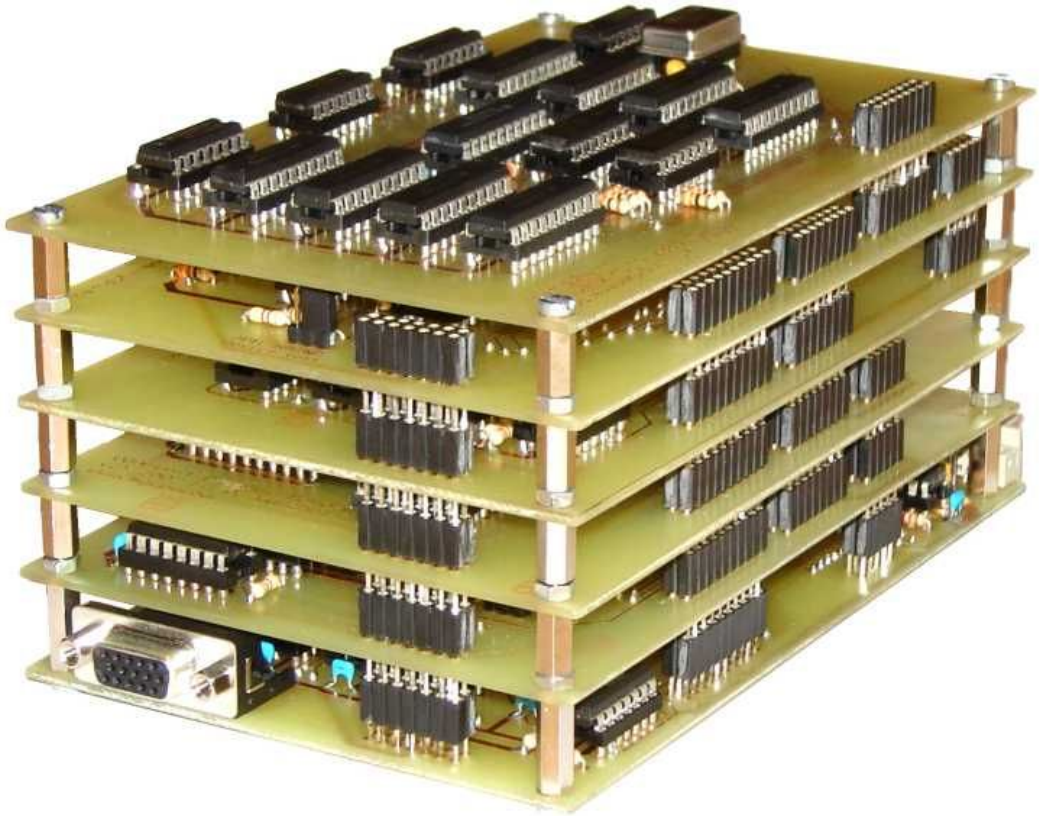
Graphic-Unit

For the MyCPU

- Selfbuild Guide -



uses `/IOEN2:2C00h-2FFFh`

Dennis Kuschel · Corintostraße 21 · 28279 Bremen · Germany · www.mycpu.eu · dennis_k@freenet.de 2015-07-23

Content

Figures

1 Overview

Features:

- 6 printed circuit boards with 75 low integrated circuits
- 16 Colors, 640x400 Pixel, 70 Hz mode
- includes a character generator for 8 different character sets
- text modes supported by the Operating System: 80x25, 80x50, 40x25, 40x50
- graphic modes supported by the Operating System: 640x400x2, 160x200x16, 320x200x16, 640x400x16
- the Graphic Unit can be connected to nearly every microcontroller

Limitations:

- Only the 640x400 pixel mode is supported. But all available CRT displays should support this mode since it is the same mode that nearly every PC uses to display the BIOS or boot screen.
- There is no support for so called “sprites”. Sprites would make developing games more easy. The logic for sprites is too heavy to implement (too much parts would be required for a small feature that is rarely used).

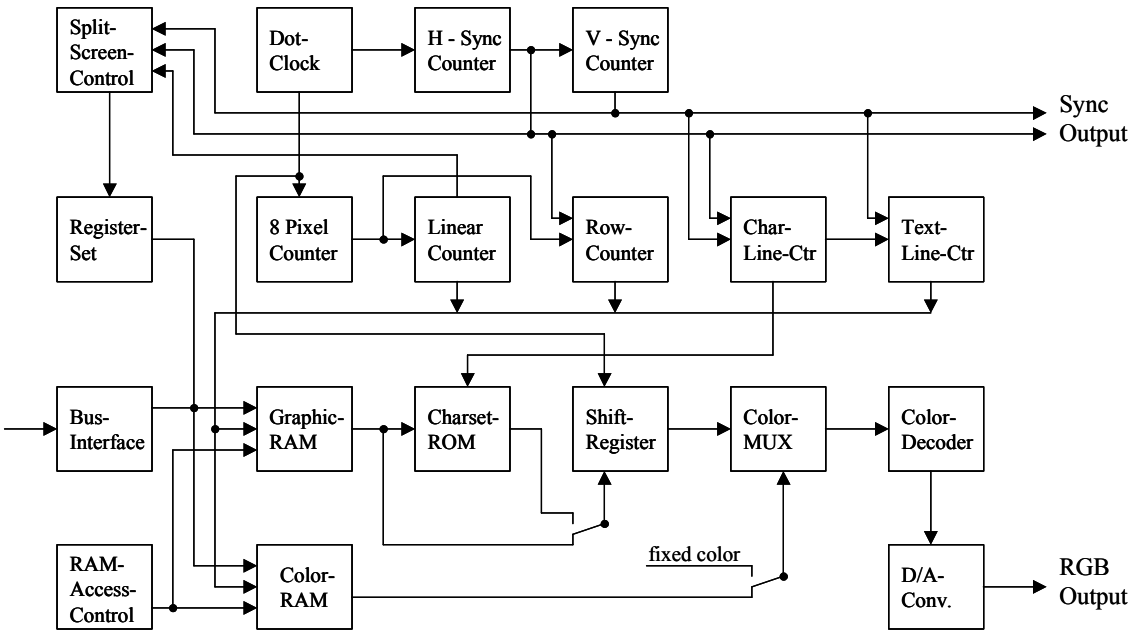


Fig. 1: Overview

Author: Dennis Kuschel
Corintostraße 21
28279 Bremen, Germany

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

2 Introduction

2.1 Overview

This document describes how to build a Graphic Unit (or commonly said, a VGA graphic adapter) for the MyCPU. Please follow this guide step by step, and build the printed circuit boards in exactly that order this document suggests. The Graphic Unit is a bit more complex than the MyCPU itself, but the advantage is that a bug in the Graphic Unit does not crash the computer system but only leads to graphic errors.

When you follow this document you will see that every single board is tested. It is really useful if you have access to an oscilloscope. But if you do not own an oscilloscope you will still be able to test the boards, but seeking errors may be a bit lengthy.

ATTENTION !

The boards require some special parts!

Before you start building the Graphic Unit please make sure you can obtain the special 74ACxxx - parts. Also two fast RAMs and a fast EPROM is required.
You may replace some 74ACxxx - parts through their 74HCxxx equivalents but then there is no guarantee that the graphic unit will work as expected. In this case you will need to try several 74HCxxx-parts maybe from different vendors until you'll find a combination of integrated circuits that works for you. The special parts are:

74AC00, 74AC08, 74AC14, 74AC32, 74AC74, 74AC541, 74AC574 (logic gates)
62256-70 (32KB RAM with 70ns or faster access time)
27C256-80 (32KB EPROM with 80ns or faster access time)

Here are some ideas for replacements:
A 74AC541 can be replaced by a 74AC245 with a bit of adaptation circuitry.
In some rare cases you can replace a 74AC574 through a 74AC373. A combination of a 74AC245 and a 74AC373 would be a full replacement for a 74AC574.

Hint:
It is possible to use the Graphic Unit with reduced resolution. With reduced resolution the dot clock is only the half and 74HCxxx circuits can be used. Note that only the big character sets are available (text resolutions 40x25 and 40x50), and the hires graphic mode is disabled. Please see chapter 4 for a detailed description.

2.2 Before you start

- Please make sure that you have the newest firmware for the MyCPU. You need at least the **Kernel version 2.10**. To find out what firmware you have installed, please start the MyCPU and type the command "`ver`" at the shell prompt.
- Try to obtain the required 74ACxxx -parts. If you can't get the parts, you can still start constructing the Graphic Unit. Please see chapter 4 for further details.
- You will need an oscilloscope for measuring and testing the boards. It's possible to test the boards without an oscilloscope but finding errors is much more complicated.
- Please follow this guide step by step. Build the boards in the order they are described here.

You should build and test the boards in the following order:

1. Build the Timing Generator Board.
2. Test the Timing Generator Board with an oscilloscope.
3. Build the Pixel Control Board.
4. Test the Pixel Control Board together with the Timing Generator Board with an oscilloscope.
5. Build the Interface Board.
6. Test the Interface Board together with the Timing Generator and the Pixel Control Board. Attach a CRT or TFT Display to the Interface Board. You should now see vertical stripes on the Display.
7. Build the Color Memory Board.
8. Test this board together with the other three. Attach the board stack to the MyCPU. Connect a CRT or TFT Display to the interface board. When you switch on the MyCPU, you should see some colored horizontal lines on the display followed by much colored garbage.
9. Build the Address-Counter Board.
10. Test this board together with the other four. Attach the board stack to the MyCPU. Connect a CRT or TFT Display to the interface board. When you switch on the MyCPU, you should see much colored garbage on the screen. Now enter the commands "gutest 1" and "gutest 2" at the shell prompt to generate the test pattern.
11. Build the Graphic-Memory Board.
12. Test all boards together with the MyCPU. Congratulations! You are done now.

3 Boards

3.1 Timing Generator Board

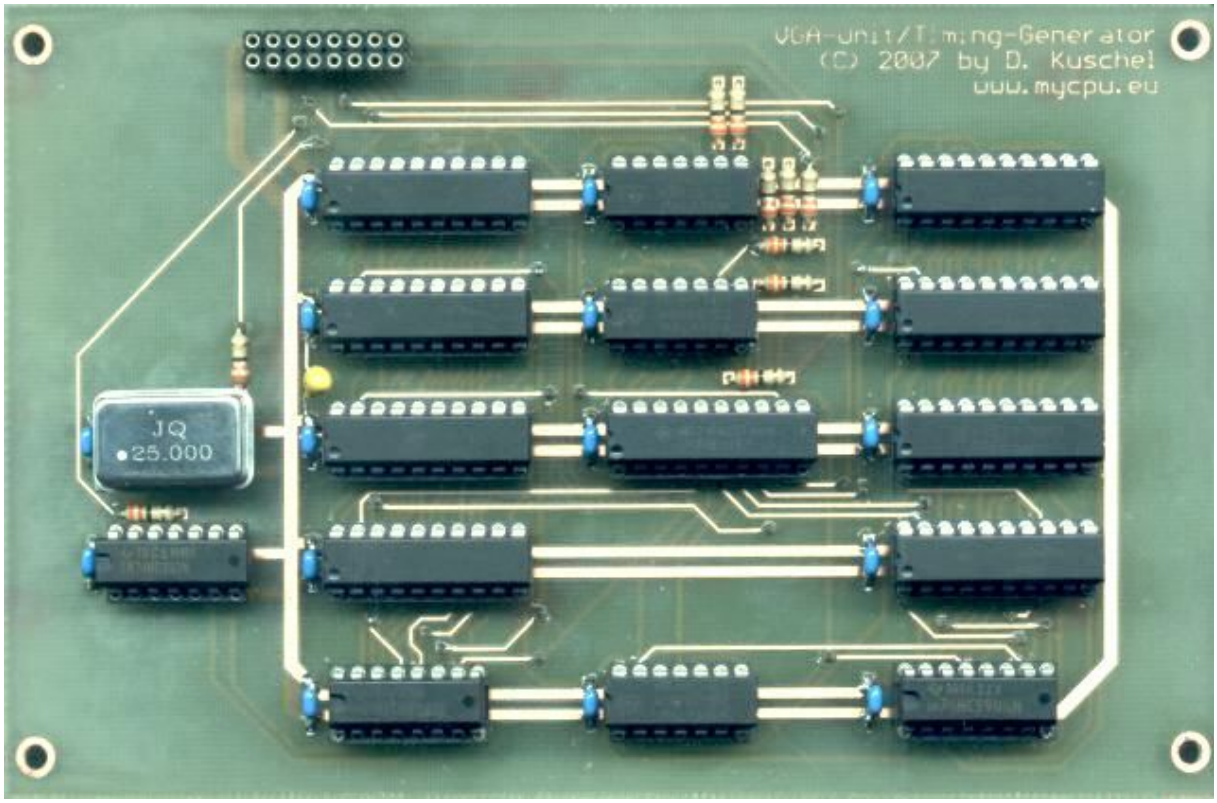


Fig. 2: Timing Generator Board

3.1.1 Description

The Timing Generator Board provides the basic timing signals for the Graphic Unit. The basic signals are HSYNC, VSYNC and pixel-clock ("dot clock"). It also provides the blanking signals that switch off the video in the hidden area of the screen. All signals are also provided in the inverted form.

3.1.2 Selection of Components

There are no 74ACxxx parts used in the circuitry, so only the easy-to-obtain 74HCxxx parts are used. A bit special is the required oscillator with 25.175 MHz. Although this is the standard VGA frequency, also a 25.000 MHz oscillator will do fine for the case you have trouble obtaining the 25.175 MHz oscillator.

3.1.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the 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 color in the placeplan below. Please check if you have really soldered these pads!

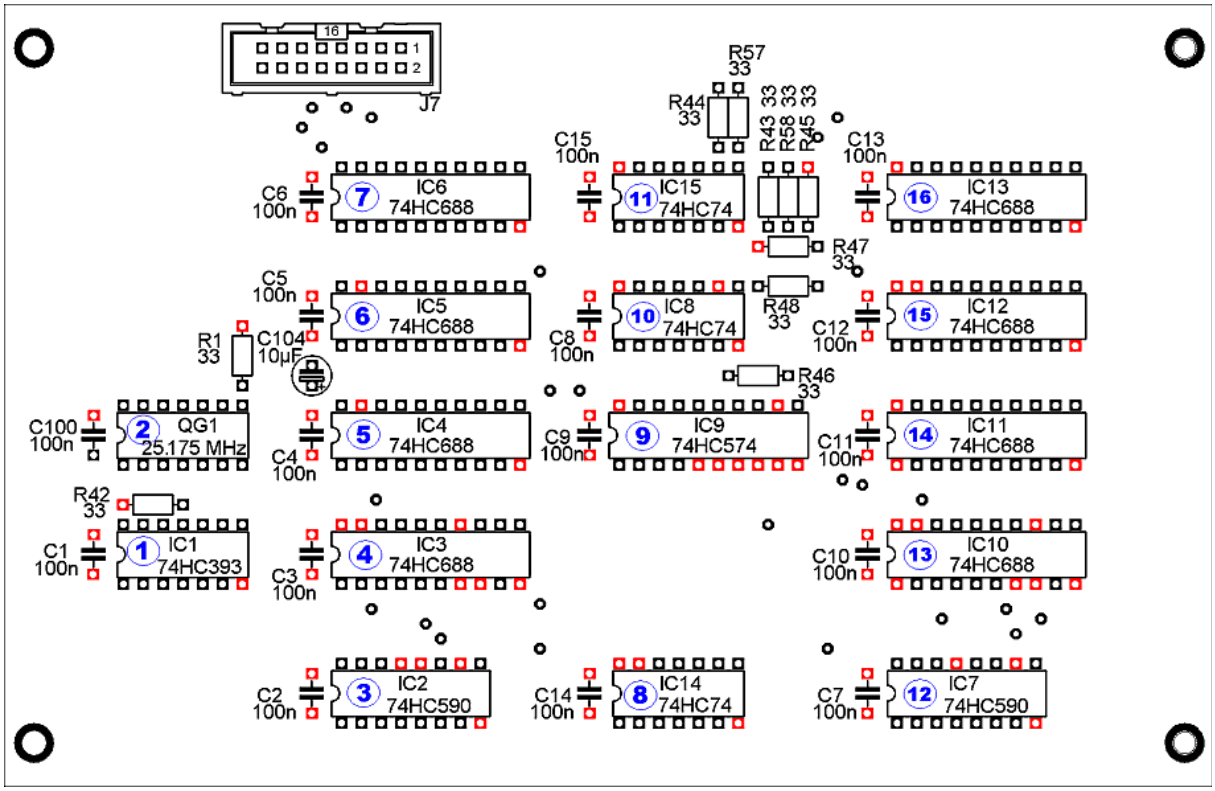


Fig. 3: Timing Generator Board - Placeplan

3.1.4 Partlist

74HC74	IC8, IC14, IC15
74HC393 [1)]	IC1
74HC574	IC9
74HC590	IC2, IC7
74HC688	IC3, IC4, IC5, IC6, IC10, IC11, IC12, IC13
Oscillator 25.175 MHz	QG1
33 Ohm	R1, R42, R43, R44, R45, R46, R47, R48, R57, R58
100nF ceramic capacitor	C1 – C15, C100
10µF / 16V, tantal	C104
16 pin header	J7

Footnote:

[1)] If you observe that your VGA screen is showing a jittering image (lines are randomly shifted by 8 pixel from left to right), you can try to use a 74AC393 instead of the proposed 74HC393.

3.1.5 Testing

For testing the timing generator board you will need a 5V power supply and an oscilloscope. Please apply the 5V to the pins 16 (+5V) and 15 (GND) of J7.

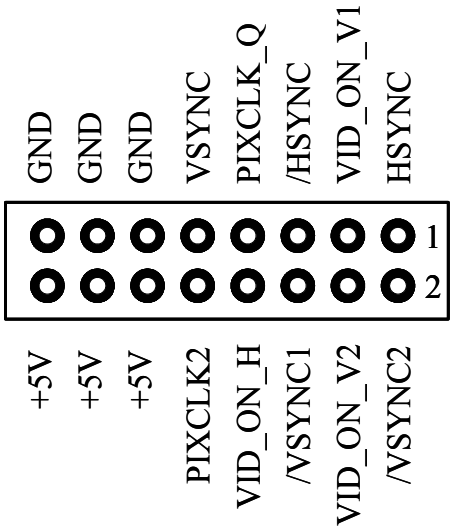
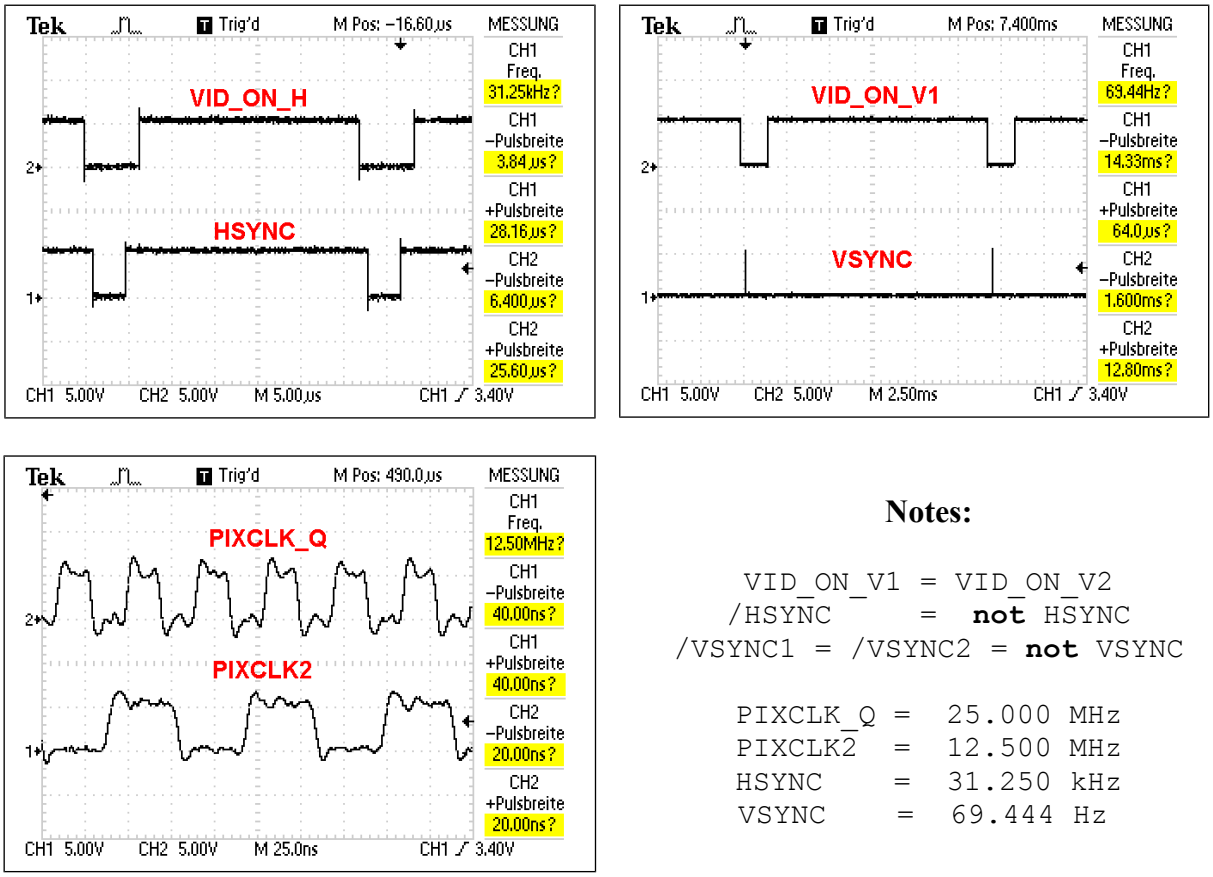


Fig. 4: Signals on connector J7



Notes:

```
VID_ON_V1 = VID_ON_V2
/HSYNC    =  not HSYNC
/VSYNC1 = /VSYNC2 = not VSYNC

PIXCLK_Q =  25.000 MHz
PIXCLK2  =  12.500 MHz
HSYNC    =  31.250 kHz
VSYNC    =  69.444 Hz
```

Fig. 5: Measured Signals on J7 with QG1 = 25.000 MHz

3.2 Pixel Control Board

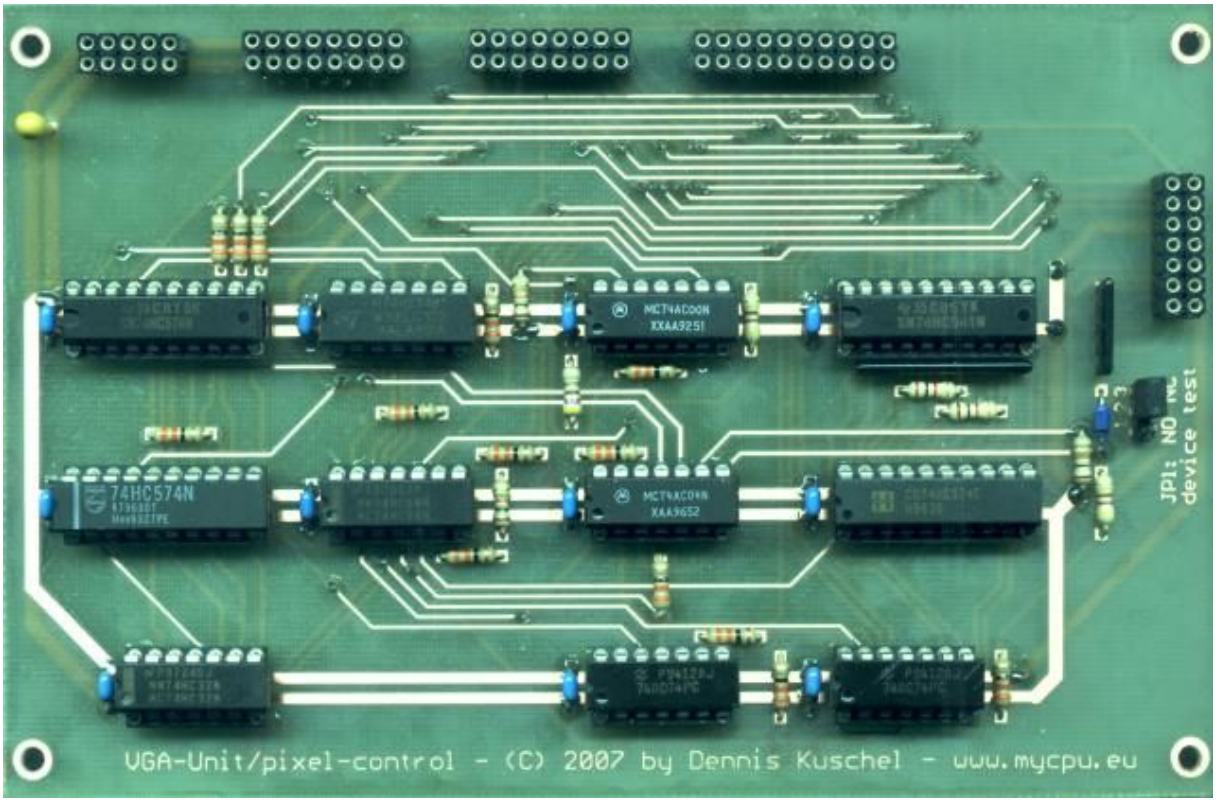


Fig. 6: Pixel Control Board

3.2.1 Description

The Pixel Control Board contains a very fast shift register that is used to generate the serial pixel stream for the RGB outputs. Furthermore some complex timing logic is on this board: For example the timing signals for synchronous RAM access, for the Character Generator and the 640x400 plain graphic mode are generated here.

3.2.2 Selection of Components

There are several 74ACxxx parts used in the circuitry. I strongly recommend to use the 74AC-types. If you can't obtain these parts you will get big trouble with the circuitry.
At least the 74AC04, 74AC08 and the 74AC74's are mandatory !!!!
Please see chapter 4 for a possible solution.

3.2.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the 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 color in the placeplan below. Please check if you have really soldered these pads!

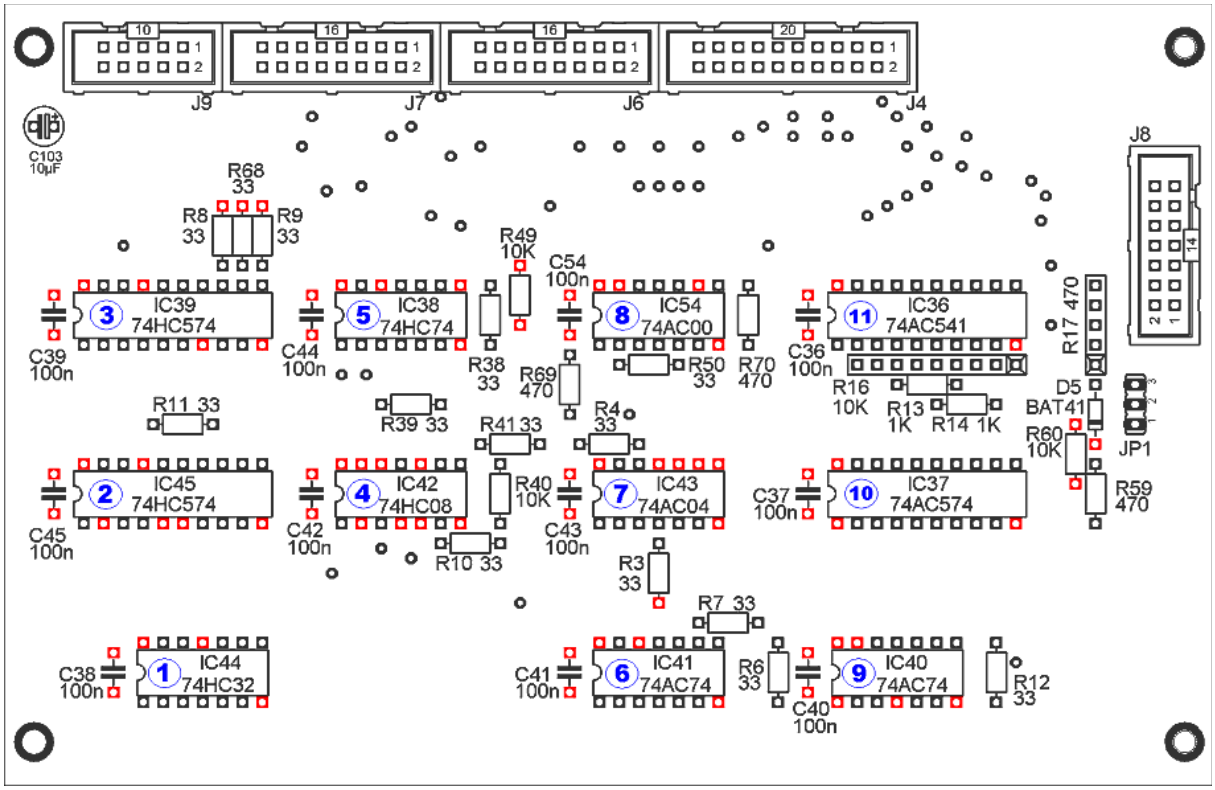


Fig. 7: Pixel Control Board - Placeplan

3.2.4 Partlist

74AC00	IC54
74AC04	IC43
74HC08	IC42
74HC32	IC44
74AC74	IC40, IC41
74HC74	IC38
74AC541	IC36
74AC574	IC37
74HC574	IC39, IC45
BAT 41	D5
33 Ohm	R3, R4, R6 - R12, R38, R39, R41, R50, R68
470 Ohm	R59, R69, R70
1 K	R13, R14
10 K	R40, R49, R60
SIL 4 x 470 Ohm	R17
SIL 8 x 10 K	R16
100nF ceramic capacitor	C36 – C45, C54
10µF / 16V, tantal	C103
3 pin jumper block	JP1
10 pin header	J9
14 pin header	J8
16 pin header	J6, J7
20 pin header	J4

3.2.5 Connectors

The connection to the other boards is done with board-to-board connectors. I prefer the wire-wrap-connectors with long pins. With this connectors it is easy to stack the boards.

Unfortunately these connectors are no more available. For a variant of making the connection please refer to the "MyCPU Selfbuild Guide", chapter 3 and "Memory Unit Selfbuild Guide", chapter 2.

Fig. 8: Wire-Wrap Connector

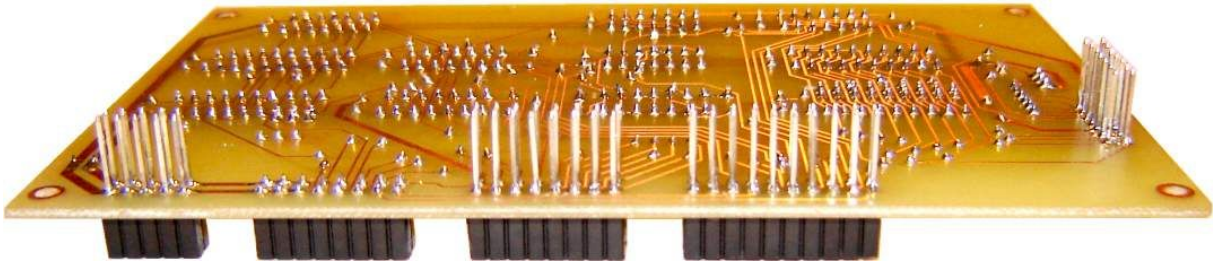
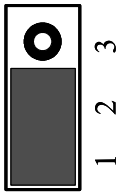


Fig. 9: Board with wire-wrap connectors

3.2.6 Testing

Please follow the following steps carefully!

3.2.6.1 Put the Jumper JP1 to the test-position 1-2.



The shown jumper position is used for testing the Interface Board. In fact for testing the Pixel Control Board the jumper position is irrelevant, but before you forget to put the jumper to the test-position when testing the Pixel Control Board, put the jumper to the test-position just now.

3.2.6.2 Put the boards together

Mount the Timing Generator Board on top of the Pixel Control Board:

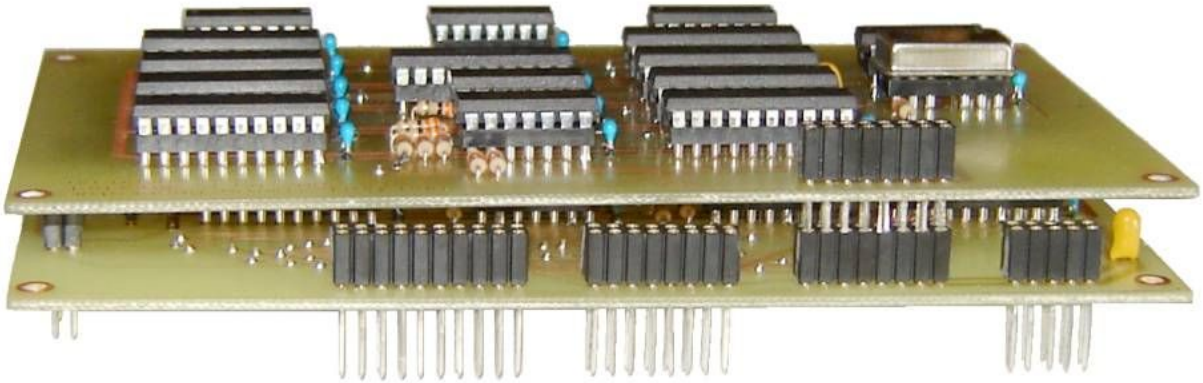
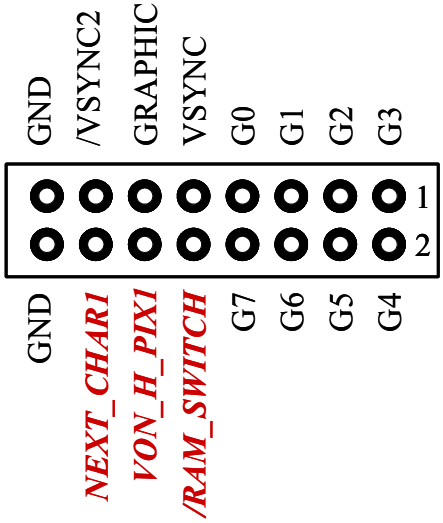
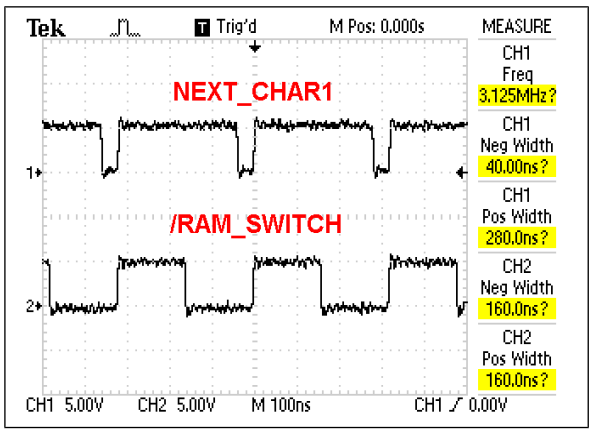


Fig. 10: Timing Generator mounted on Pixel Control Board

3.2.6.3 Measure the signals on J6





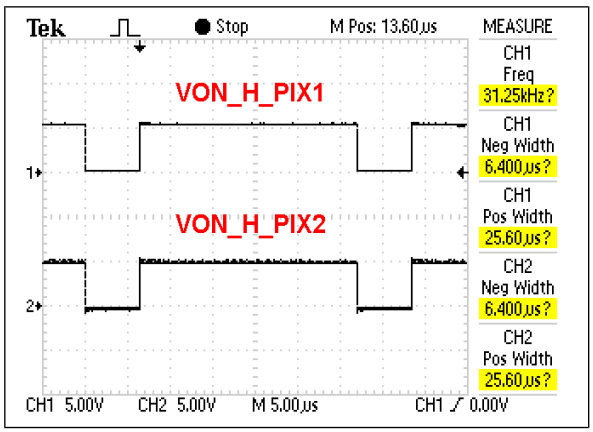
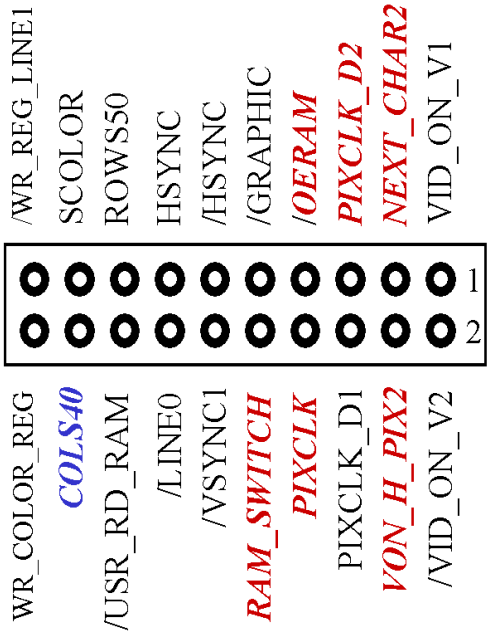
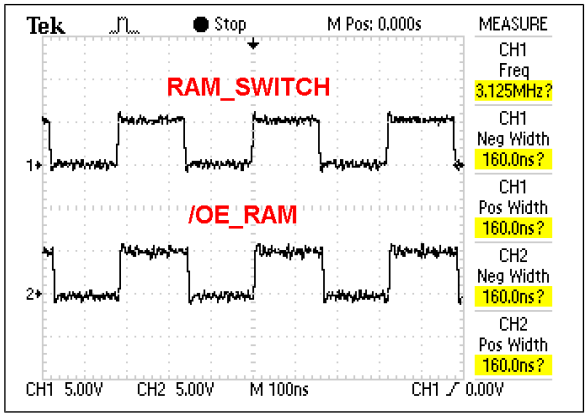


Fig. 11: Measured signals on Connector J6 with QG1 = 25.000 MHz

3.2.6.4 Measure the signals on J4





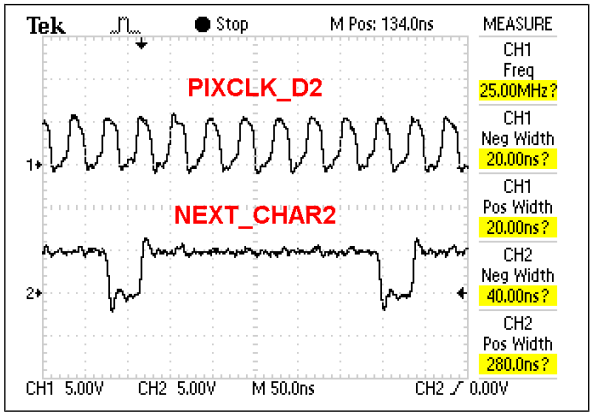
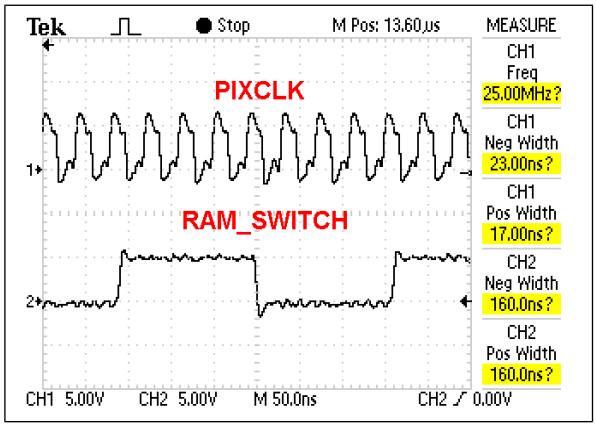
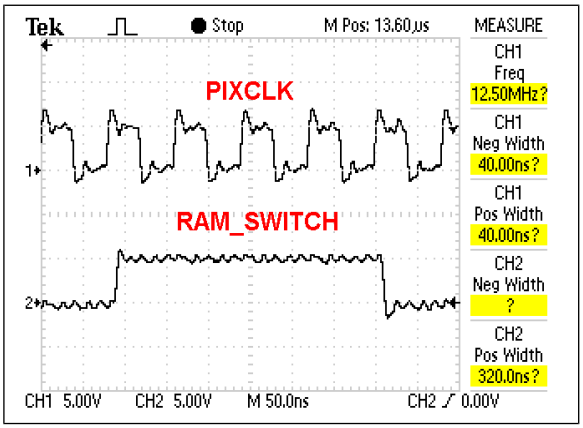


Fig. 12: Measured signals on Connector J6 with QG1 = 25.000 MHz

Please test the COLS40-Signal:
PIXCLK must have ½ of the Crystal-Frequency when COLS40 is tied to +5V



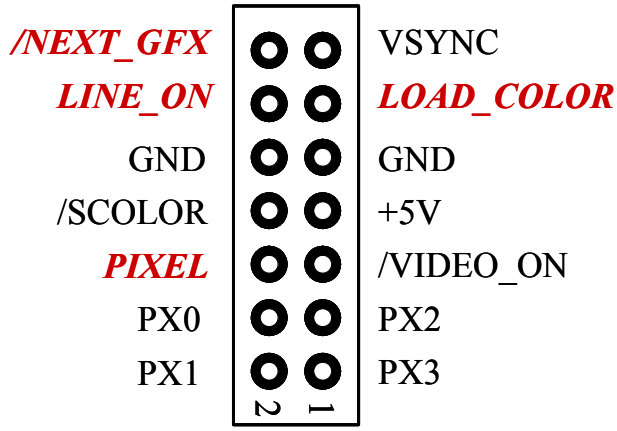
COLS40 left open



COLS40 tied to +5V

Fig. 13: Test of the COLS40 –signal with QG1 = 25.000 MHz

3.2.6.5 Measure the signals on J8



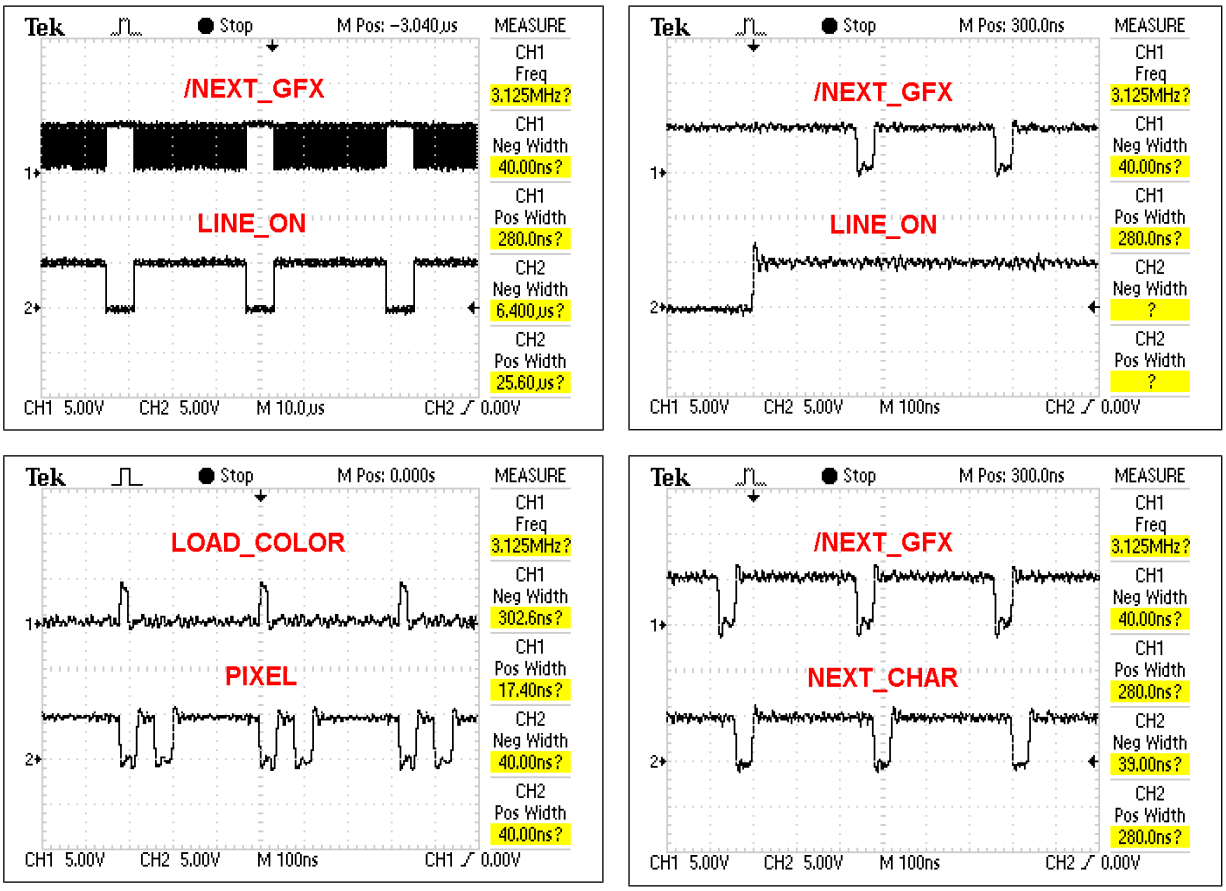


Fig. 14: Measured signals on Connector J8 with QG1 = 25.000 MHz

3.3 Interface Board

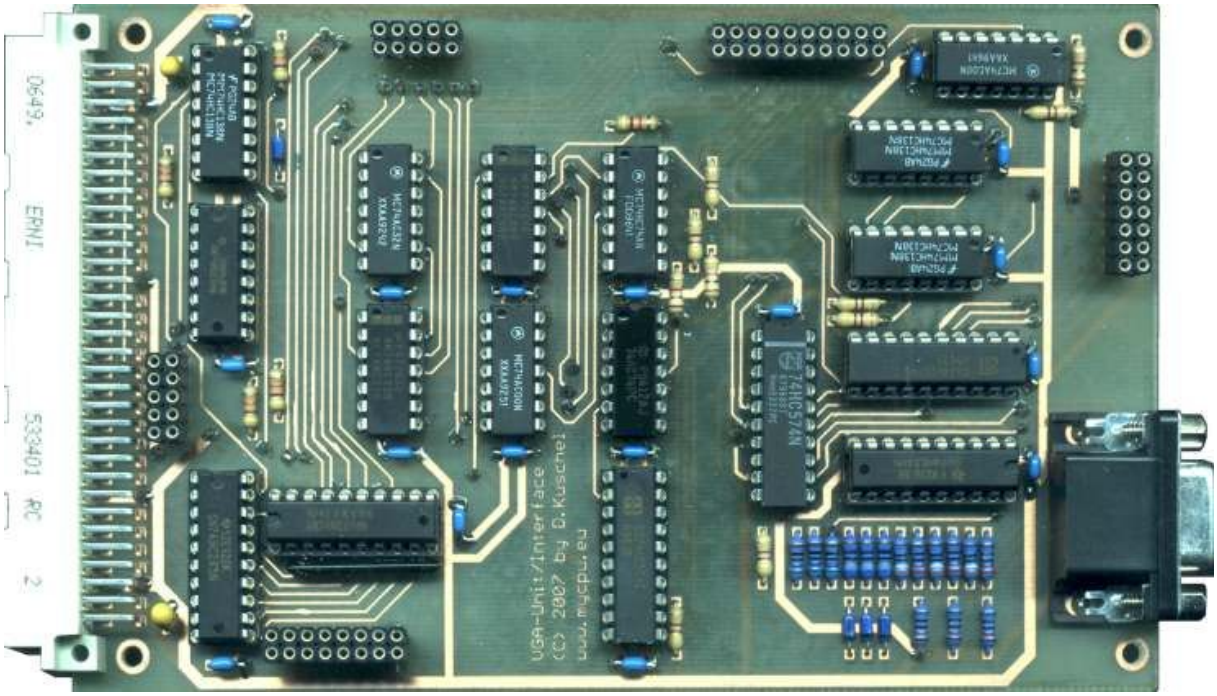


Fig. 15: Interface Board

3.3.1 Description

The Interface Board has connectors for the MyCPU bus and the analogue VGA display. The MyCPU bus part consists of some bus glue logic, address decoder, bus driver and bus timing control. For the VGA output there is a simple digital-to-analogue converter on the board.

3.3.2 Selection of Components

There are several 74ACxxx parts on the board. If you want to use the Graphic Unit with a high MyCPU bus speed (above 4 MHz) you must use 74ACxxx types for the IC69, IC70, IC72 and IC73. The IC49 and IC53 must always be 74ACxxx types. You may try 74HCxxx types for IC49 and IC53, but then the picture on the CRT may not be sharp.

3.3.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the connectors are placed and soldered.

When soldering the connectors please keep in mind that you need more space between the interface board and the board that sits on top of the interface board. This is because the bus connector X2 is a bit higher than the other parts. Please see Fig. 17 for details.

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 color in the placeplan below. Please check if you have really soldered these pads!

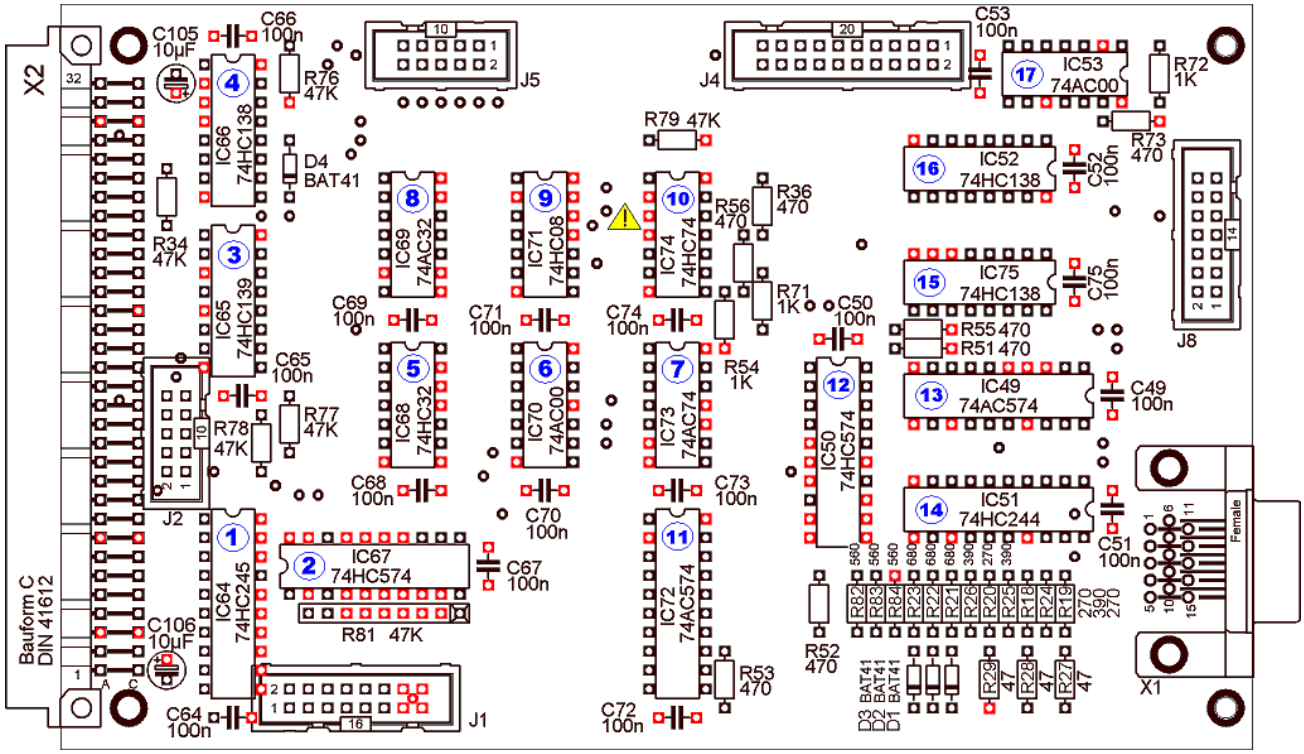


Fig. 16: Interface Board - Placeplan

3.3.4 Partlist

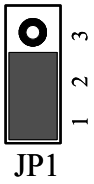
74AC00	IC53, IC70
74HC08	IC71
74AC32	IC69
74HC32	IC68
74AC74	IC73
74HC74	IC74
74HC138	IC52, IC66, IC75
74HC139	IC65
74HC244	IC51
74HC245	IC64
74AC574	IC49, IC72
74HC574	IC50, IC67
BAT 41	D1, D2, D3, D4
47 Ohm	R27, R28, R29
270 Ohm	R18, R19, R20
390 Ohm	R24, R25, R26
470 Ohm	R36, R51, R52, R53, R55, R56, R73
560 Ohm	R82, R83, R84
680 Ohm	R21, R22, R23
1 K	R54, R71, R72
47 K	R34, R76, R77, R78, R79,
8 x 47 K	R81
100nF ceramic capacitor	C49 - C53, C64 - C75

10µF / 16V, tantal	C105, C106
10 pin header	J2, J5
14 pin header	J8
16 pin header	J1
20 pin header	J4
15 pin fine-pitch Sub-D VGA connector, female	X1
DIN 41612 Connector	X2

3.3.5 Testing

Please follow the following steps carefully!

3.3.5.1 Set Jumper JP1 to the test-position 1-2.



The jumper JP1 resides on the Pixel Control Board. Please put the jumper to the shown position to enable the test-mode for the Interface Board.

3.3.5.2 Put the boards together

Please stack all boards like shown on the picture. The Interface Board is at the bottom and the Timing Generator Board is on top.

Fig. 17: All three boards mounted together

3.3.5.3 Test the Interface and the VGA output

Connect the three boards of the Graphic Unit with the MyCPU and a CRT display.
Power on the MyCPU. You should now see this test pattern on the screen:

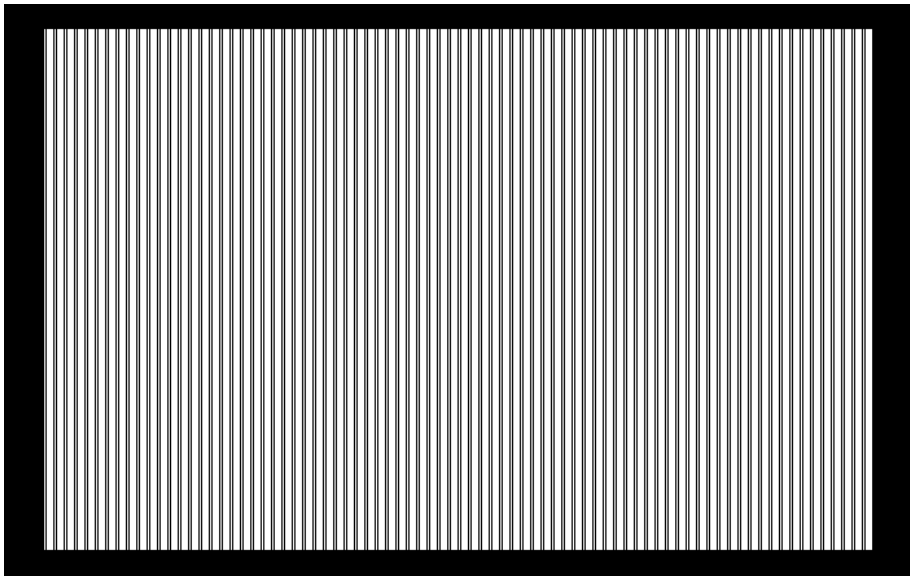


Fig. 18: Test Pattern 1

Now enter the command "`eb $2F80 $10`" at the MyCPU shell prompt. The test pattern should now change to this:

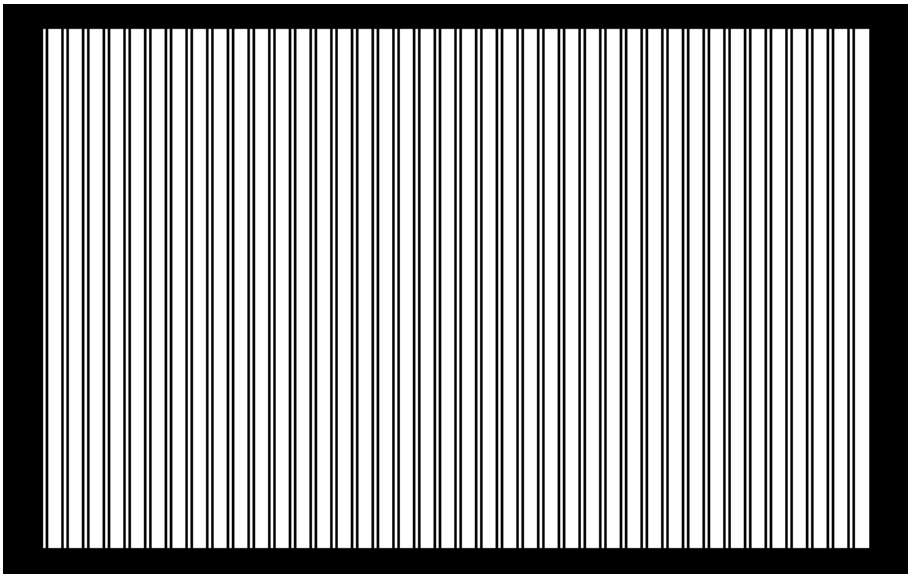


Fig. 19: Test Pattern 2

Note: The command "`eb $2F80 $00`" shows test pattern 1 again

If you do not see any of the test pattern please check the parts at the right side of the Interface Board. If the command "`eb $2F80 $10`" does not have any effect, the bus-interface does not work properly. Please check the parts on the left side of the Interface Board.

3.4 Color Memory Board

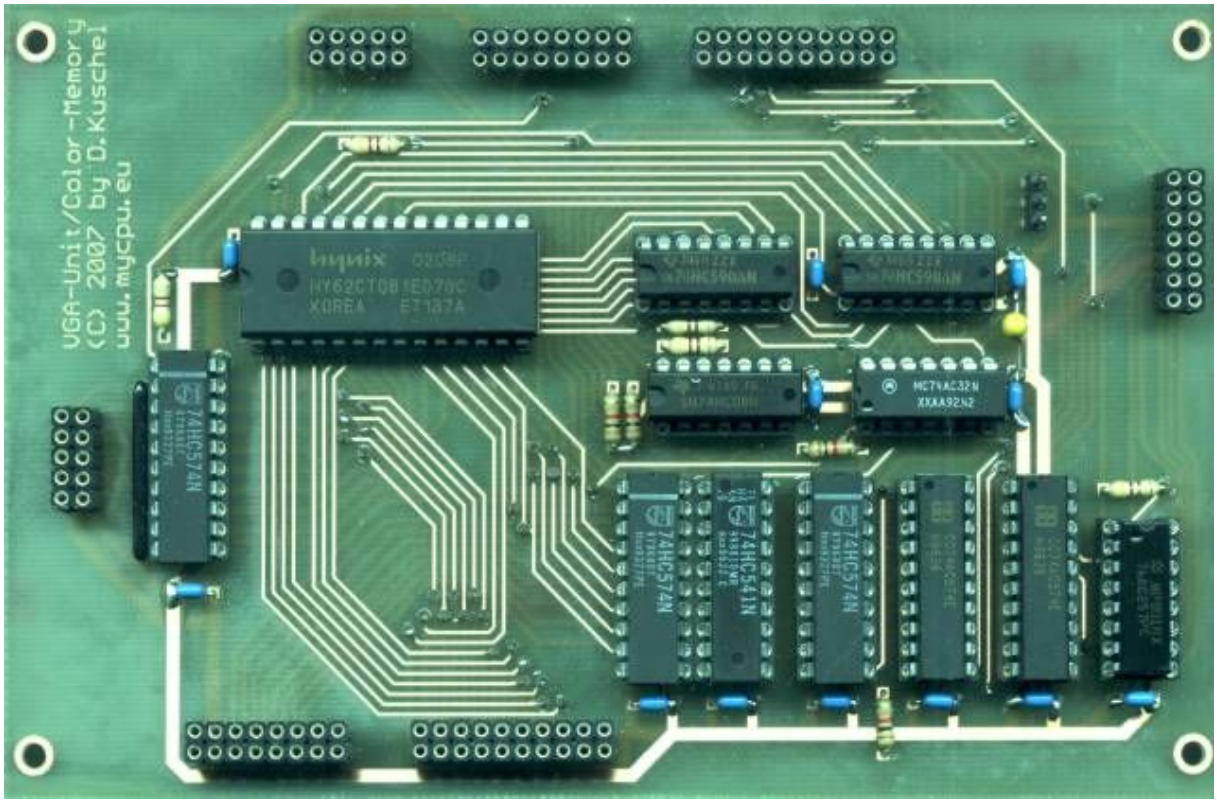


Fig. 20: Color Memory Board (prototype without R74 and R75)

3.4.1 Description

The Color Memory Board is populated with the Color RAM and an address counter for the high resolution graphic mode. The Color RAM only contains color information and no pixel data. One byte in the RAM contains the foreground color (lower 4 bit) and the background color (upper 4 bit) for 8 pixel in series. That means, 8 consecutive pixel in the horizontal direction can only have two different colors.

3.4.2 Selection of Components

There are several 74ACxxx parts used in the circuitry. I strongly recommend to use the 74AC-types. If you can't obtain these parts please consider to reduce the graphic resolution. Please see chapter 4 for details. **A fast RAM (70ns or faster) is mandatory!**

3.4.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the 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 color in the placeplan below. Please check if you have really soldered these pads!

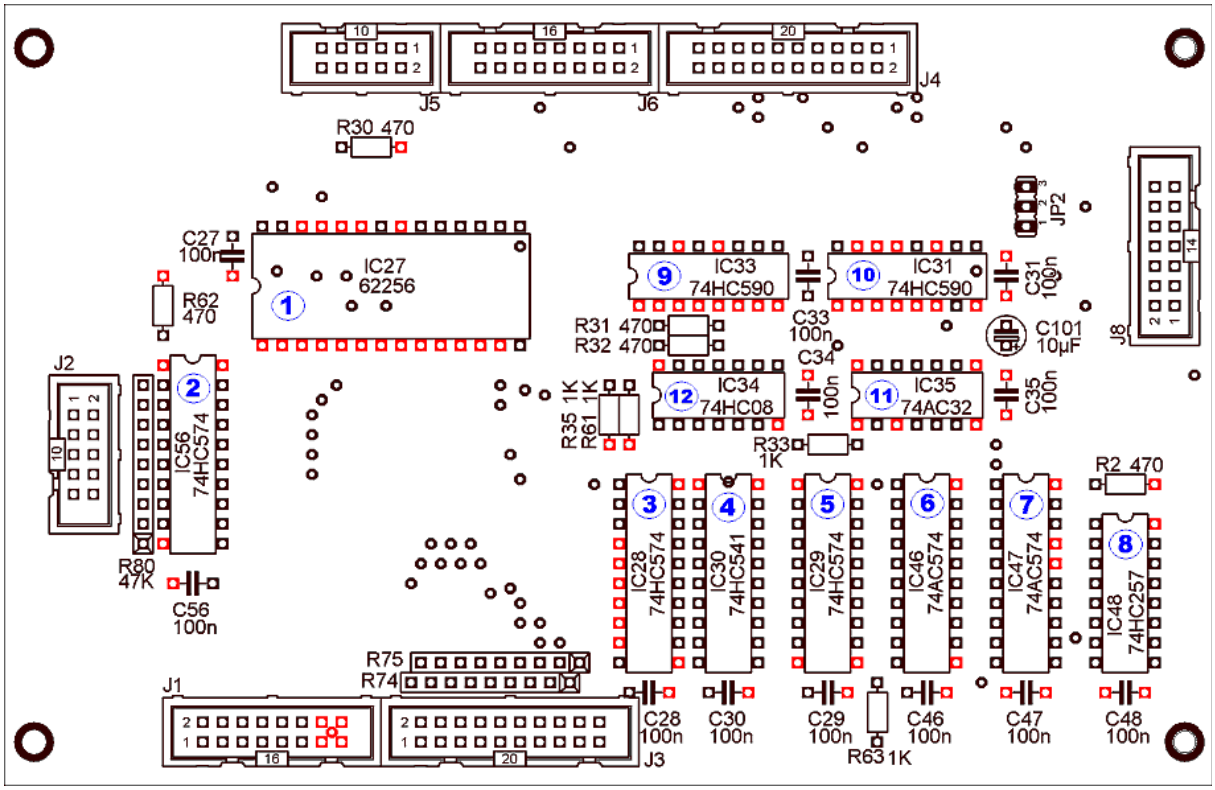


Fig. 21: Color Board - Placeplan

3.4.4 Partlist

74HC08	IC34	
74AC32	IC35	
74HC257	IC48	
74HC590	IC31, IC33	
74HC541	IC30	
74AC574	IC46, IC47	
74HC574	IC28, IC29, IC56	
RAM 62256 70ns	IC27	***70ns or faster !***
470 Ohm	R2, R30, R31, R32, R62	
1 K	R33, R35, R61, R63	
SIL 8 x 10K	R74, R75	
SIL 8 x 47K	R80	
100nF ceramic capacitor	C27-C31, C33-C35, C46-C48, C56	
10µF / 16V, tantal	C101	
10 pin header	J2, J5	
14 pin header	J8	
16 pin header	J1, J6	
20 pin header	J3, J4	
3 pin jumper block	JP2	

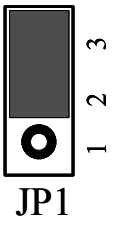
3.4.5 Testing

Please follow the following steps carefully!

You must have Kernel version 2.10 or newer installed, otherwise the tests will fail!
You can check the Kernel version by entering the command "`ver`" at the shell prompt.

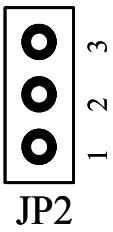
3.4.5.1 Set Jumper JP1 to the normal position 2-3.

Set Jumper JP1 (on the Pixel Control Board) back to the normal position 2-3:



3.4.5.2 Leave Jumper JP2 open

Do not set Jumper JP2. The jumper is not required because testing the "/GRAPHIC"-signal is done through the MyCPU bus interface.



3.4.5.3 Put the boards together

Please stack all boards like shown on the picture. The Color Memory Board is inserted between the Pixel Control Board and the Interface Board:

Color Memory Board

Fig. 22: All four boards mounted together

3.4.5.4 Test Pattern

Please connect the 4 boards of the Graphic Unit with the MyCPU and power on the system. You should now see the following test pattern. Please note the colored bar at the top of the screen. If you see the bar on your screen, everything is OK:

Fig. 23: Test Pattern 3

If you can't see the 4 colored stripes in the first upper lines in Pattern 3 most probably the Bus Interface does not work. Please check all data- and address-lines on the Color Memory Board and check all parts on the left side of the Bus Interface Board.
Important: You must have Kernel version 2.10 or newer installed !!

Now enter the command " `eb $2F80 $30` " at the MyCPU shell prompt. The test pattern should now change to this:

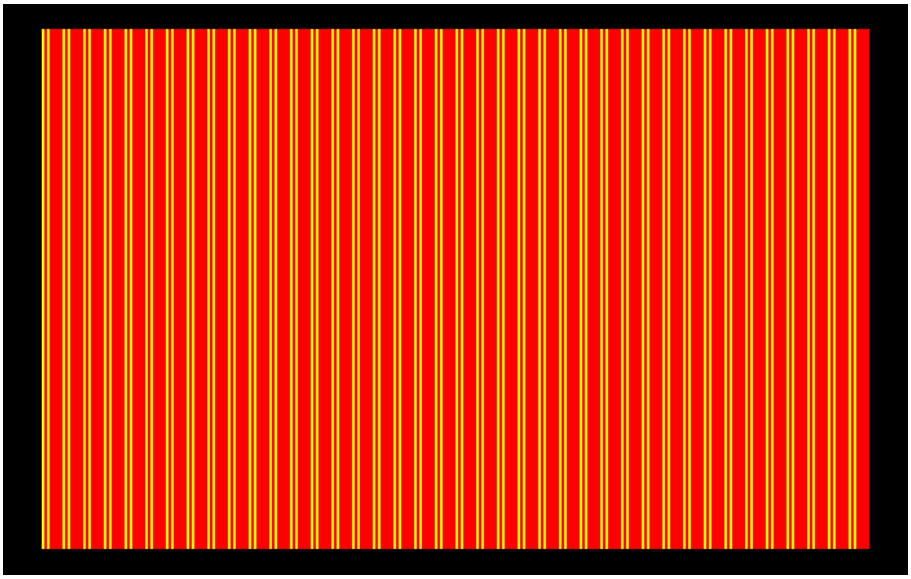


Fig. 24: Test Pattern 4

You can change the colors of the stripes by writing some values to the register $2F00. Example: " `eb $2F00 $1B` "

3.5 Address Counter Board

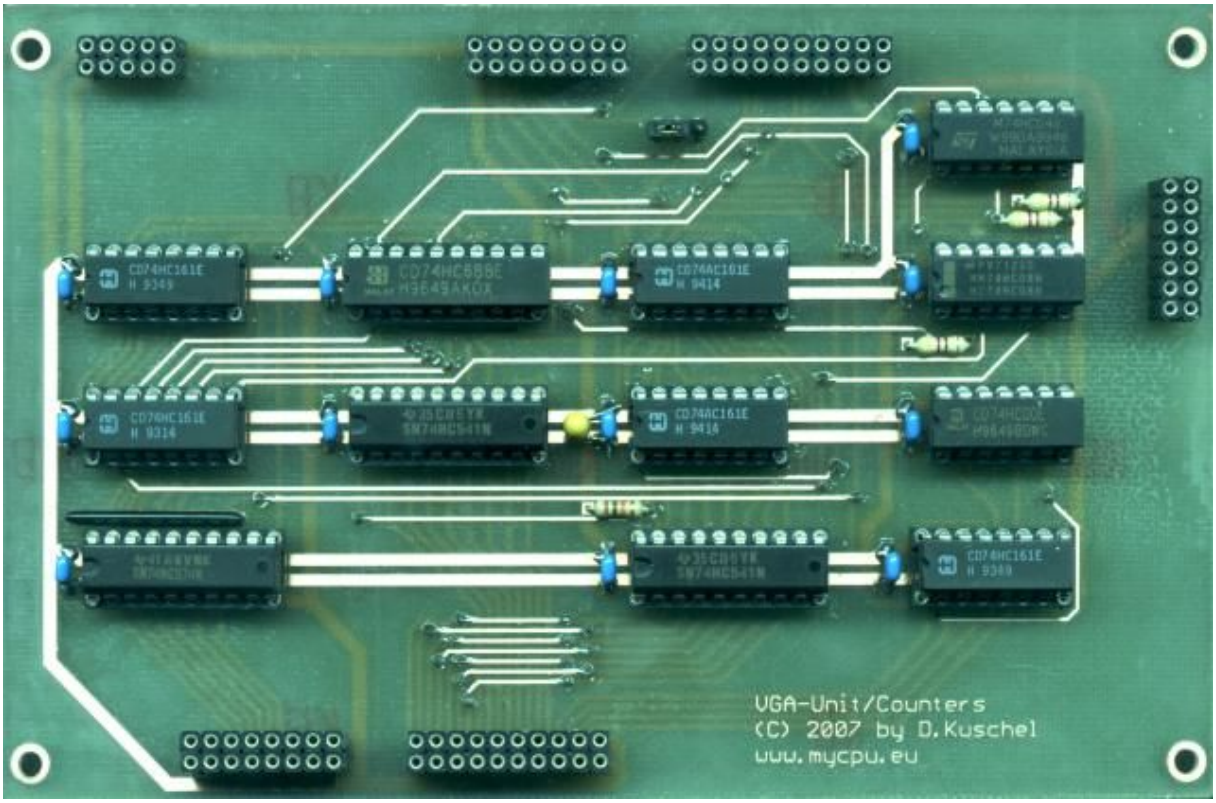


Fig. 25: Address Counter Board (Prototype)

3.5.1 Description

The Address Counter Board contains the logic that is required for the Text Modes. A text line on the screen is max. 80 characters wide, in the RAM the text line is stored in a portion of 128 bytes of memory (48 bytes remain unused). Furthermore this board contains a hardware scroll logic for the text screen, so the CPU is not required to copy the memory when the screen must scroll up.

3.5.2 Selection of Components

There are some 74ACxxx parts on the board. If you are using very fast RAMs (70ns or faster) for the color- and the graphic-memory, you may replace the 74ACxxx types by their 74HCxxx counterparts without any side effect.

3.5.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the 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 color in the placeplan below. Please check if you have really soldered these pads!

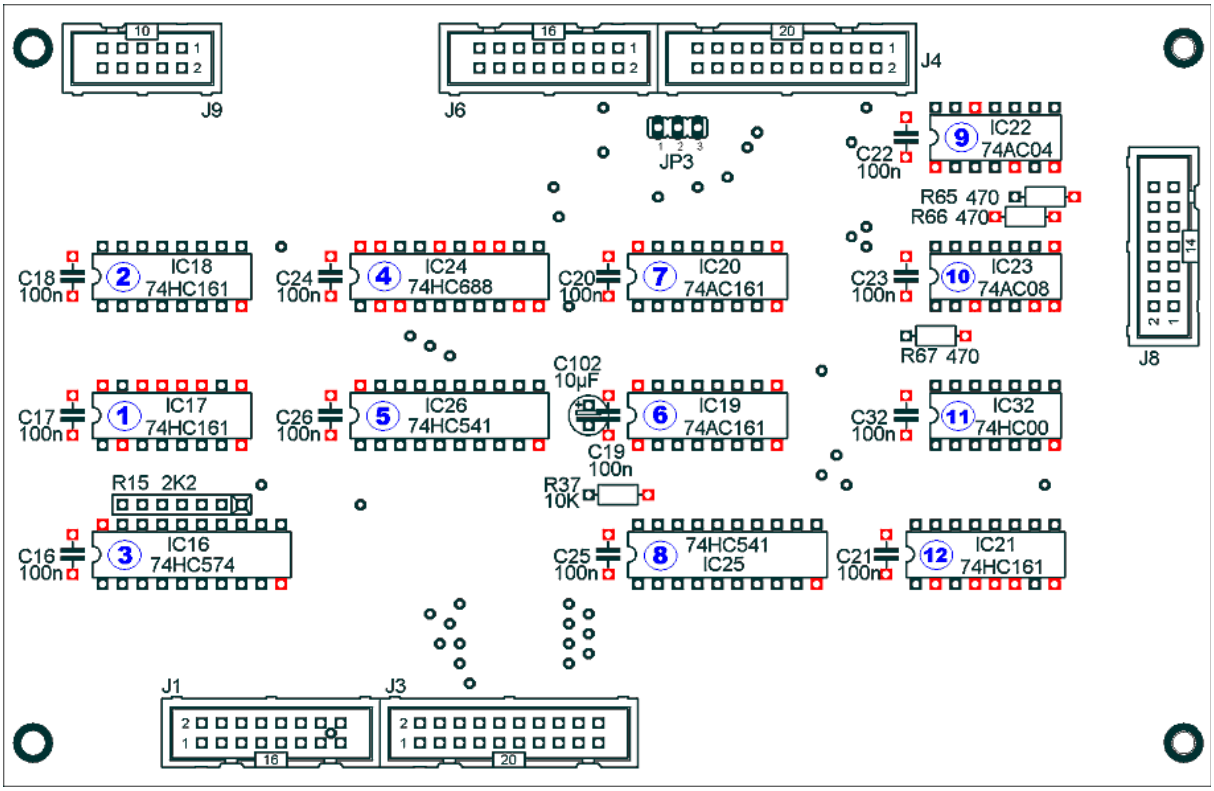


Fig. 26: Address Counter Board - Placeplan

3.5.4 Partlist

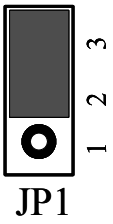
74HC00	IC32
74AC04	IC22
74AC08	IC23
74AC161	IC19, IC20
74HC161	IC17, IC18, IC21
74HC541	IC25, IC26
74HC574	IC16
74HC688	IC24
470 Ohm	R65, R66, R67
10 K	R37
SIL 6 x 2K2	R15
100nF ceramic capacitor	C16 - C26, C32
10µF / 16V, tantal	C102
10 pin header	J9
14 pin header	J8
16 pin header	J1, J6
20 pin header	J3, J4
3 pin jumper block	JP3

3.5.5 Testing

Please follow the following steps carefully!

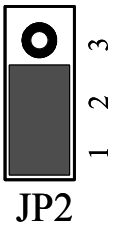
3.5.5.1 Set Jumper JP1 to the normal position 2-3.

Check if Jumper JP1 (on the Pixel Control Board) is still in the normal position 2-3:



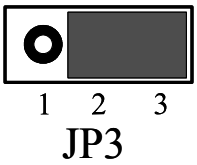
3.5.5.2 Set Jumper JP2 to the test position 1-2.

Set Jumper JP2 (on the Color Memory Board) to the test position 2-3:



3.5.5.3 Set Jumper JP3 to the test position 2-3.

Set Jumper JP3 (on the Address Counter Board) to the test position 2-3:



3.5.5.4 Put the boards together

Please stack all boards like shown on the picture. The Address Counter Board is inserted between the Pixel Control Board and the Color Memory Board:

Timing-Generator

Pixel-Control

Address-Counter

Color-Memory

Interface

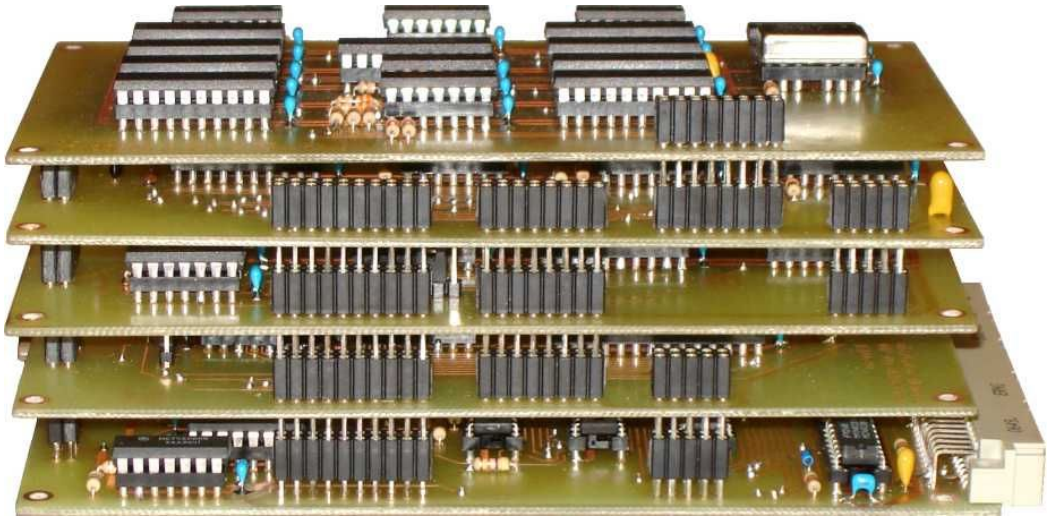
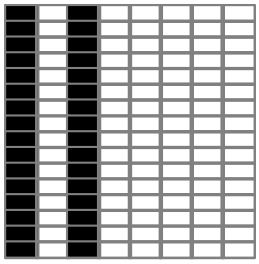


Fig. 27: All five boards mounted together

3.5.5.5 Power on the MyCPU

Connect the Graphic Unit with the MyCPU. Power on the MyCPU system and watch the screen. You should see a random colored picture. The picture is divided into 80x25 fields where a field looks like this:



You should see this field 2000 times on the screen.
Note: When the foreground and the background color is the same in the field, you won't see the both stripes.

3.5.5.6 Test the Scroll Logic

Please enter the command "`gutest 1`" at the shell prompt directly after powering on the MyCPU. You should see the test pattern moving up and down. If the movement is not smooth but maybe "zigzag", you may want to check the IC16 and the logic around.
Stop the test by pressing CTRL+C.

3.5.5.7 Test the Address Counter

Please enter now the command "`gutest 2`" at the shell prompt. After a while you should see 80 colored stripes going from top to the bottom of the screen. Note that there may still remain some garbage characters on the screen (this is not an error).

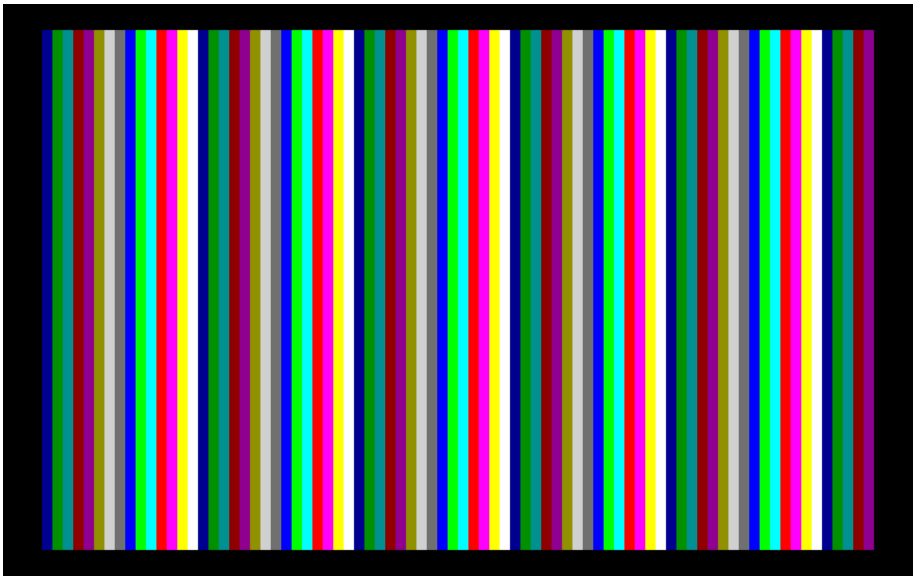


Fig. 28: Test Pattern 5

3.5.5.8 Test the 50 Lines Mode

After you have entered "`gutest 2`" at the shell prompt, enter now the command "`eb $2f80 $08`". The vertical lines of test pattern 5 are now disturbed by some garbage. The screen should show (from top to bottom) vertical lines – garbage – vertical lines. There should be 50% vertical lines and 50% garbage on the screen. After entering "`gutest 2`" at the shell prompt again the garbage should disappear.

3.6 Graphic Memory Board

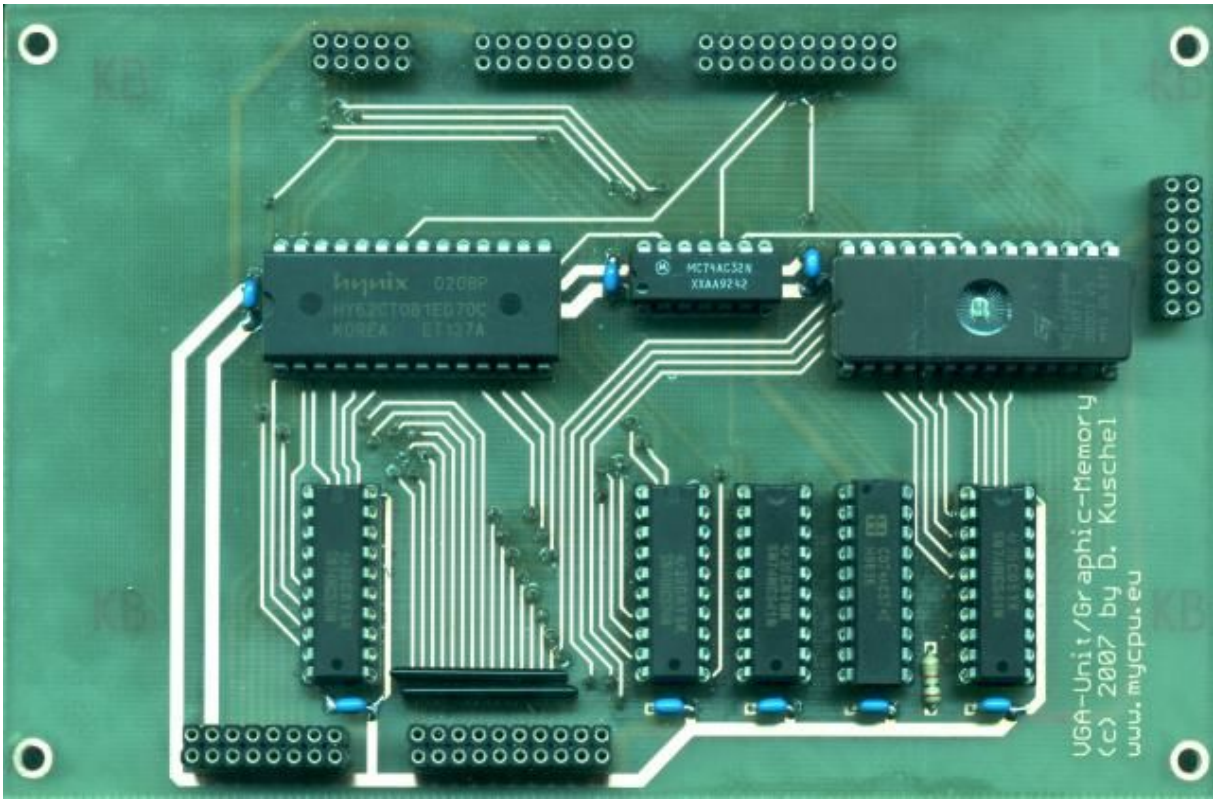


Fig. 29: Graphic Memory Board (prototype)

3.6.1 Description

This board contains the graphic memory (for pixel-data and ASCII characters) and a charset generator. The charset generator is realized by an EPROM that can be bypassed for the highres graphic modes.

3.6.2 Selection of Components

There is one 74AC32 on the board. This one **should** be a AC type, but **may** be a HC type. The RAM and the EPROM must be very fast types. The RAM must have 70ns or less. The EPROM that was used in the prototype has an access time of 80ns.

3.6.3 Placement of Components

After you have soldered all VIA's, you can continue with the integrated circuits. I strongly recommend you not to use sockets for the IC's because it is nearly impossible to solder all sockets correctly to a self-etched PCB. A manufactured PCB does not have this limitation. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I 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 and the 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 color in the placeplan below. Please check if you have really soldered these pads!

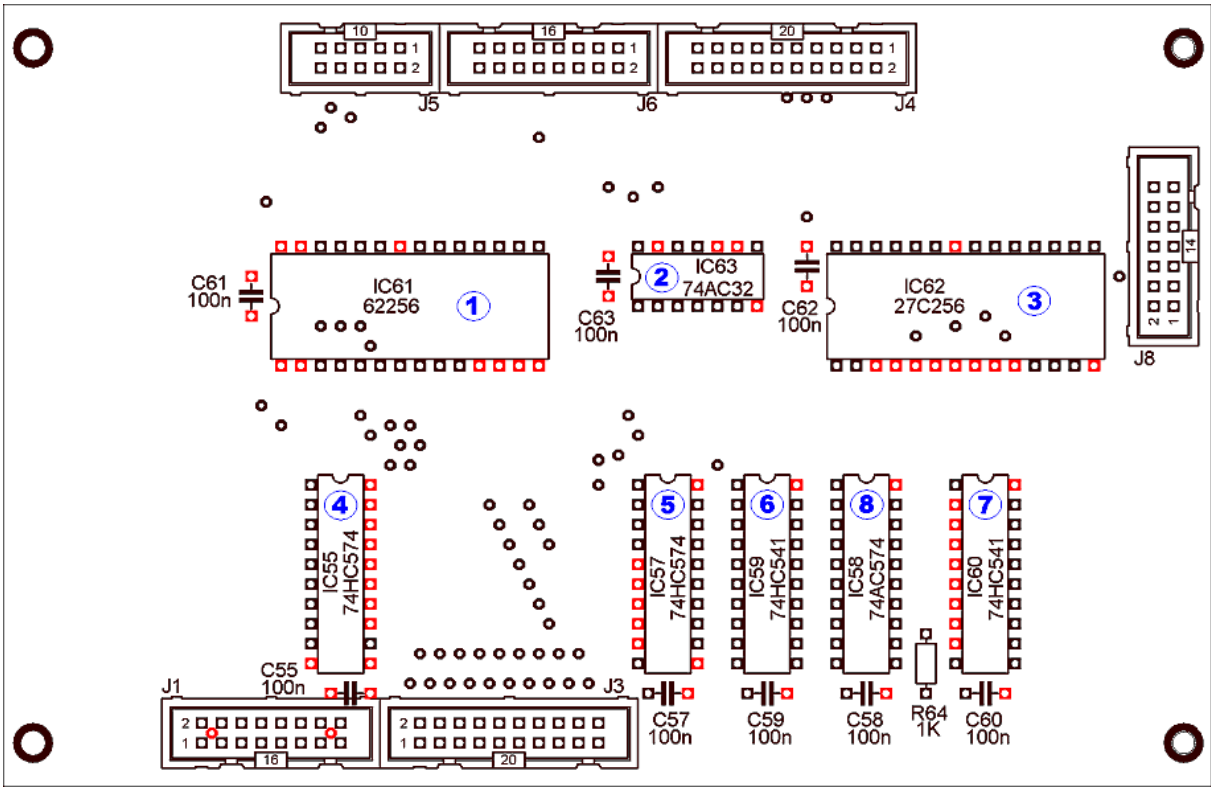


Fig. 30: Graphic Memory Board - Placeplan

3.6.4 Partlist

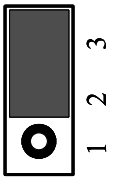
74AC32	IC63	
74HC541	IC59, IC60	
74AC574	IC58	
74HC574	IC55, IC57	
62256 / 70ns	IC61	***70ns or faster***
27C256 / 80ns	IC62	***Program with file "charset.bin"***
1 K	R64	
100nF ceramic capacitor	C55, C57 - C63	
10 pin header	J5	
14 pin header	J8	
16 pin header	J1, J6	
20 pin header	J3, J4	

3.6.5 Testing

Please check the jumper settings.
A wrong setting can damage the Graphic Unit!

3.6.5.1 Set Jumper JP1 to the normal position 2-3.

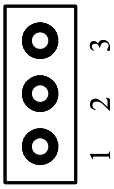
Check if Jumper JP1 (on the Pixel Control Board) is still in the normal position 2-3:



JP1

3.6.5.2 Remove the Jumper JP2.

Remove Jumper JP2 (on the Color Memory Board) from the jumper block:



JP2

3.6.5.3 Set Jumper JP3 to the normal position 1-2.

Set Jumper JP2 (on the Address Counter Board) to the normal position 1-2:

JP3

3.6.5.4 Put all boards together

Please stack all boards like shown on the picture. The Graphic Memory Board is inserted between the Address Counter Board and the Color Memory Board:

Timing Generator

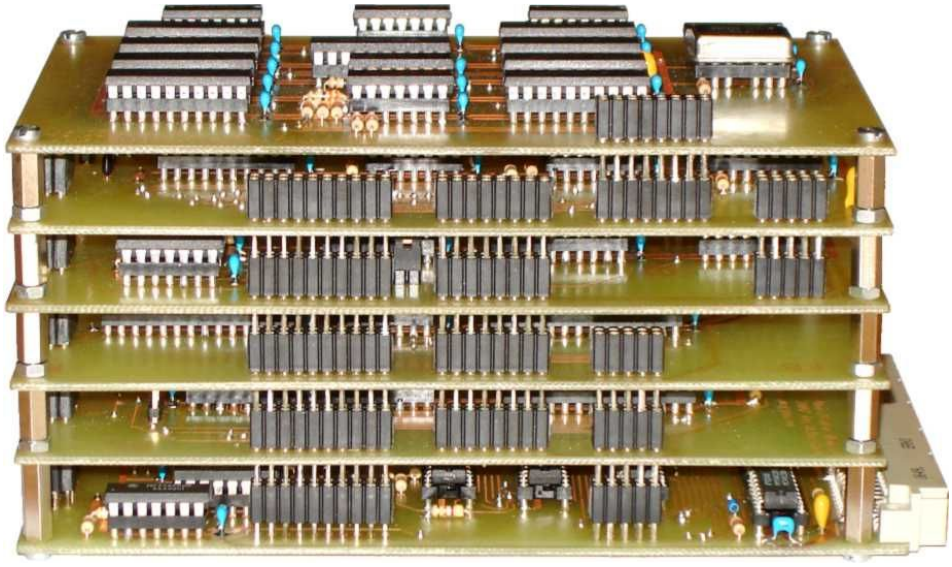
Pixel Control

Address Counter

Graphic Memory

Color Memory

Interface Board



4 Overall Part List

4.1 Detailed list of required parts

3 x	74AC00
1 x	74HC00
2 x	74AC04
1 x	74AC08
3 x	74HC08
3 x	74AC32
2 x	74HC32
3 x	74AC74
5 x	74HC74
3 x	74HC138
1 x	74HC139
2 x	74AC161
3 x	74HC161
1 x	74HC244
1 x	74HC245
1 x	74HC257
1 x	74HC393
1 x	74AC541
5 x	74HC541
6 x	74AC574
11 x	74HC574
4 x	74HC590
9 x	74HC688
2 x	62256 / 70ns
1 x	27C256 / 80ns
5 x	BAT 41
1 x	Oscillator 25.175 MHz
24 x	33 Ohm
3 x	47 Ohm
3 x	270 Ohm
3 x	390 Ohm
18 x	470 Ohm
3 x	560 Ohm
3 x	680 Ohm
10 x	1 K
4 x	10 K
5 x	47 K
1 x	SIL 4 x 470 Ohm
1 x	SIL 6 x 2K2
3 x	SIL 8 x 10 kOhm
2 x	SIL 8 x 47K
76 x	100nF ceramic capacitor
6 x	10µF / 16V, tantalum
3 x	3 pin jumper block
1 x	15 pin fine-pitch Sub-D VGA connector, female
1 x	DIN 41612 Connector
7 x	10 pin header
5 x	14 pin header
10 x	16 pin header
8 x	20 pin header

5 Solution: Use only 74HCxxx

5.1 Abstract

For some people the faster 74ACxxx parts are hard to obtain. But in some cases the usual 74HCxxx parts are too slow, so there is only one solution when you must use the slower 74HCxxx circuits: The clock frequency must be reduced.
But reducing the clock frequency has two disadvantages: First, a pixel is doubled in its horizontal size, and the high resolution graphic mode is no more available.
But for the first tests its still useful to run the Graphic Unit with a lower dot clock.

Warning!
When you do not use the recommended 74ACxxx parts, the test-pattern can look different!

5.2 Method 1

Use only the low resolution character sets. Unfortunately the MyCPU boots with the high resolution character set, so you may be unable to read the screen because of graphic errors. To solve this problem, edit the init-file (8:/init) and add the line “charset 5” to the end of the file. Make sure you have the charset-tool installed on your local drive: “8:/bin/charset”.

5.3 Method 2

Hard-wire the Graphic Unit to only use the low resolution character sets. The only thing you need to do is to add a 390 Ohm resistor between the pins 8 and 12 of the IC 67. See the figure below for details.
Note: This modification requires at least the MyCPU Kernel version 2.10.

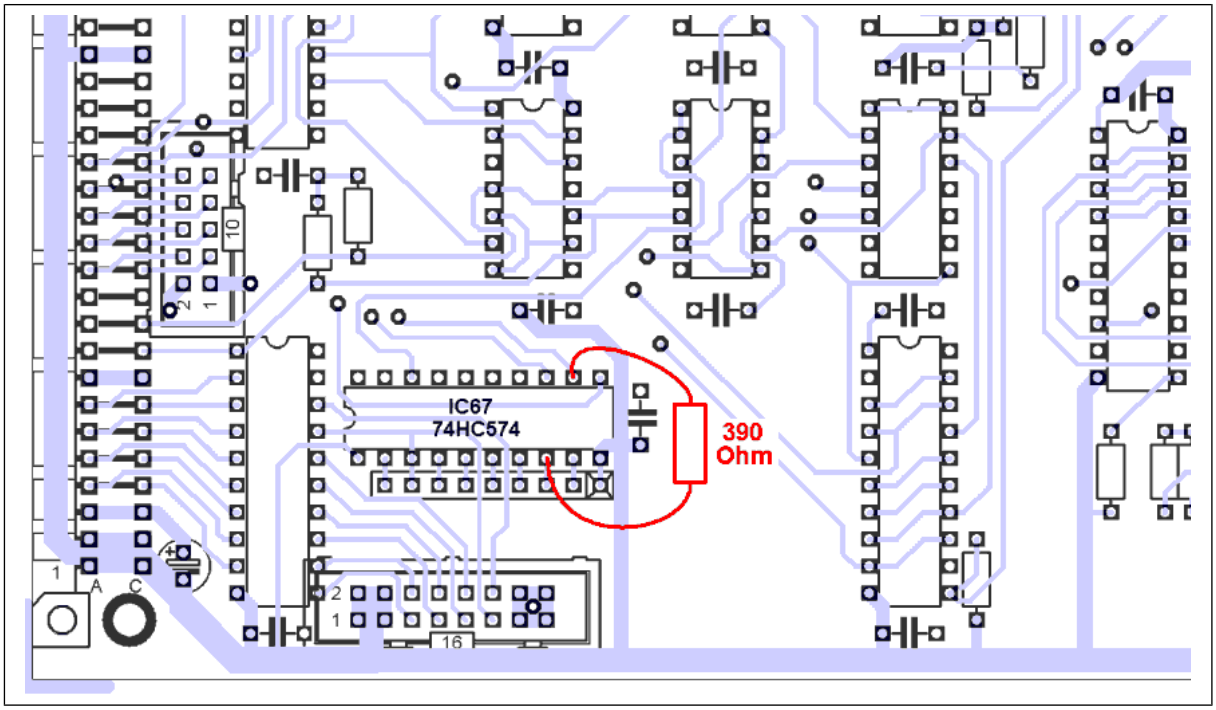


Fig. 31: Additional 390 Ohm resistor on the Interface Board

6 Registers

6.1 Overview

Address	Register Name	Rd/Wr	Function
2C00h-2CFFh	SCREEN_DATA	R/W	Access to graphic memory
2D00h-2DFFh	SCREEN_COLOR	R/W	Access to color memory
2E00h	REG_SCR_FIRSTLINE	W	Sets the first screen line
2E80h	REG_SCR_RAMBLOCK	W	Sets the memory bank for 2C00-2DFF
2F00h	REG_SCR_GFXCOLOR	W	Sets graphic fg/bg-color
2F80h	REG_SCR_FLAGS	W	Control Flags

6.2 SCREEN_DATA

This is a window with a size of 256 byte into the graphic memory. The window-position (or page) is set through the register REG_SCR_RAMBLOCK. 128 pages can be selected, thus the hole graphic memory can be accessed: 256 byte x 128 pages = 32 kb.
The meaning of the screen data depends on the graphic mode. In text mode, screen data is the ASCII text that shall appear on the screen. In graphic mode, the screen data represents a serial pixel stream where the MSB is printed first (appears at the left side on the screen).
Another important thing to know for the text mode is, that two text lines fit into one page, that means the first text line is stored from 2C00h to 2C4Fh, and the second one from 2C80h to 2CCFh. The register REG_SCR_RAMBLOCK selects the line pair that shall be modified.

Address: 2C00h - 2CFFh (read / write)

MSB(D7)							LSB(D0)
PIX7	PIX6	PIX5	PIX4	PIX3	PIX2	PIX1	PIX0

PIX7 – PIX0 When Graphic Mode is selected, this bits contains the pixel data that is displayed on the screen. A ‘1’-pixel gets the foreground color and a ‘0’-pixel gets the background color. The color is set through the bank SCREEN_COLOR or the register REG_SCR_GFXCOLOR. PIX7 appears at the left side on the screen and PIX0 on the right. In Text Mode PIX[7..0] is a pointer into the character set generator ROM. That means the value of PIX[7..0] can be an ASCII character.

6.3 SCREEN_COLOR

This is a window with a size of 256 byte into the color memory. The window-position (or page) is set through the register `REG_SCR_RAMBLOCK`. 128 pages can be selected, thus the hole color memory can be accessed: 256 byte x 128 pages = 32 kb.
In general, a byte in the color memory describes the foreground and the background color of a horizontal slice of 8 pixel. That means, 8 pixel share always the same foreground and background color. In Text Mode a character has always a width of 8 pixel, so this is no limitation. But in Graphic Mode the limitation is worse. Only in the low resolution 160x200 every pixel can have its own color (in fact, in this mode a dot is a package of 4x2 pixel).

Address: 2D00h – 2DFFh (read / write)

MSB(D7)							LSB(D0)
BG3	BG2	BG1	BG0	FG3	FG2	FG1	FG0

`FG0 – FG3` Foreground Color
`BG0 – BG3` Background Color

6.4 REG_SCR_FIRSTLINE

This register is only useful in the Text Mode. It describes the line in RAM that will appear on top of the screen. The text memory is organized as a ring buffer with either 25 or 50 text lines. When this register is set to zero, line 0 in the memory will become line 0 on the screen. But when this register is set to any other value, the assignment will become like this (example):
REG_SCR_FIRSTLINE is set to 20 and the 25 lines text mode is selected. The assignment will be: line 20 in memory gets line 0 on screen, line 21 in memory gets line 1 on screen, line 22 -> line 2, line 23 -> line 3, line 24 -> line 4, line 0 -> line5, line 1 -> line 6 and so on.
This register is used to implement some kind of “hardware assisted text scrolling”: It is not required to copy the memory when the screen scrolls one line up, but only the starting line must be changed.

Address: 2E00h (write only)

MSB(D7)							LSB(D0)
		FL5	FL4	FL3	FL2	FL1	FL0

`FL5 – FL0` Number of the first text line in memory. This value must be in the range 0..24 for the 25 lines text mode or in the range 0..49 for the 50 lines text mode.

6.5 REG_SCR_RAMBLOCK

Page register for graphic- and color-memory access. The memories are partitioned into 128 blocks of 256 bytes each. A block can be accessed through the window 2C00h-2CFFh (graphic memory) and 2D00h-2DFFh (color memory).

Address: 2E80h (write only)

MSB(D7)							LSB(D0)
	P6	P5	P4	P3	P2	P1	P0

`P6 – P0` Memory page number.

6.6 REG_SCR_GFXCOLOR

This register is used to set the two colors for the monochrome graphic mode.

Address: 2F00h (write only)

MSB(D7)							LSB(D0)
B3	B2	B1	B0	F3	F2	F1	F0

`F0 - F3` Foreground Color
`B0 - B3` Background Color

6.7 REG_SCR_FLAGS

This register is an accumulation of several flags used for controlling the Graphic Unit.

Address: 2F80h (write only)

MSB(D7)							LSB(D0)
$\overline{\text{GRA}}$	SPL	$\overline{\text{COL}}$	C40	R50	CS2	CS1	CS0

`CS0-CS2` Character-Set for Text Mode.

`R50` When this bit is set to 1 the text screen is 50 rows (=lines) high. Otherwise, when this bit is set to 0 (this is the default), the text screen is only 25 rows high.

`C40` When this bit is set to 1 the text screen is 40 columns wide. Otherwise, when this bit is set to 0 (this is the default), the text screen is 80 columns wide.

`COL` This bit enables the monochrome mode. When it is set to 1 the source for all colors is the register `REG_SCR_COLOR`, thus the screen will display a monochrome picture. If this bit is set to 0 (this is the default) then the color memory is used as source for all colors.

`SPL` Split screen enable. This bit takes only effect when bit `GRA` is set to 1. Set `SPL` to 1 to enable the split screen. In split screen mode the upper half of the screen operates in text mode and the lower half of the screen displays in graphic mode. The register `REG_SCR_FIRSTLINE` indicates the text line after which the display will be switch from text to graphic.

`GRA` Graphic Enable. Set this bit to 0 to enable the high resolution graphic display. Usually this bit is set to 1 to display the text mode.

7 Memory Layout

7.1 Text Memory

The following memory layout applies only when REG_SCR_FIRSTLINE is set to 0.

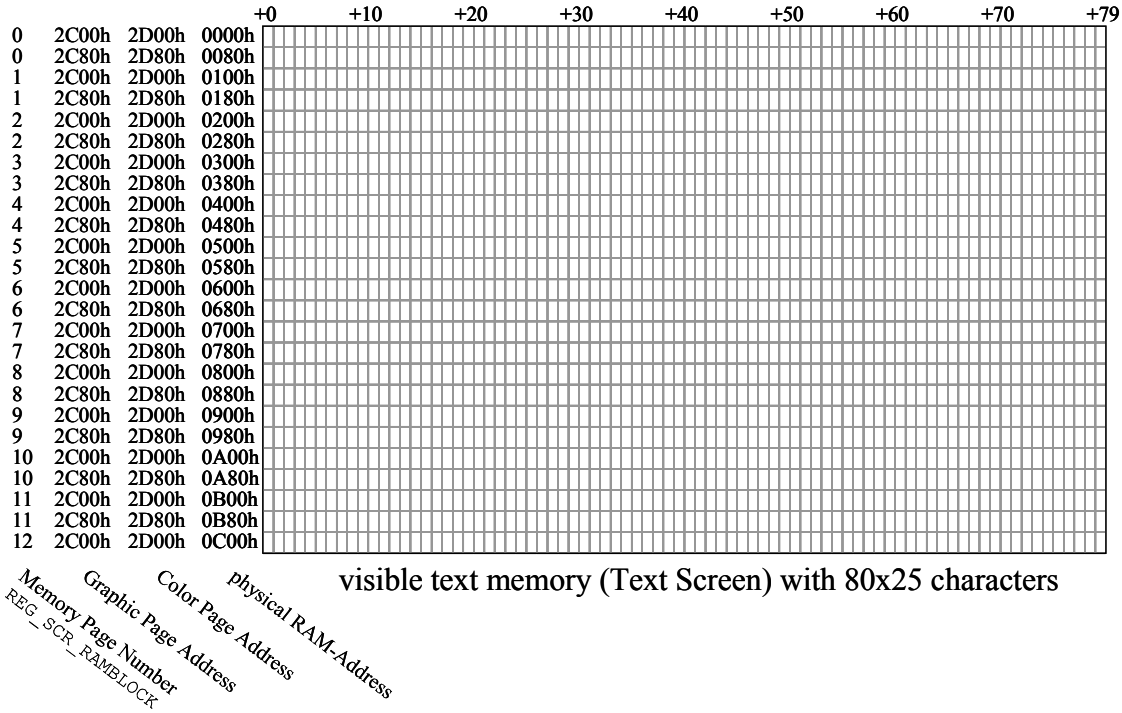


Fig. 32: Text Screen – Memory Layout

7.2 Graphic Memory

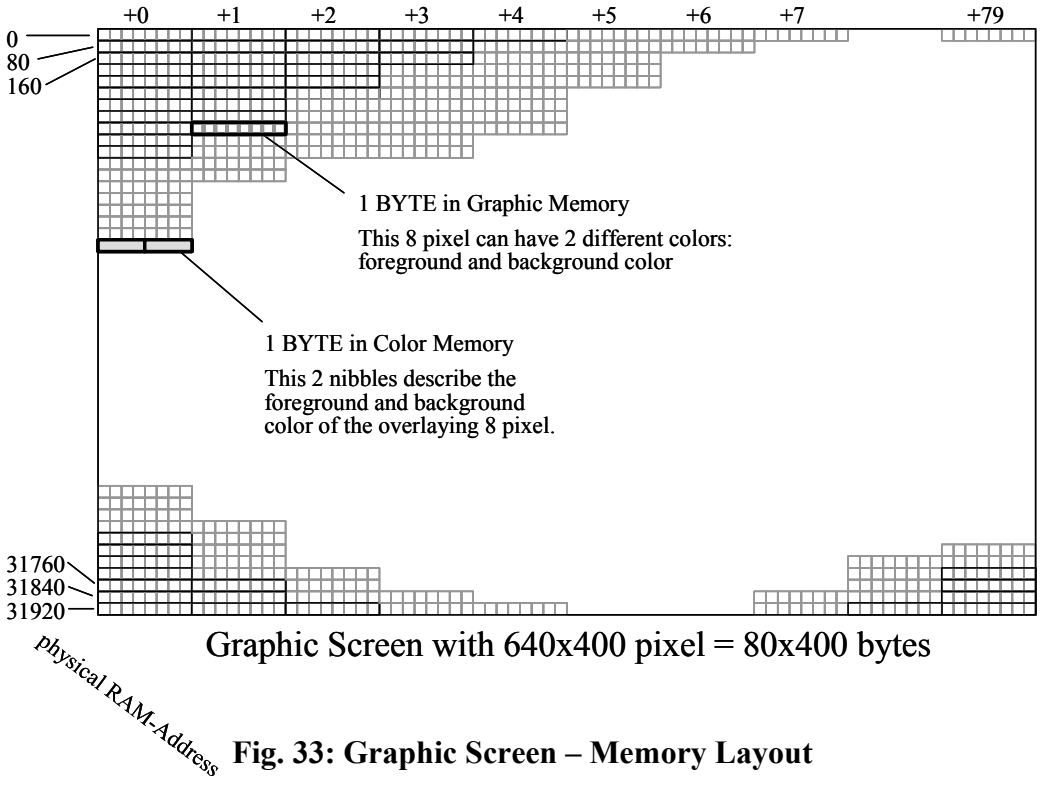


Fig. 33: Graphic Screen – Memory Layout

8 Graphic Commands

8.1 Basic Commands

The **Basic Interpreter** in the version 2 comes with a set of useful graphic extensions.

8.1.1 CHARSET

The charset - command selects a new character set.

Syntax:

```
charset cset
```

cset: A charset number between 0 .. 15

Examples:

charset 0 - Display boot screen charset (80x25)
charset 5 - Display high resolution charset (80x50)

8.1.2 GRAPHIC

The graphic - command is used to change the graphic mode.

Syntax:

```
graphic mode[,splitline[,backgroundcol[,foregroundcol]]]
```

mode: one of these modes:

0 = text mode
1 = 640x400x1
2 = 160x200x16
3 = 320x200x16
4 = 640x400x16

splitline: numbers of text lines at the top of the screen (usually 0)

backgroundcol, foregroundcol: colors (value in the range 0..15)

Examples:

graphic 0	-	Switch graphic off (return to text only mode)
graphic 1	-	Switch to monochrom graphic mode
graphic 2,15,0	-	Switch to 160x200 color graphic, split screen at line 15 (mixed text+graphic mode), set background color to black and clear the screen
graphic 4,,0,15	-	Switch to 640x400 color graphic (pure graphic mode), set background color to black and foreground to white and clear the screen

8.1.3 PIXEL

This command outputs a pixel on the graphic screen.

Syntax:

```
pixel x,y[,col]
```

x,y - Coordinates (maximum x,y depends on selected graphic mode)

col - Optional color of the pixel. If the color is not given, the last used color (or default color) will be used.

Examples:

pixel 40,50 - Outputs a pixel with the last used color
pixel 27,183,14 - Outputs a yellow pixel at position (27/183)

8.1.4 GPIX(x,y)

This function returns the color of a pixel.

Syntax:

```
a = gpix(x,y)
```

x,y - Coordinates (maximum x,y depends on selected graphic mode)

Examples:

print gpix(10,51) - prints the color code of the pixel at pos. (10/51)

8.1.5 LINE

The line - command draws a line on the graphic screen

Syntax:

```
line x1,y1,x2,y2[,x3,y3[,xn,yn[,...]]] [,color]
```

x1,y1 - start coordinates of the line
x2,y2 - target coordinates of the line
xn,yn - target coordinates of the next line fragment (this draws a line from point to point)
color - Optional color of the line. If the color is not given, the last used color (or default color) will be used.

Examples:

line 0,0,30,40 - draws a line from (0/0) to (30/40) with the last used color.
line 2,5,21,32,14 - draws a yellow line from (2/5) to (21/32).
line 0,0,4,5,10,3 - draws to lines. The first from (0/0) to (4/5) and the second from (4/5) to (10/3).

8.1.6 BOX

This command draws a box on the graphic screen.

Syntax:

```
box [f],x1,y1,x2,y2[,color]
```

f	- flag: 0 = don't fill the box, 1 = fill box with color
x1,y1	- coordinates of the upper left edge of the box
x2,y2	- coordinates of the lower right edge of the box
color	- Optional color of the box. If the color is not given, the last used color (or default color) will be used.

Examples:

box 1,0,0,30,40	- draws a filled box with upper left edge at (0/0) and lower right edge at (30/40) with the last used color.
box 0,5,5,12,20,14	- draws the outline of a box with upper left edge at (5/5) and lower right edge at (12/20). The color is yellow (14).

8.1.7 POLYGON

This command draws a polygon on the graphic screen.

Syntax:

```
polygon [f],x1,y1,x2,y2[,x3,y3[,xn,yn[,...]]] [,color]
```

f	- flag: 0 = don't fill the polygon, 1 = fill polygon with color
x1,y1	- first coordinate of the polygon
x2,y2	- second target coordinate of the polygon
xn,yn	- n'th coordinate of the polygon
color	- Optional color of the polygon. If the color is not given, the last used color (or default color) will be used.

Hint: The polygon-function can be used to draw a triangle !

Examples:

polygon 1,100,110,150,160,50,170,14	- Paints a filled, yellow triangle. The edges of the triangle are at (100/110), (150/160) and (50/170).

8.1.8 CIRCLE

This command draws a circle on graphic screen.

Syntax:

```
circle [f],x,y,r[,color]
```

f	- flag: 0 = don't fill the circle, 1 = fill circle with color
x,y	- center position of the circle
r	- radius of the circle
color	- Optional color of the polygon. If the color is not given, the last used color (or default color) will be used.

Examples:

circle 0,100,150,60	- Draws the outline of a circle with its center at (100/150). The radius is 60 and the color is the last used one.
circle 1,80,60,30,14	- Draws a filled, yellow circle with its center at (80/60) and a radius of 30.

8.1.9 STYLE

The style - command sets the line style that is used by the commands line, box, polygon and circle.

Syntax:

```
style [w][,color]
```

w	- Width of lines that are drawn by "polygon" and "circle". The line width must be in the range 1..3
color	- Sets the default color for all graphic commands.

Examples:

style 2	- sets the line width to 2 pixel
style ,4	- sets an other painting color (foreground color)
style 3,15	- sets line width to 3 pixel and color to yellow

8.1.10 TEXT

This command prints a text as bitmap graphic.

Syntax:

```
text x,y,[s],"text"[,[col][,backgr]]
```

x,y	- position of the top left edge of the text
s	- size of the text (usually 1, max. 31)
text	- the string to print
col	- text color (if this parameter is not set the foreground will be opaque)
backgr	- background color (if this parameter is not set the background will be opaque)

Examples:

text 20,30,,"Hello World"	- Outputs a text at position (20/30).
text 0,0,2,"foo",14,0	- Outputs a double-sized text at (0/0) with yellow(14) color and a black(0) backround.
text 0,5,,"bar",12	- Outputs a light-red(12) text at pos.(0/5) with an opaque backround.
text 6,5,,"foobar",,14	- Outputs an opaque text at position (6/5) with a yellow (14) backround.

8.2 Basic Example Program

```
10  graphic 2,0,0,15
20  c=9
30  for i=0 to 79 step 3
40  c=c+1 : if c>15 then c=9
50  line i,i,159-i,i,c
60  line 159-i,i,159-i,199-i,c
70  line 159-i,199-i,i,199-i,c
80  line i,199-i,i,i,c
90  next
100 get a$:if a$="" then 100
110 graphic 0
```

8.3 Graphic Demo Programs

Please download the latest MyCPU-Software-Package from www.mycpu.eu.
After installation you will find a new folder on your MyCPU. The folder `8:/gfx/` contains a set of demo programs and pictures that demonstrate the capabilities of the Graphic Unit.

8.4 Shell Commands

8.4.1 charset

The charset command is used to change the current character set and/or screen resolution.

Syntax:

```
charset cset
```

cset: A charset number between 0 .. 15

0	IBM style character set, 80x25
1	CBM style character set, 80x25
2	C64 charset 1, 80x25
3	C64 charset 2, 80x25
4	IBM style character set, 80x50
5	CBM style character set, 80x50
6	C64 charset 1, 80x50
7	C64 charset 2, 80x50
8	IBM style character set, 40x25
9	CBM style character set, 40x25
10	C64 charset 1, 40x25
11	C64 charset 2, 40x25
12	IBM style character set, 40x50
13	CBM style character set, 40x50
14	C64 charset 1, 40x50
15	C64 charset 2, 40x50

8.4.2 split

The split command splits the screen in a text part and a graphic part. The text part is always above the graphic part.

Syntax:

```
split lines
```

lines: The number of text lines that shall be displayed. The rest of the screen is graphic. If this parameter is set to 0 the hole screen displays text.

Hint:

The command "`split 0`" disables any graphic screen and returns to a pure text screen.

8.4.3 view

This is a bitmap viewing program. It accepts files with a size of 32000 bytes (monochrome graphic) or 64000 bytes (color graphic). You can generate your own bitmap files with the tool "bmp2mycpu" that you'll find in the MyCPU software distribution package.

Syntax:

```
view file [file2 [file3 [fileN]]]
```

The filenames can contain joker signs. All files that match the joker will be displayed.

9 Graphic Unit without MyCPU

It is possible to use the Graphic Unit with any other master processor than the MyCPU. For example, you can use the Graphic Unit in conjunction with a microcontroller. The bus-interface is rather simple: It has a Data Bus and an Address Bus, and two steering lines for reading and writing data. But there is one difficulty: In the MyCPU-design the Graphic Unit is the bus master, it stops the MyCPU when the MyCPU wants to read/write from/to the Graphic Unit while the Graphic Unit is still not ready for the access. For this case the Graphic Unit asserts the /HALT-signal on the Bus and halts the MyCPU. This is done by simply switching off the clock signal in the MyCPU.

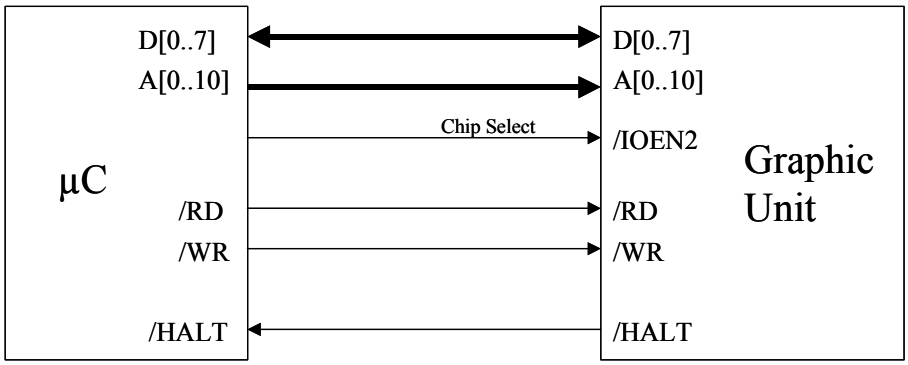


Fig. 34: Graphic Unit and Microcontroller

Note that the /HALT-signal is essential. If you can not handle the /HALT-signal correctly you can use the Graphic Unit anyway. If you can't provide the /HALT-signal make sure that a bus access takes at least 640ns (the /RD or /WR signal must be low for at least 640ns). When reading from the Graphic Unit fetch the data byte just before /RD goes high again.

In the MyCPU-design /IOEN2 goes low when the MyCPU accesses the address range 2800h-2FFFh.

10 Schematics

On the following pages you will find the schematics of the six GraphicUnit Boards.
The PCB layouts can be found in a separate PDF file (see file GU_Layouts.pdf).

10.1 List of all Schematics

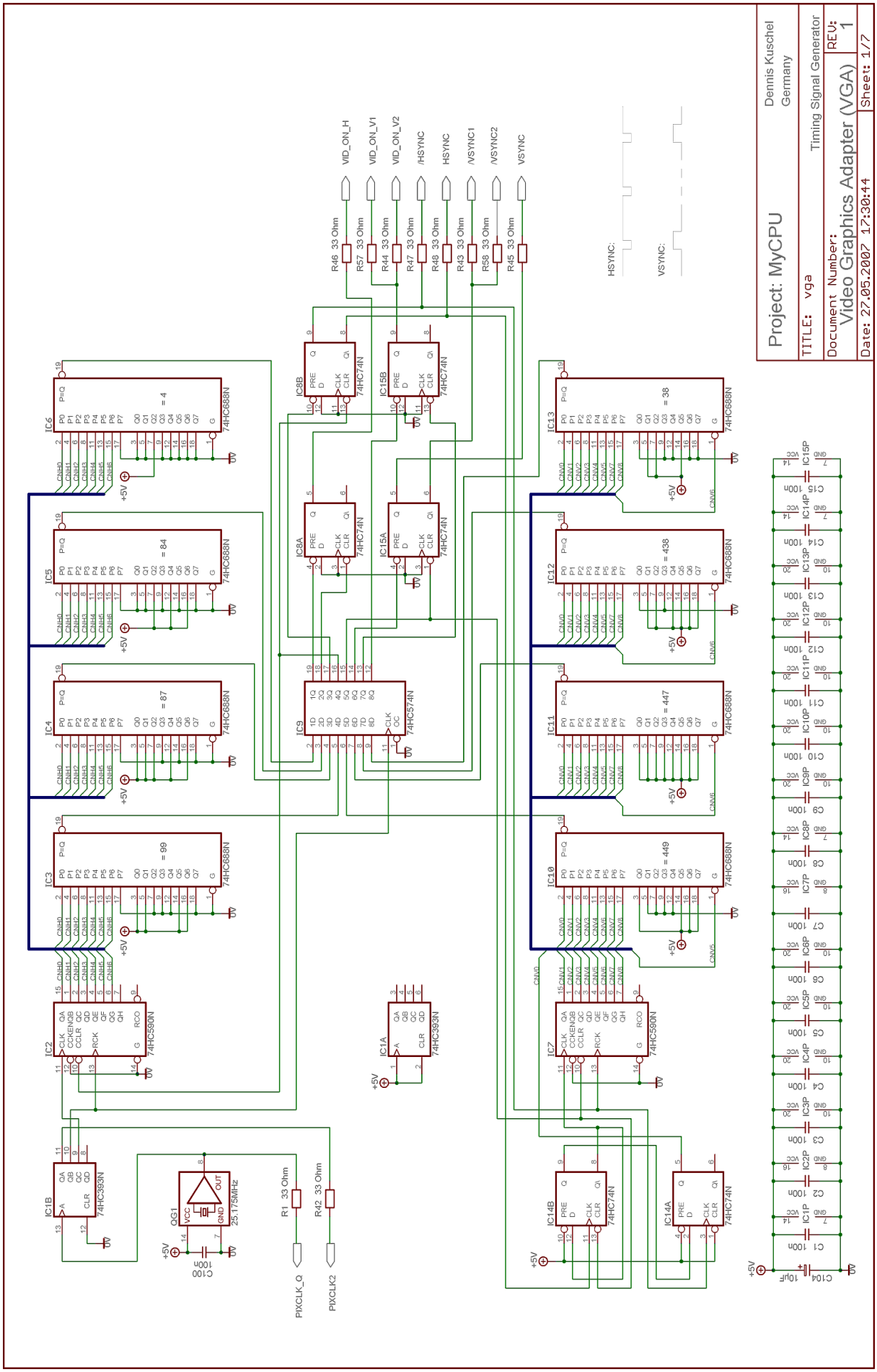


Fig. 35: Timing Generator

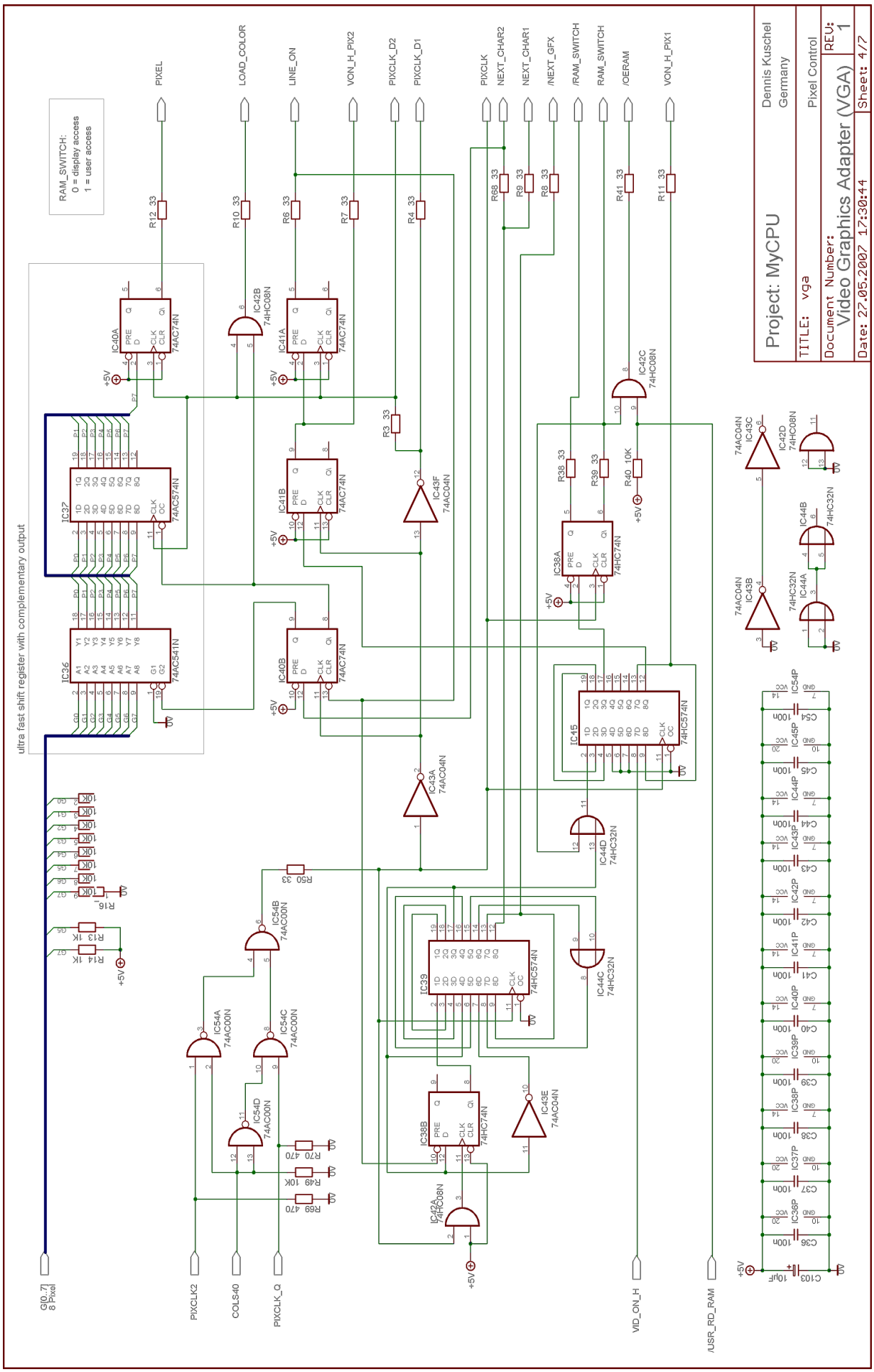


Fig. 36: Pixel Control

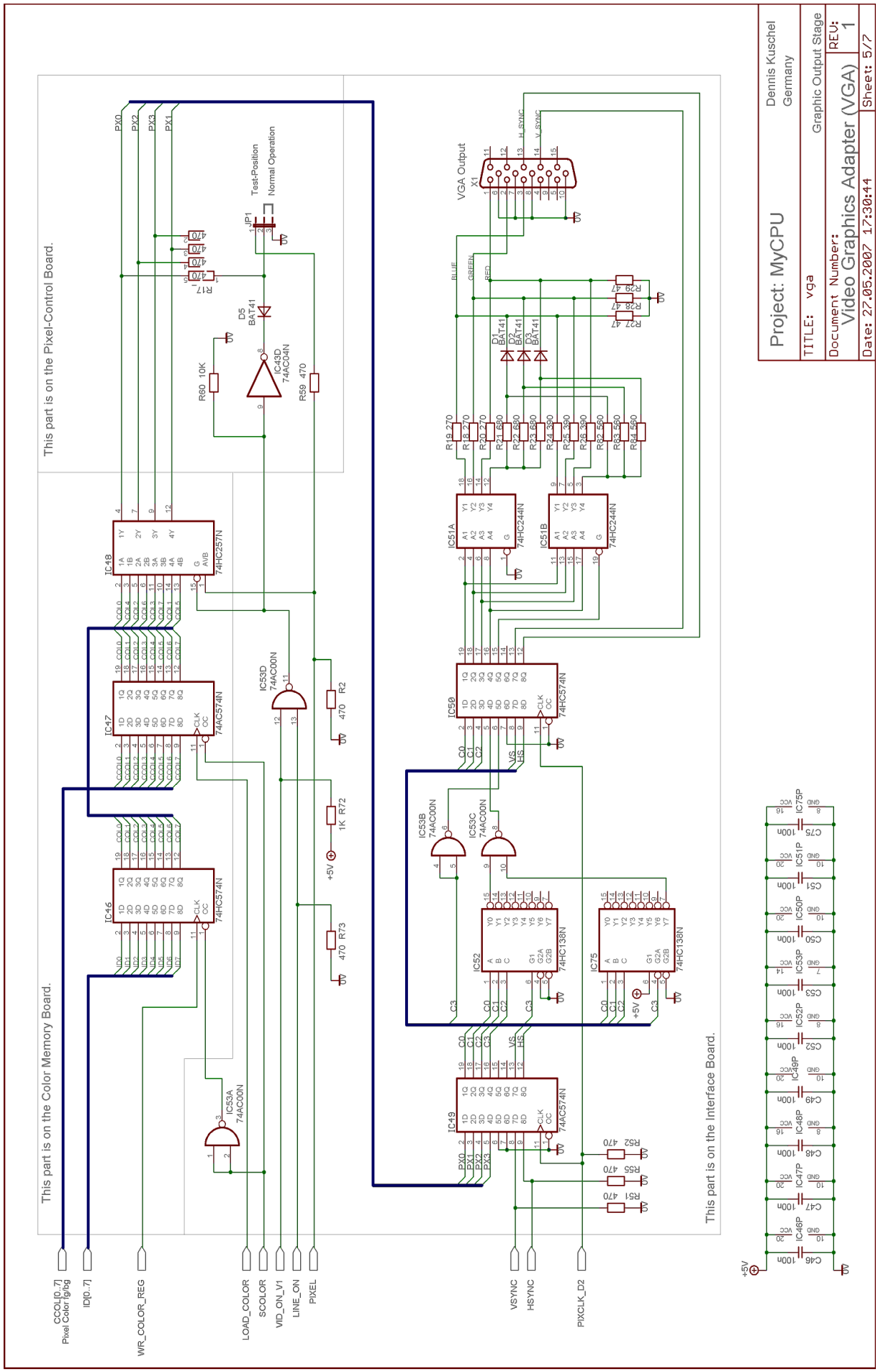


Fig. 37: Graphic Output Stage

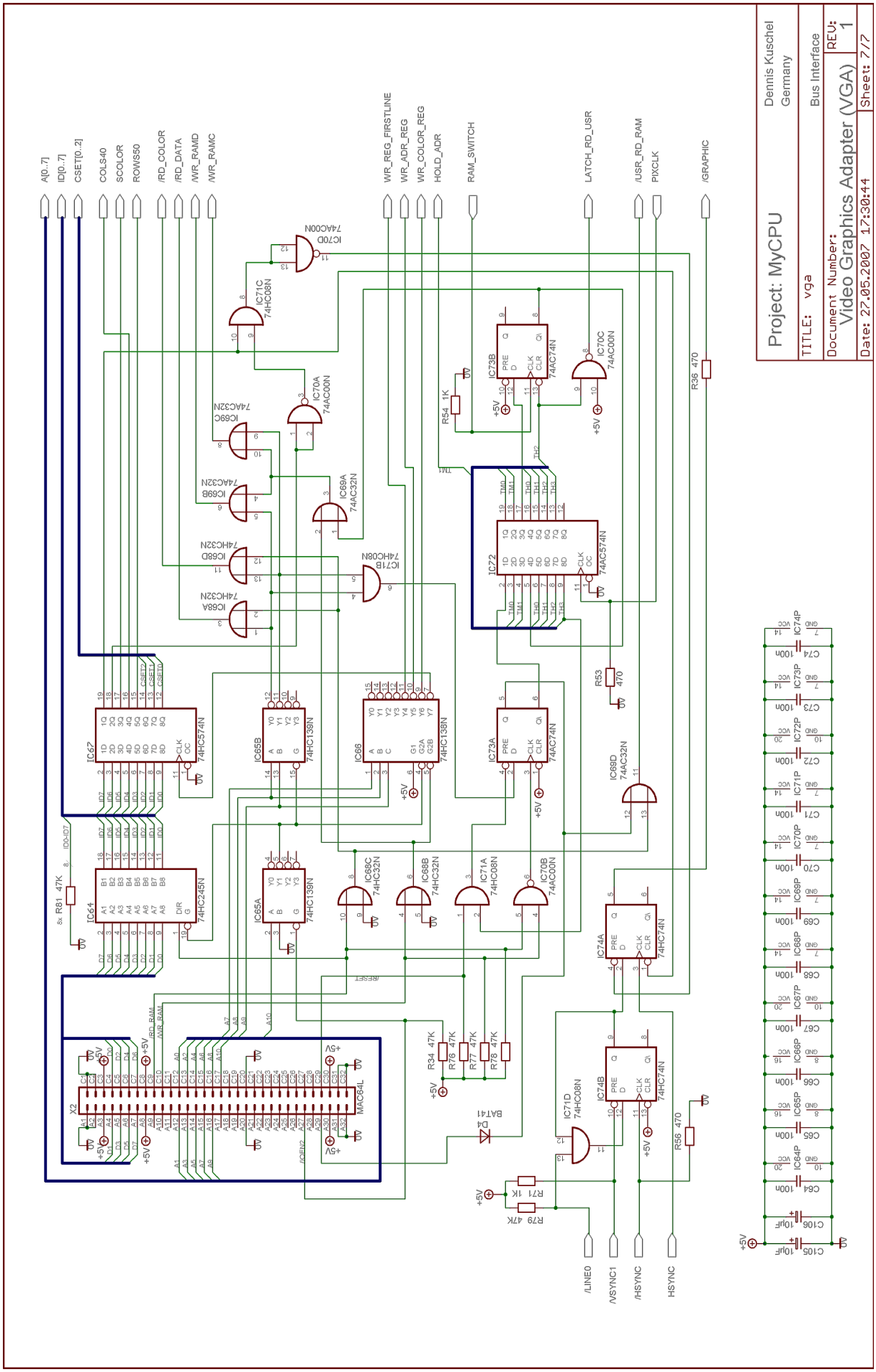


Fig. 38: Bus Interface

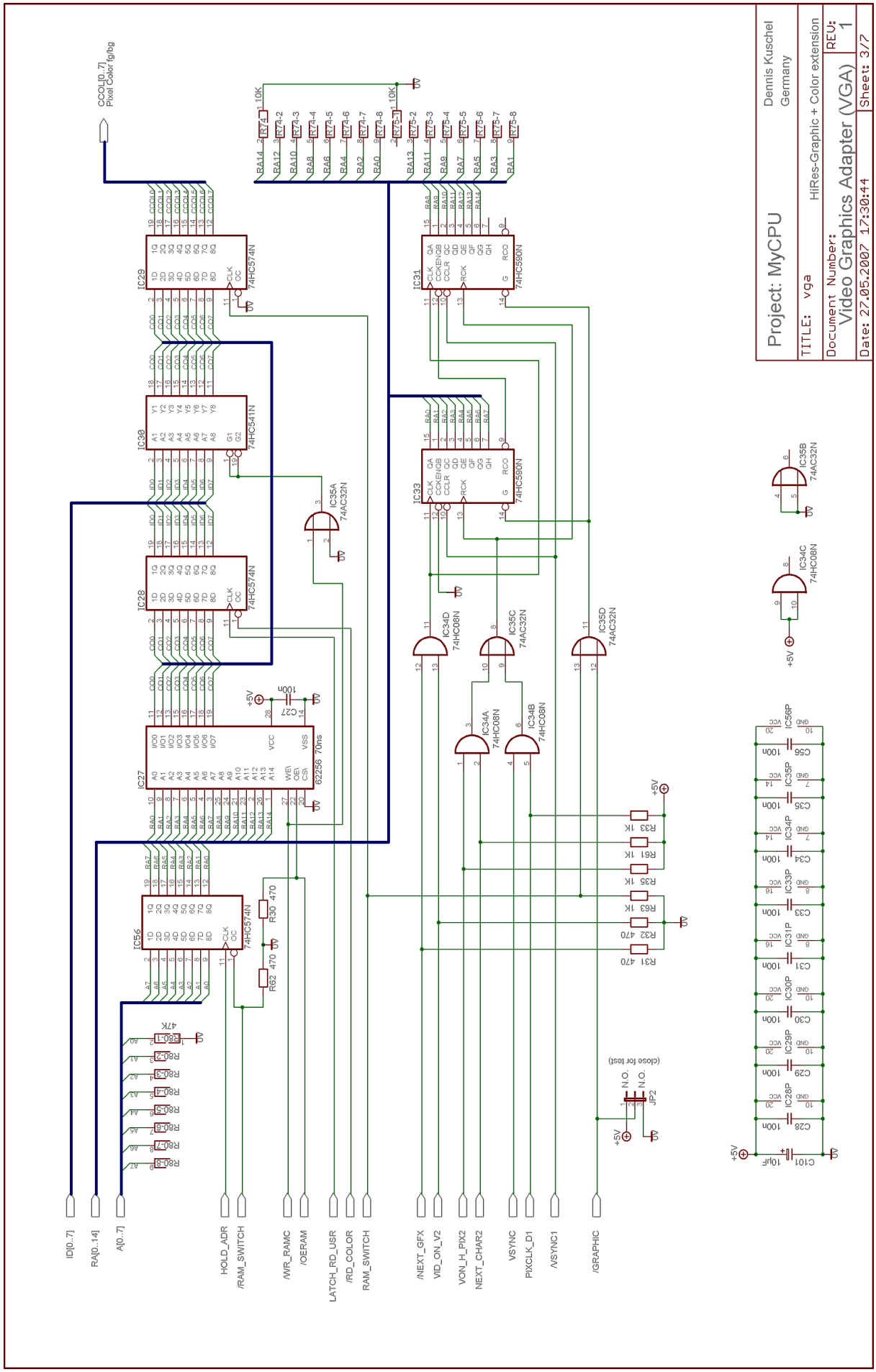


Fig. 39: Color Memory and Graphic Address Counter

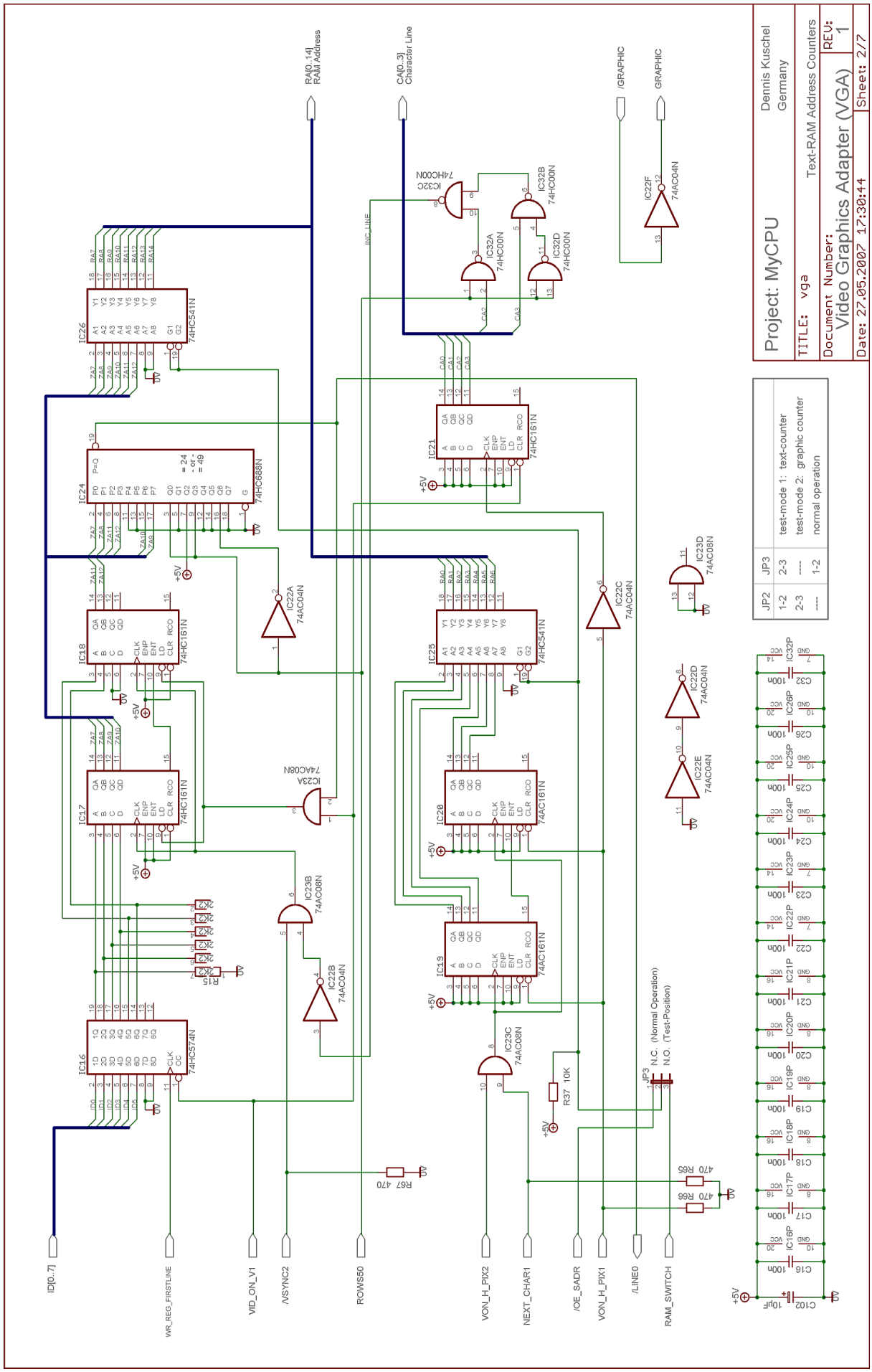


Fig. 40: Text Mode Address Counter

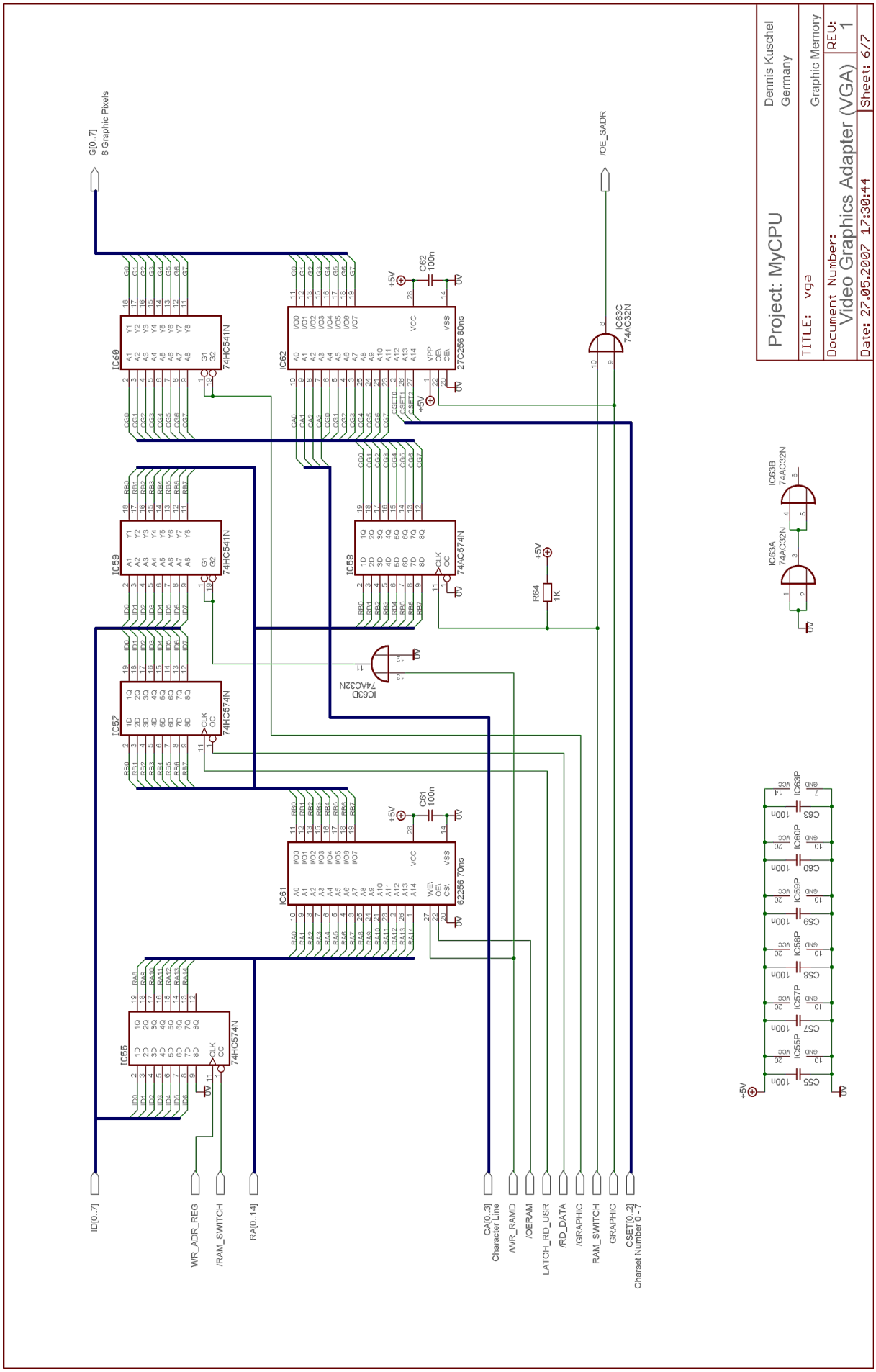


Fig. 41: Graphic Memory

11 Change Log

11.1 Changes in the Graphic Unit design

Date	Name	Chapter	Description
2007-06-23	D.Kuschel		Initial Release
2007-07-12	D.Kuschel	3.3	Value of Resistors R27, R28 and R29 changed to 47 Ohm
		4.1	Parts list updated
		10.1	Schematics updated
2008-11-05	D.Kuschel	3.1.4	Missing C100 added to parts list.
		3.2.4	Missing R60 added to part list.
		3.2.6.5	Text in Fig.14 corrected
		3.6.4	IC58 is a 74AC574, not a 74HC574.
2009-01-10	D.Kuschel	all	Converted to OpenOffice format
2015-07-23	D.Kuschel	3.1.4	Added a note about IC1
2015-07-23	D.Kuschel	3.2.5	Added a note about the used wire-wrap connectors