
Busy Flag / Address Read	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	-
CG RAM / DD RAM Data write	1	0	Write Data								Writes data into DD RAM or CG RAM	40µs
CG RAM / DD RAM Data Read	1	1	Read Data								Reads data from DD RAM or CG RAM	40µs

Tab. 5: Summary of LCD Commands (controller HD44780)

4.5 Parallel Printer Port

4.5.1 REG_PTR_DATA

The value written to the Printer Data Register appears on the printer data lines.

Address: 2380h (read/writeable)

MSB(D7)							LSB(D0)
PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0

`PD7 – PD0` Bits of next data byte for the printer.

4.5.2 REG_PTR_CTRL

Printer Control Register. This register controls several wires of the printer port.

Address: 2381h (write only)

MSB(D7)							LSB(D0)
				SEL	RES	AFD	STR

`STR` Data Strobe (low active, 0 = strobe the data)

`AFD` Auto Feed (low active, 0 = automatically feed paper)

`RES` Printer Reset (low active, 0 = reset the printer)

`SEL` Select the Printer (low active, 0 = select the printer)

4.5.3 REG_PTR_STATUS

Printer Status Register. Reflects the state of some wires of the printer port.

Address: 2381h (read only)

MSB(D7)							LSB(D0)
BSY	ACK	PE	FAU	SEL	RES	AFD	STR

`STR` Data Strobe (current state of `STR` -bit in REG_PTR_CTRL register)

`AFD` Auto Feed (current state of `AFD`-bit in REG_PTR_CTRL register)

`RES` Printer Reset (current state of `RES`-bit in REG_PTR_CTRL register)

`SEL` Printer Selected (state of "Select"-output from printer)

`FAU` Printer Fault (low active, 0 = fault)

`PE` Paper Empty

`ACK` Data Acknowledge (low active, 0 = ack)

`BSY` Printer Busy

4.6 Parallel I/O-Port

4.6.1 REG_MPIO_OUT

The value written to the Multi-Purpose Output Register appears at the output pins of connector J9.

Address: 2382h (read/writeable)

MSB(D7)							LSB(D0)
OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

`OUT7 – OUT0` Each bit is connected with the appropriated pin at connector J9.

4.6.2 REG_MPIO_IN

The value read from the Multi-Purpose Input Register reflects the current state of the input pins of connector J9.

Address: 2383h (read/writeable)

MSB(D7)							LSB(D0)
IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0

`IN7 – IN0` Each bit is connected with the appropriated pin at connector J9.

5 Schematics

On the following three pages you will find all schematics of the Multi-I/O-Unit.

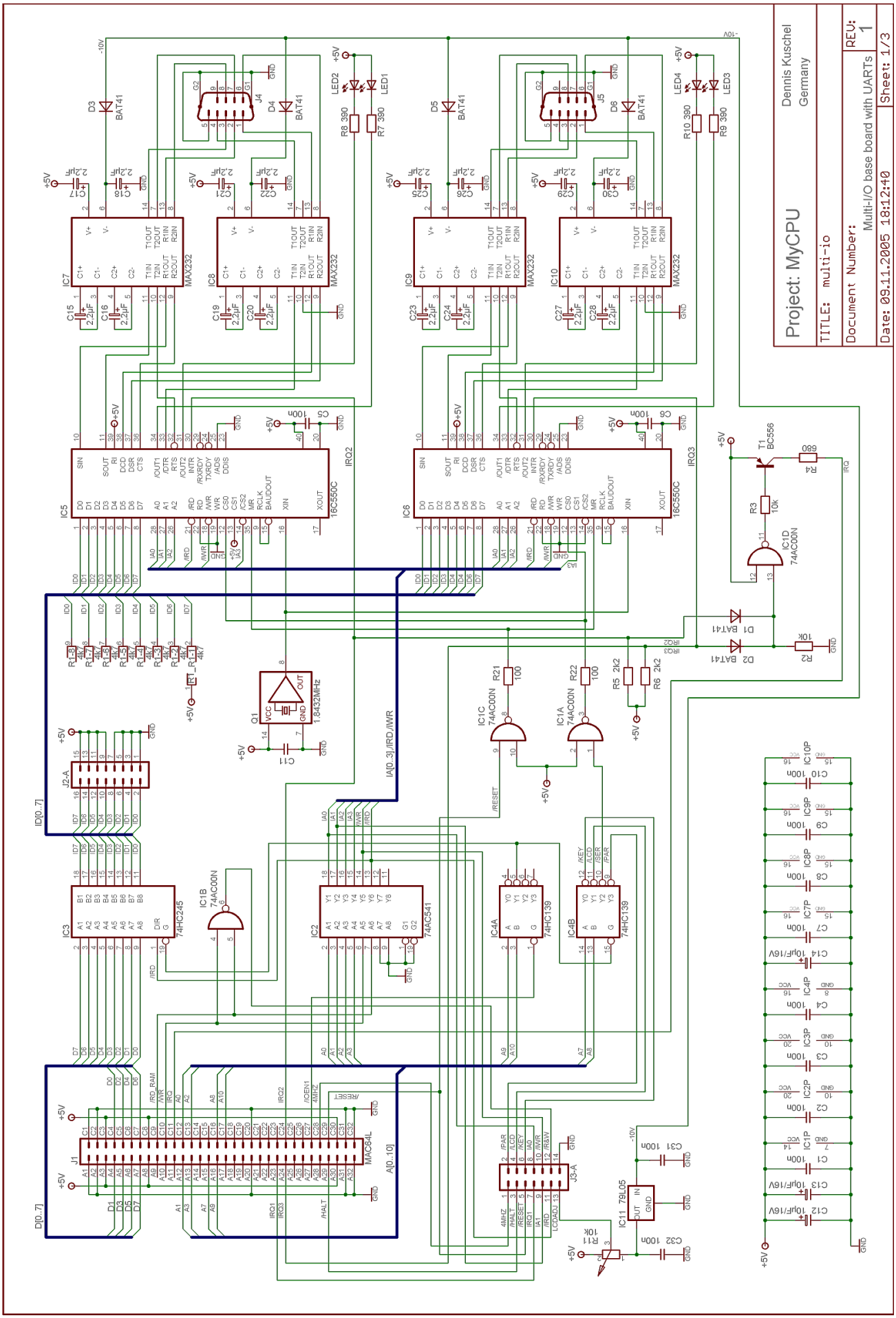


Fig. 21: Schematic of the Multi-I/O Base Board with 2xRS232

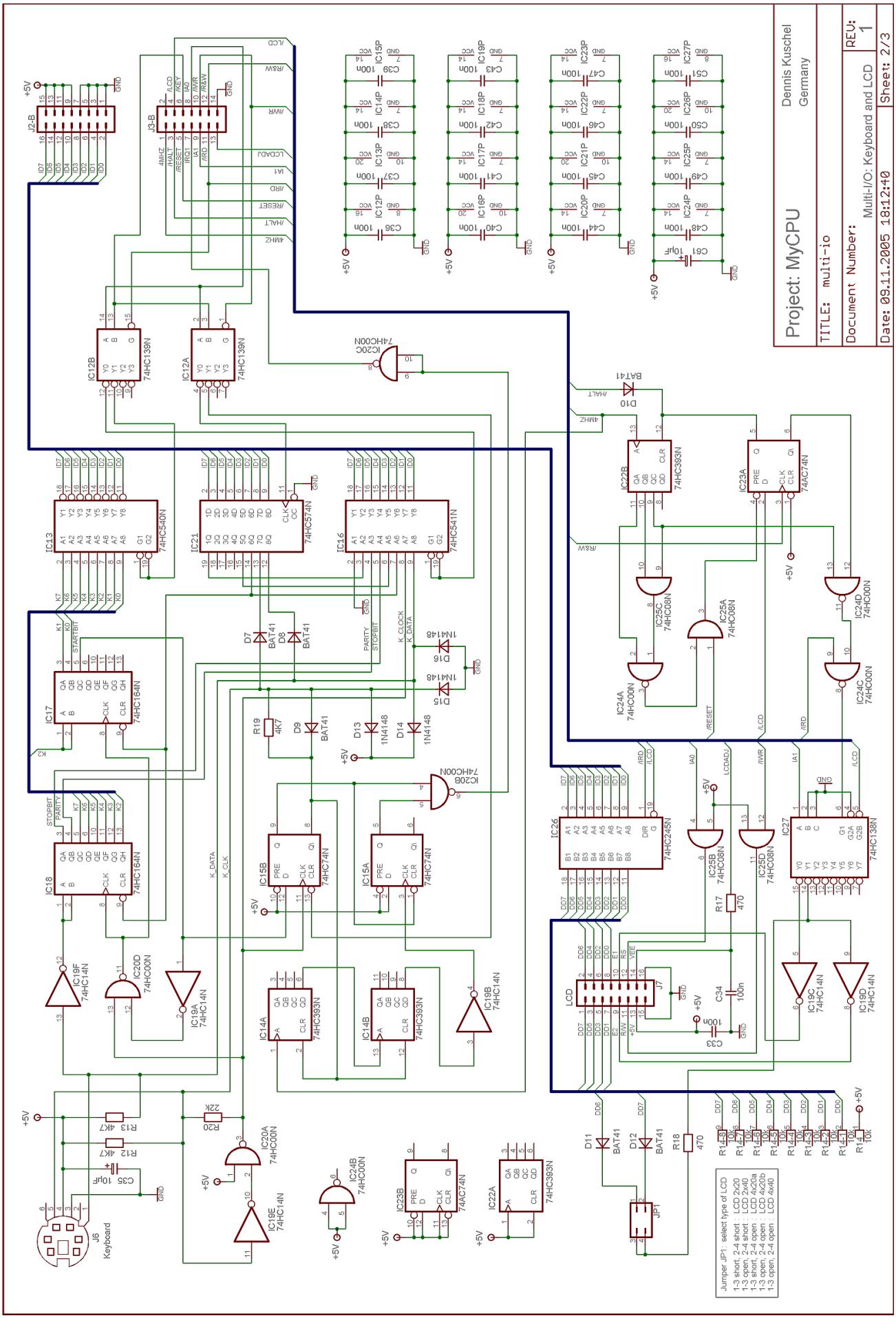


Fig. 22: Schematic of the Keyboard- and LCD- Interface Board

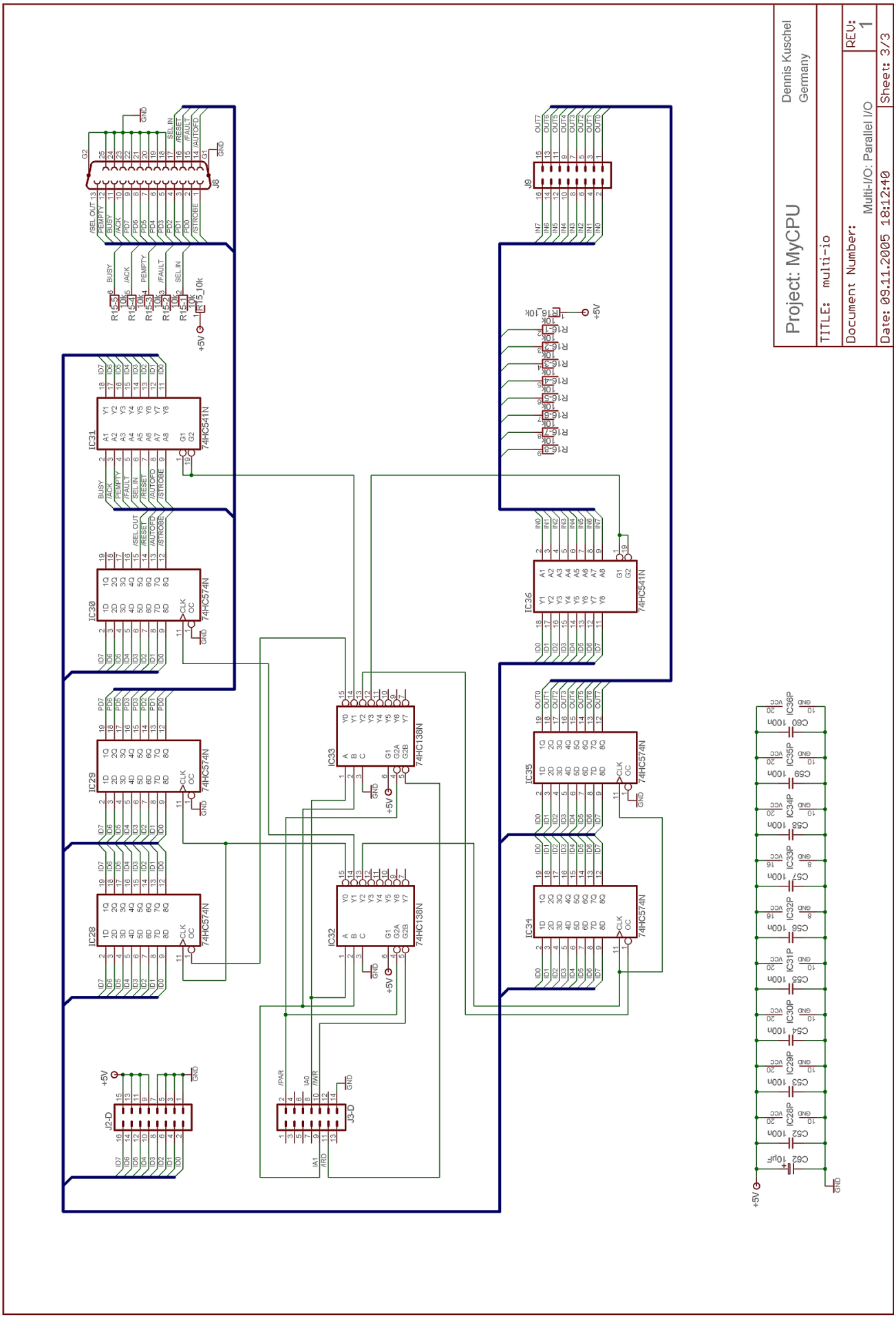


Fig. 23: Schematic of the Parallel Ports Interface Board

6 Change Log

6.1 Changes in the Multi-I/O-design

Date	Name	Chapter	Description
2008-12-22	D.Kuschel	all	Document converted to OpenOffice format
2015-07-14	D.Kuschel		Document revised