Versa64Cart

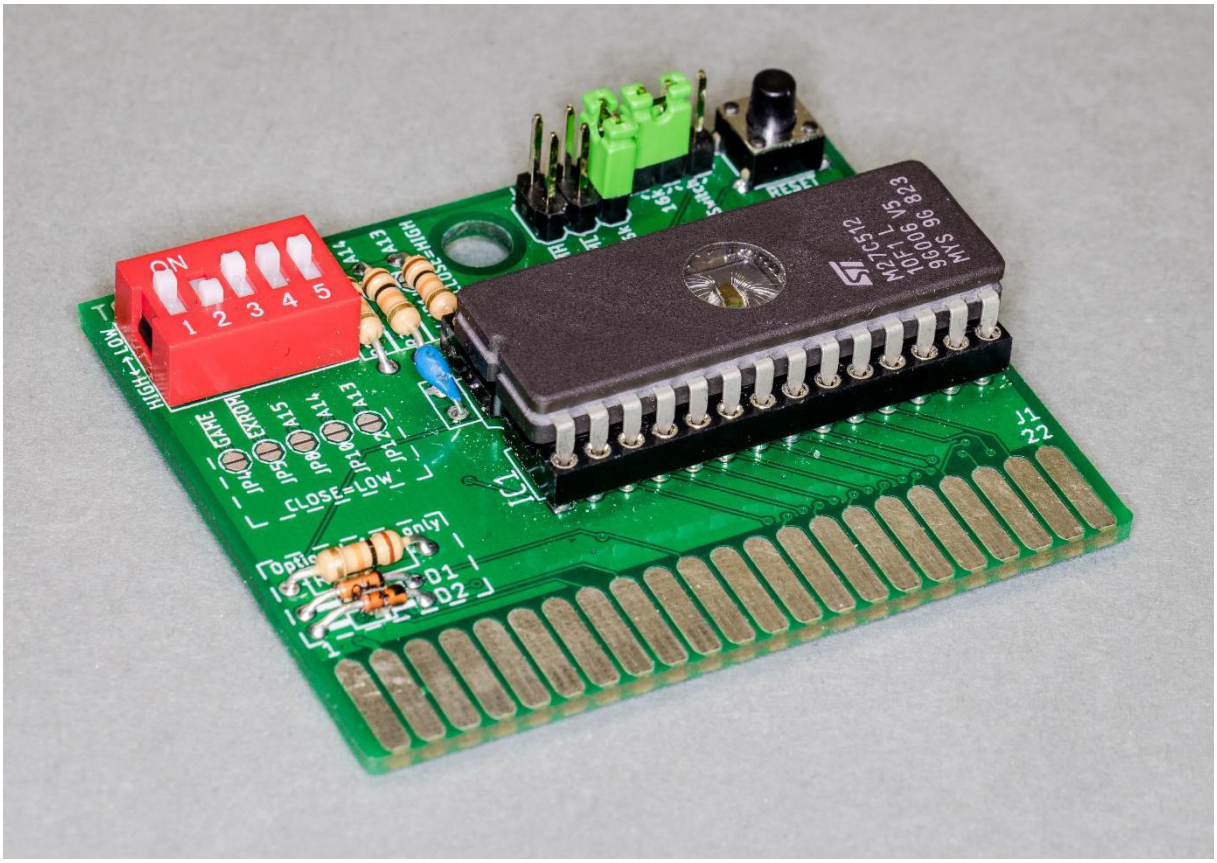
Project Documentation

BWACK's Versa64Cart

Project number: 121

Revision: 1.4

Date: 25.06.2019



Versa64Cart v1.4

Module Description

Content

1. Introduction

The Versa64Cart is a generic EPROM cartridge for the Commodore C64 and C128. It can be configured as 8k, 16k, 8k ultimax or 16k ultimax cartridge. Further on, the most sufficient address bits of the EPROM can be set by a DIP-switch which offers a manual bank switching for multi game/multi ROM cart option.

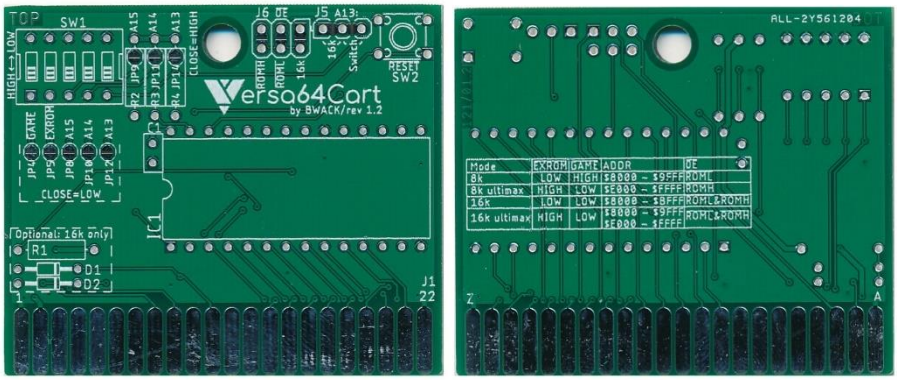


Figure 1: Both sides of the Versa64Cart rev 1.2

The Versa64Cart is not suitable to run a Kernal or software, which require bank switching by that software (e.g. games > 16k).

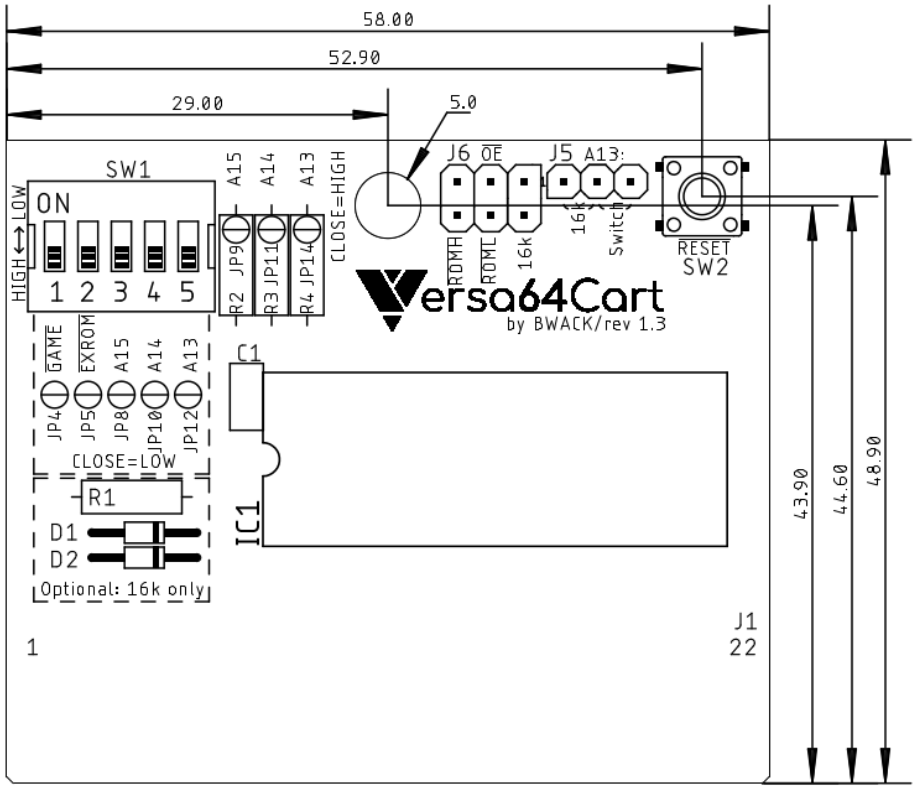


Figure 2: Versa64Cart - placement and dimensions

2. Configuration

2.1. Explanation

The extension cartridges signalize their configuration with two flags. Those are $\overline{\text{GAME}}$ and $\overline{\text{EXROM}}$. For addressing the EPROM, the C64 offers two chip-select signals for an 8kB address space, which are $\overline{\text{ROML}}$ and $\overline{\text{ROMH}}$. The chip-select signals are required to locate the EPROM content within the address space of the C64. It is possible to use either $\overline{\text{ROML}}$, $\overline{\text{ROMH}}$ or combining both to get a 16kB address space in total.

Mode	$\overline{\text{EXROM}}$	$\overline{\text{GAME}}$	$\overline{\text{OE}}$	Address space
8k	LOW	HIGH	$\overline{\text{ROML}}$	0x8000 – 0x9FFF
8k ultimax	HIGH	LOW	$\overline{\text{ROMH}}$	0xE000 – 0xFFFF
16k	LOW	LOW	$\overline{\text{ROML}}$ & $\overline{\text{ROMH}}$ = "16k"	0x8000 – 0xBFFF
16k ultimax	HIGH	LOW	$\overline{\text{ROML}}$ & $\overline{\text{ROMH}}$ = "16k"	0x8000 – 0x9FFF + 0xE000 – 0xFFFF

Table 1: Cartridge configuration

In the "ultimax" modes, the Kernal ROM (0xE000 – 0xFFFF) is replaced by the content of the EPROM of the Versa64Cart. The Reset vector is located at 0xFFFC and 0xFFFD. This is a guidepost for the microprocessor, which shows where to start the execution after the processor was powered up (or received a RESET pulse). This way, the software in the EPROM will completely take control.

In EXROM mode, the C64 Kernal (the one on the mainboard) will check the memory locations 0x8004-0x8008 for the "cartridge signature" CBM80 (PETSCII 0xC3, 0xC2, 0xCD, 0x38, 0x30). If this sequence is found, the execution will follow the cartridge cold start vector located at the first two memory addresses of the EPROM (0x8000 and 0x8001 in the C64's address space).

2.2. Addresses

This document mentions different sorts of addresses. All are expressed as hexadecimal numbers. This is indicated by a 0x… in front of the number (like it is common in the programming language C). In the Commodore world, a $... in front of the number indicates the hexadecimal format.

The addresses shown in Table 1 are absolute addresses in the memory space of the C64. The addresses shown in Table 2 and Table 3 are EPROM offset addresses. Both must no be confused. Depending on the cartridge mode, the selected memory bank is mapped to the addresses shown in Table 1 (by the PLA, the main logic chip of the C64).

2.3. Jumper & DIP-Switch Settings

The DIP-Switch SW1 configures the cartridge mode and the three most sufficient address bit of the EPROM (the "bank switching"). Annotation: ON position connects the signal to ground (=LOW).

The jumper J6 ($\overline{OE}$) sets the chip select signal for the EPROM. It can be $\overline{ROML}$, $\overline{ROMH}$ or "16k"(this is a diode logic AND of $\overline{ROML}$ and $\overline{ROMH}$).

The jumper J5 (A13) can be set to "Switch" or "16k". "Switch" means, that A13 is set by the DIP-Switch, "16k" means, that A13 is controlled by the $\overline{ROML}$ signal.

A 16k cartridge requires both jumpers (J5 and J6) set to "16k".

For the required settings refer to Table 1.

2.4. Bank Switching

The EPROM offers more memory (27C512 = 64kB), than it is required for a generic 8k or 16k cartridge. On account of this an EPROM can keep the content of several cartridges. The DIP-Switch SW1 (3, 4, 5 = A15, A14 and A13) selects, which of these cartridges is selected. In 16k mode, the setting of A13 is ignored.

DIP-Switch			Address Bits			EPROM Address
3	4	5	A15	A14	A13	(Offset)
ON	ON	ON	L	L	L	0x0000 – 0x1FFF
ON	ON	OFF	L	L	H	0x2000 – 0x3FFF
ON	OFF	ON	L	H	L	0x4000 – 0x5FFF
ON	OFF	OFF	L	H	H	0x6000 – 0x7FFF
OFF	ON	ON	H	L	L	0x8000 – 0x9FFF
OFF	ON	OFF	H	L	H	0xA000 – 0xBFFF
OFF	OFF	ON	H	H	L	0xC000 – 0xDFFF
OFF	OFF	OFF	H	H	H	0xE000 – 0xFFFF

Table 2: 8k cartridges memory banks

DIP-Switch			Address Bits			EPROM Address
3	4	5	A15	A14	A13	(Offset)
ON	ON	X	L	L	X	0x0000 – 0x3FFF
ON	OFF	X	L	H	X	0x4000 – 0x7FFF
OFF	ON	X	H	L	X	0x8000 – 0xBFFF
OFF	OFF	X	H	H	X	0xC000 – 0xFFFF

Table 3: 16k cartridges memory banks

The C64 address must not be confused with the EPROM Address offset.

2.5. Solder Bridges

Instead of setting jumpers and the DIP-Switch, solder bridges can be utilized to configure the Versa64Cart. This is an option, in case the configuration of the Versa64Cart is not prone to be changed. Solder bridges are used to "hard wire" the configuration.

Signal	HIGH	LOW
$\overline{\text{GAME}}$	JP4 = open	JP4 = closed
$\overline{\text{EXROM}}$	JP5 = open	JP5 = closed
A15	JP8 = open, (JP9 = closed)	JP8 = closed, (JP9 = open)
A14	JP10 = open, (JP11 = closed)	JP10 = closed, (JP11 = open)
A13	JP12 = open, (JP14 = closed)	JP12 = closed, (JP14 = open)

Table 4: Configuration with solder bridges

The jumpers in brackets ( ) (JP9, JP11 and J14) are only required, in case the pull-up resistors R2, R3 and R4 are not populated. **Never (!!!!) close both jumpers of one address signal. It cannot be LOW and HIGH at the same time, this will produce a SHORT CIRCUIT!!!!**

The footprints of the jumpers J5 and J6 are designed to be solder bridged, when not populated.

3. Assembly of the Versa64Cart

The Versa64Cart can work with a minimum of components when being configured with solder bridges. Just the EPROM and a 100nF capacitor is required.

- The reset switch is nice to have, but it is just an option.
- The DIP-switch can be replaced by configuring the solder bridges JP4, 5, 8, 10 and 12.
- The pull-up resistors (R2-4) can be omitted and JP9, 11, 14 can be utilized to set a HIGH level on A15 … A13 **(BE CAREFUL, see warning above!). If you are not sure, keep the pull up resistors.**
- J5 and J6 can be hard wired with solder bridges.
- R1, D1 and D2 are only required in 16k modes.
- The EPROM can be soldered, a socket is not necessary

Soldering the EPROM can be required if a shallow cartridge case is used. A socket might add too much height. In this case, it is advised to test the EPROM with a Versa64Cart with a socket before soldering.

4. Retrieving the binary and the settings from a CRT file

4.1. VICE cartconv

The CRT file format is created to emulate cartridges with VICE or devices like the Ultimate II+ etc. It should not be programmed into an EPROM; this will not work.

Instead, the binary has to be retrieved from the CRT file. The binary is a file that can be used for programming EPROMs. Also, it is possible to find out the required settings for $\overline{\text{GAME}}$ and $\overline{\text{EXROM}}$.

The utility that is used here is CARTCONV, which is a command line toot that comes with VICE (C64 emulator). For the examples in this chapter , CARTCONV and the

cartridge files are copied to the same directory. To start, enter *cmd* into the search box on the task bar of Windows (Press Windows-Key, write cmd, press ENTER).

First the CRT information is required. Enter the following command

```
cartconv -f mycartridge.crt
```

mycartridge.crt stands for the crt file, that you want to convert.

The second step is to generate a *.bin file from the *.crt. The *.bin file can be used to program EPROMs.

```
cartconv -i mycartridge.crt -o mycartridge.bin
```

4.2. Example I - deadtest.crt (Mode 8k ultimax)

In this first example we will look at the output of cartconv and use it to make a dead test cartridge step by step. The important information is colored red.

1. List the CRT information:

```
C:\cartconv>cartconv -f deadtest.CRT
CRT Version: 1.0
Name: C64DEADTESTREV718220
Hardware ID: 0 (Generic Cartridge)
Mode: exrom: 1 game: 0 (ultimax)

offset  sig  type  bank start size  chunklen
$000040 CHIP ROM   #000 $e000 $2000 $2010

total banks: 1 size: $002000
```

2. Check the hardware ID. It must be 0. Many (game or special) cartridges require special circuitry. The Final Cartridge and Super Zaxxon are examples of that and will not work with the Versa64Cart.
3. Check the Mode. From the mode we know how to configure the jumpers on the board. It shows that exrom=1, game=0 and the mode is ultimax.
4. Check the size. Here it is $2000, meaning 8k.

With the information above we see that

- $2000: we need an 27C64 8k EPROM. If you select a larger ROM, tie A13-A15 to GND. (SW1-3, SW1-4 and SW1-5 to ON) to ensure that the lowest part of the rom is addressed.
- exrom: 1 game: 0 (ultimax): set SW1-1 to ON and SW1-2 to OFF.

Note: A switch in the OFF position means HIGH or 1. Nicely confusing.

$\overline{\text{EXROM}}$ has to be 1 (HIGH) and $\overline{\text{GAME}}$ has to be 0 (LOW) ⇒ DIP Switch 1=ON, 2=OFF. The ultimax mode is mentioned here, too. The start address is $e000 (0xE000). Referring to Table 1, J6 has to be set to $\overline{\text{ROMH}}$. The size is $2000 (which is hexadecimal for 8k). Hence jumper J5 is set to "switch". It is assumed, that the binary is programmed to the

first 8k of the EPROM, so the address lines A15, A14 and A13 are L, L, L ⇒ the switches 3-5 are ON.

Item	cartconv	Setting
$\overline{\text{GAME}}$	0 (= LOW)	SW1-1 = ON
$\overline{\text{EXROM}}$	1 (= HIGH)	SW1-2 = OFF
size	$2000 (=8k cartridge)	A13 (J5) = "Switch"
start	$E000	$\overline{\text{OE}}$ (J6) = $\overline{\text{ROMH}}$
EPROM offset	$2000	A15…13 = LLL ⇒ SW1-3 = ON SW1-4 = ON SW1-5 = ON

Now the deadtest.bin file can be generated:

```
C:\cartconv>cartconv -i deadtest.CRT -o deadtest.bin
Input file : deadtest.CRT
Output file : deadtest.bin
Conversion from Generic Cartridge .crt to binary format successful.
```

4.3. Example II - diag04.crt (Mode 8k game)

```
C:\cartconv>cartconv -f diag04.crt
CRT Version: 1.0
Name: 586220PLUS_0.4
Hardware ID: 0 (Generic Cartridge)
Mode: exrom: 0 game: 1 (8k Game)
offset  sig  type  bank start size  chunklen
$000040 CHIP ROM   #000 $8000 $2000 $2010
total banks: 1 size: $002000
```

Here, the Hardware ID is 0 (Generic Cartridge) again. The Versa64Cart is suitable for running the software. Now, it is assumed, that the first 8k are already in use and the binary should be programmed to the second 8k slot of the EPROM.

Item	cartconv	Setting
$\overline{\text{GAME}}$	1 (= HIGH)	SW1-1 = OFF
$\overline{\text{EXROM}}$	0 (= LOW)	SW1-2 = ON
size	$2000 (=8k cartridge)	A13 (J5) = "Switch"
start	$8000	$\overline{\text{OE}}$ (J6) = $\overline{\text{ROML}}$
EPROM offset	$2000	A15…13 = LLH ⇒ SW1-3 = ON SW1-4 = ON SW1-5 = OFF

The binary conversion is exactly the same like in Example I.

4.4. Example III - neutron.crt (Mode 16k game)

```
C:\cartconv>cartconv -f neutron.crt
CRT Version: 1.0
Name: NEUTRON
Hardware ID: 0 (Generic Cartridge)
Mode: exrom: 0 game: 0 (16k Game)
offset  sig  type  bank start size  chunklen
$000040 CHIP ROM   #000 $8000 $4000 $4010
total banks: 1 size: $004000
```

Again, it is assumed, the binaries should be programmed to the first 16k of the EPROM.

Item	cartconv	Setting
$\overline{\text{GAME}}$	0 (= LOW)	SW1-1 = ON
$\overline{\text{EXROM}}$	0 (= LOW)	SW1-2 = ON
size	$4000 (=16k cartridge)	A13 (J5) = "16k" $\overline{\text{OE}}$ (J6) = "16k"
start	$8000	Not relevant in 16k mode
EPROM offset	$0000	A15…13 = LLL ⇒ SW1-3 = ON SW1-4 = ON SW1-5 = ON

The binary conversion is exactly the same like in Example I.

4.5. Example IV - non suitable formats

```
C:\cartconv>cartconv -f batman.crt
CRT Version: 1.0
Name: batman
Hardware ID: 5 (Ocean)
Mode: exrom: 0 game: 1 (8k Game)
Warning: game in crt image set incorrectly.
offset  sig  type  bank start size  chunklen
$000040 CHIP ROM   #000 $8000 $2000 $2010
$002050 CHIP ROM   #001 $8000 $2000 $2010
[…]
$01e130 CHIP ROM   #015 $8000 $2000 $2010
total banks: 16 size: $020000
```

Here, the hardware ID is ***not*** 0 (generic). The Versa64Cart is ***not*** suitable for this software. The size is $20000, which is 128kB, more than what fits into the memory space of the C64. This kind of software requires a cartridge with a certain automatic bank switching

circuit, in this case the Ocean cartridge type and a 27C010 (128kx8bit) EPROM is required.

5. EPROMs

Four different types/sizes of EPROMs can be used with the Versa64Cart, not all settings make sense with them. Their pin out is shown in Table 5.

The effect of the settings and the recommended configurations are shown in Table 6.

						27C64					
						27C128					
						27C256					
						27C512					
						SOCKET					
Vpp	Vpp	Vpp	A15	1	A15	VCC	28	VCC	VCC	VCC	VCC
A12	A12	A12	A12	2	A12	A14	27	A14	A14	/PGM	/PGM
A7	A7	A7	A7	3	A7	A13	26	A13	A13	A13	n.c.
A6	A6	A6	A6	4	A6	A8	25	A8	A8	A8	A8
A5	A5	A5	A5	5	A5	A9	24	A9	A9	A9	A9
A4	A4	A4	A4	6	A4	A11	23	A11	A11	A11	A11
A3	A3	A3	A3	7	A3	/OE	22	/G/Vpp	/G	/G	/G
A2	A2	A2	A2	8	A2	A10	21	A10	A10	A10	A10
A1	A1	A1	A1	9	A1	GND	20	/E	/E	/E	/E
A0	A0	A0	A0	10	A0	D7	19	D7	D7	D7	D7
D0	D0	D0	D0	11	D0	D6	18	D6	D6	D6	D6
D1	D1	D1	D1	12	D1	D5	17	D5	D5	D5	D5
D2	D2	D2	D2	13	D2	D4	16	D4	D4	D4	D4
GND	GND	GND	GND	14	GND	D3	15	D3	D3	D3	D3

Table 5: EPROM pin compatibility

EPROM	Size	A15	A14	A13	16k
27C512	64kx8	yes	yes	yes	yes
27C256	32kx8	HIGH	yes	yes	yes
27C128	16kx8	HIGH	HIGH	yes	yes
27C64	8kx8	HIGH	HIGH	HIGH	no

Table 6: Settings per EPROM type

In case Vpp is located at a dedicated pin (pin 1), A15 has no effect anymore. A HIGH level is recommended (switch is off) . The /PGM Pin should be set HIGH. The n.c. (not connected) pin should be HIGH (with pull-up) or open. For an 8k EPROM, the 16k setting makes no sense.

6. Setting up an EPROM Image

6.1. Introduction

Since the size of the supported EPROMs usually exceed the 8k or 16k limit, it is possible to combine two or more *.bin files to a file that fills more than one memory banks in the EPROM. The possible bank addresses are shown in Table 2 and Table 3. In case you want to combine 8k and 16k cartridges, you can place the 16k *.bin files first and the 8k *.bin files behind it. This way you prevent the 16k *.bin files located to an bank

address, that is not shown in Table 3.It is of course possible to have an even number of 8k *.bin in front of the 16k *.bin.

The *.bin files can be merged to one EPROM image (also a *.bin file) in different ways:

- The software of the Programmer
- A hex editor
- Merging the files with Windows

6.2. Merging *.bin Files with the TL866 Programmer Software

The software mentioned above is capable of loading multiple files into the buffer, which can be saved and be used for programming the EPROM. After selecting the EPROM type, a buffer in the size of the EPROM is present in memory.

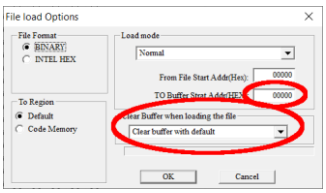


*Figure 3: Loading a *.bin file with default options*

The first *.bin file can be loaded with the default options (as shown in Figure 3). It will be placed in the lowest memory bank and the rest of the buffer is filled with 0xFF, which is handy, because this is the state of a not programmed byte.

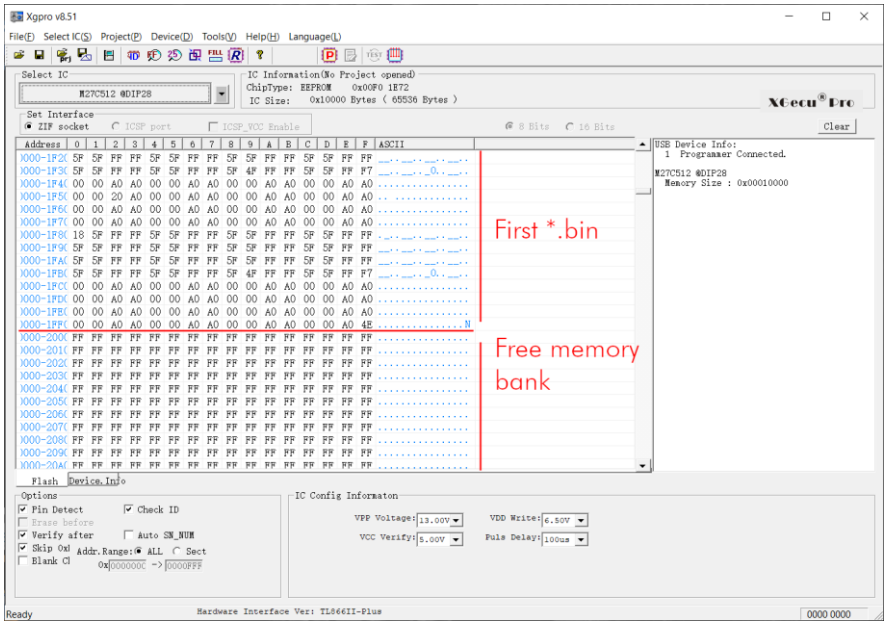


Figure 4: Program buffer

All following *.bin files have to be loaded with a proper address offset and "Clear Buffer…" disabled (Figure 5). The "TO Buffer… Addr (HEX)" address can be found in Table 2: 8k cartridges memory banks or Table 3: 16k cartridges memory banks.

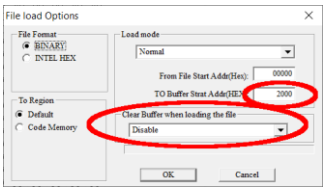


Figure 5: Options for the second 8k file

Note: It is even possible to load the content of an already programmed but not completely filled EPROM, find a free memory bank (which is filled with 0xFF), place a further *.bin there and program the EPROM without having to erase it prior to that.

It is strictly required to load the additional *.bin files to the memory banks in Table 2 and Table 3.

6.3. Merging *.bin Files with the HxD HEX Editor

A HEX editor, which allows to see the bytes as hexadecimal numbers is probably suitable to merge the *.bin files. The HxD editor (https://mh-nexus.de/en/downloads.php?product=HxD20) offers this functionality. Thus, it will be explained, how to do the task with this software.

First, it is important to check, whether the images have exactly the right size (which is 8k or 16k). Otherwise the merged images will not be in the right position in the begin of a memory bank. This will cause a malfunction, since the software will not start properly. The result might be a black screen or a boot screen, showing a number less than 38911 basic bytes free.

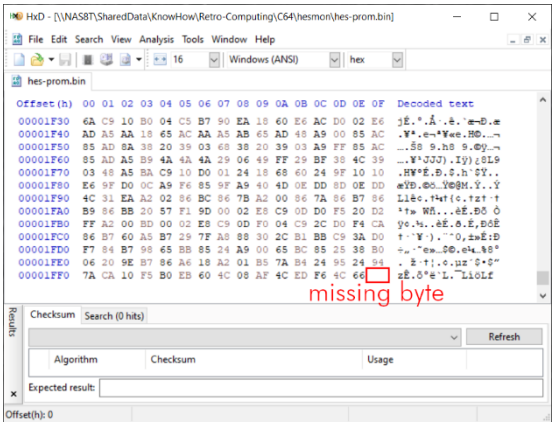


*Figure 6: *.bin file of a not suitable size*

An 8k *.bin has to fill the exact address space of 0x0000 – 0x1FFF, which is a byte count of 0x2000 (=8kbyte). The file shown in Figure 6 is missing one byte. This has to be filled up with preferably 0xFF (or any other value).

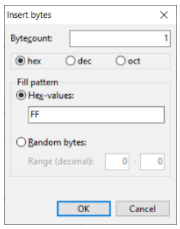


Figure 7: Inserting a byte with the HxD editor

Move the cursor at the position of the missing byte and select the operation “Insert bytes…” from the “Edit” menu. Here one byte is missing, so the byte count is “1”. The HEX-value is “FF”.

Some old EPROM images are 4k in size. This has to be filled up. So, again, position the cursor after the end of the data (0x1000, assuming, that the size is exactly 4k) and insert a hex byte count of 0x1000 (which is exactly 4k) to fill up to an 8k image.

Finally save the *.bin file. Fill up all files to 8k (0x2000 bytes) or 16k (0x4000 bytes).

After this operation, the files can be concatenated to get a proper image for the complete EPROM spanning over up to the maximum memory banks for the intended EPROM.

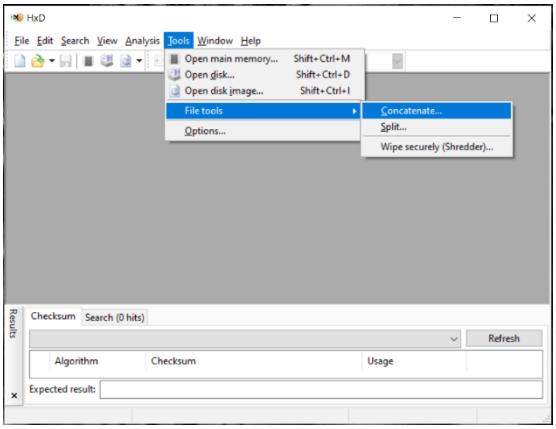


Figure 8: The Concatenate... tool

Select all files, that are intended to be programmed in one EPROM, specify the output directory and file name (…) and click ok.

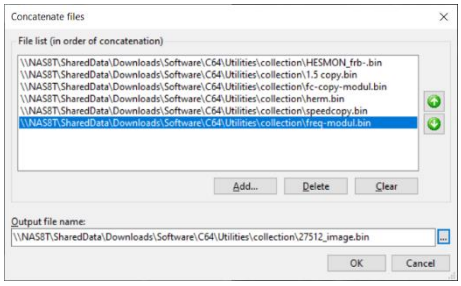


*Figure 9: Concatenating the *.bin files*

The output file can be used for programming an EPROM. The risk of the concatenate tool is, that a file does not have the proper size.

6.4. Merging *.bin Files with Windows

Combining several *.bin to one file can be accomplished with the copy instruction, that is part of Windows. To execute it, it is required to start the Windows Command Processor. This is accomplished by entering `cmd↵` in the search box like shown in Figure 10.

Figure 10: cmd being entered in the search box

First, you have to make sure, that your *.bin files are in one directory. The size is displayed by the `dir` command (Figure 11). For an 8k image the size has to be exactly 8192 bytes, for a 16k image, it is 16384 bytes. In case those values are not exactly the same, the image will not be in the right position and the cartridge will not work (except for the first image). Such a problem can be fixed with the HEX editor as described in the previous chapter.

To make sure, that 16k images are in a correct memory bank, it is suggested to put the 16image(s) on the first positions, followed by the 8k images. Alternatively, an even number of 8k images in front, will also work.

Merging of the *.bin files is accomplished with the `copy` command:

```
copy /b 16kimage1.bin + 16kimage2.bin + 8kimage1.bin epromimage.bin
```

Format of this instruction:

```
copy /b <input file 1> + <input file 2> + <input file 3>… <output file>
```

The execution of this command is shown in Figure 11. It should be paid attention to the output message of the copy command. All desired input files should be enumerated here.

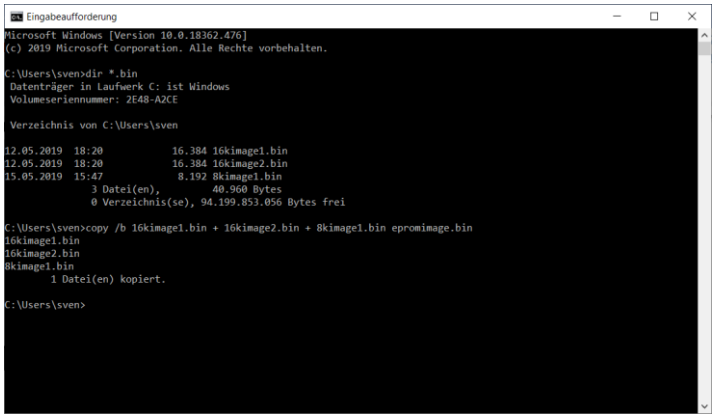


Figure 11: Executing the required commands on the Command Processor shell

7. Startup and Trouble shooting

Before you insert the Versa64Cart into the expansion port of the C64, you should make sure, that there are no fatal failures on it. In the worst case, it will produce a short circuit.

- Check the solder joints on the solder side
- Check the orientation of the socket. The notch is at the side, where the capacitor is

- In case you have used the solder bridges to configure the address, check according to the warning in chapter 2.5 or measure the resistance between pin 1 and 2 (that is GND and +5V) of the edge connector (without an EPROM inserted). It must not be less than about 3kΩ.
- After inserting the EPROM, check if the notch is at the same side like the notch of the socket. Check all pins are properly seated in the socket and not bent inwards or outwards.

After all these points are correct, nothing really bad can happen to your C64 anymore. If you get a black screen or a normal startup display with less than 38911 Basic Bytes free, you might have a configuration problem.

- Did you program the EPROM with a proper *.bin and not with the *.crt file? CRT files are not suitable to program an EPROM. Refer to chapter 4.
- Did you set the configuration bits correctly? ON means LOW
- Did you select the required $\overline{OE}$ (J6)?
- Did you jumper A13 (J5) as required?
- In case it is a 16k cartridge: Are R1, D1 and D2 placed?

8. Revision History

8.1. Rev. 1.1 ⇒ Rev. 1.2

- Schematic and layout were redrawn in Eagle (v 9.2)
- All solder bridges (jumpers) are located on the component side now (the footprint of the jumpers includes the solder bridge option)
- A13…A15 can be configured by solder bridges now
- No test points for A13…A15 anymore
- The jumpers are moved to the outer edge for easier access
- J5 for the configuration of A13 is a 3-pin jumper now
- A reset switch is added
- A configuration table is added on the solder side

8.2. Rev. 1.2 ⇒ Rev. 1.3

Some issues with cartridge cases were fixed.

- The diodes D1 & D2 and R1 were moved north by 4.93mm
- D1&D2 moved east by 4.6mm, R1 by 2.54mm
- IC1&C1 moved west by 1.59mm

8.3. Rev. 1.3 ⇒ Rev. 1.4

- The width of the contact pads of the expansion port was reduced to 1.5mm and the whole area is free of solder stop mask.

Placement:
This boards works with IC1 and C1 only.
All other components are optional.

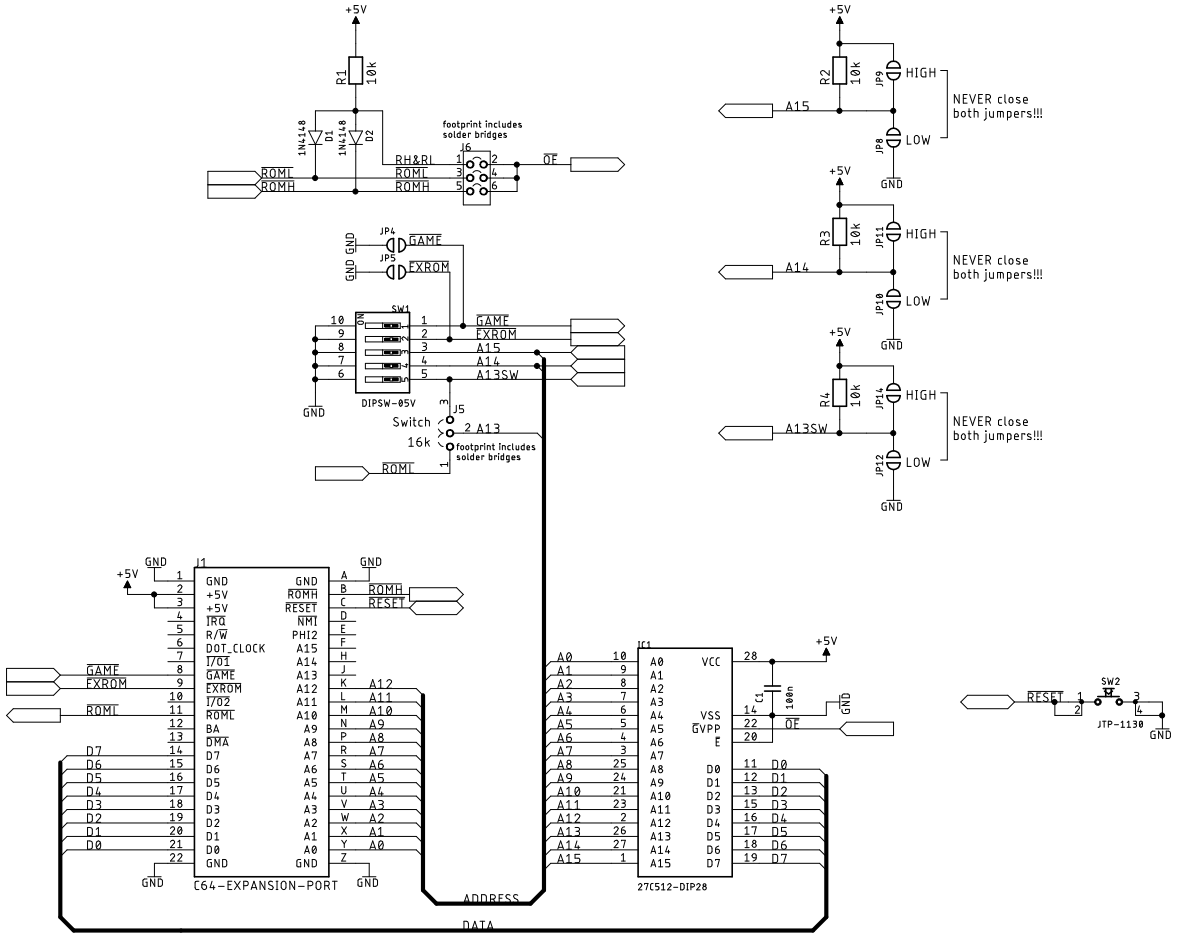
Jumpers (J5 and J6) and the DIP-switch
can be replaced with solder bridges.

D1, D2 and R1 are required for 16k.
($\overline{OE}$ = RH&RL).

R2..R4 are only required to set
A13..A15 = HIGH.

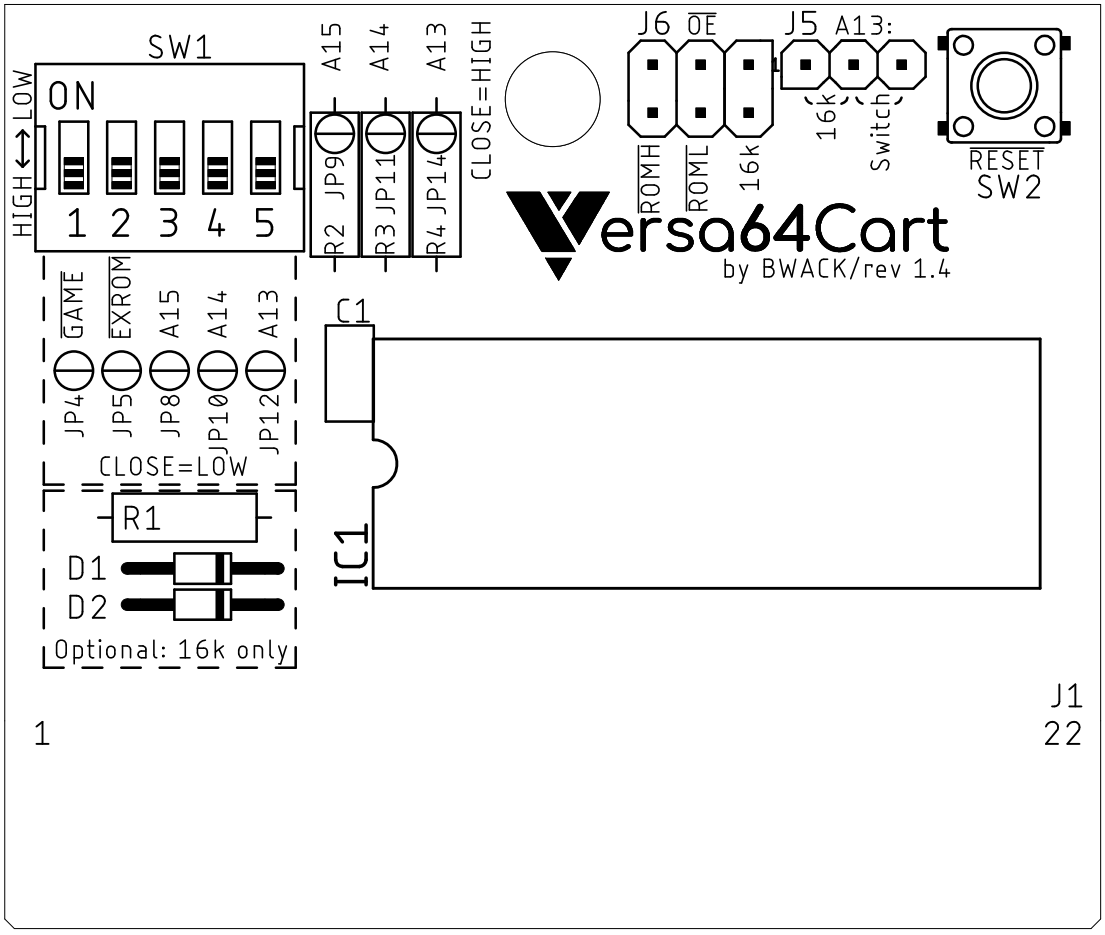
A13..A15 can be hardwired to HIGH
with a solder bridge (be careful!!!).

The RESET-switch is optional.



Title:	Versa64Cart	Doc.-No.:	121-1-01-01.4
		Draft.:	bwack
Date:	25.06.2019 10:52	Rev.: 1.4	Page 1/1
File:	Versa64Cart_v1_4		
			A3

BWACK	Doc.-No.:121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 10:52		Rev.: 1.4
placement component side		



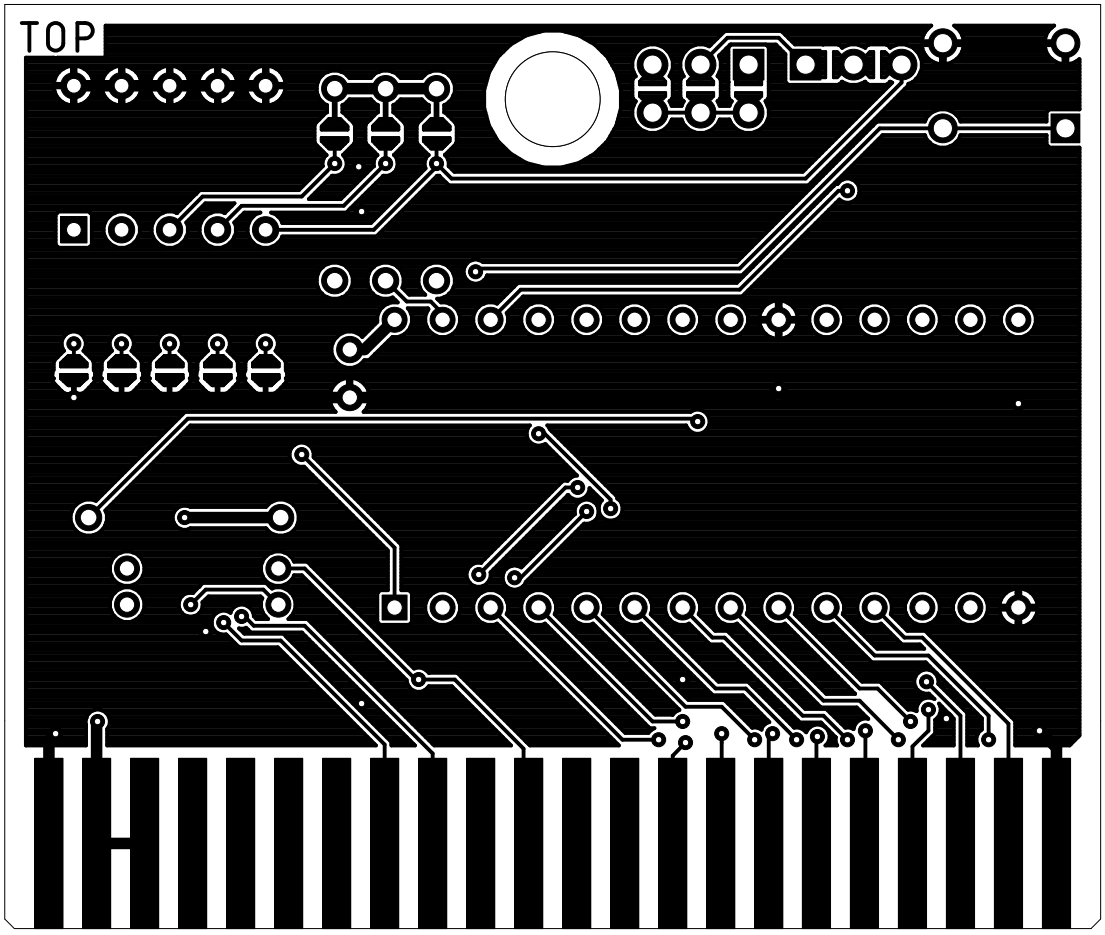
BWACK	Doc.-No.: 121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 10:52		Rev.: 1.4
placement solder side		

Mode	EXROM	GAME	ADDR	OE
8k	LOW	HIGH	$8000 – $9FFF	ROML
8k ultimax	HIGH	LOW	$E000 – $FFFF	ROMH
16k	LOW	LOW	$8000 – $BFFF	ROML&ROMH
16k ultimax	HIGH	LOW	$8000 – $9FFF $E000 – $FFFF	ROML&ROMH

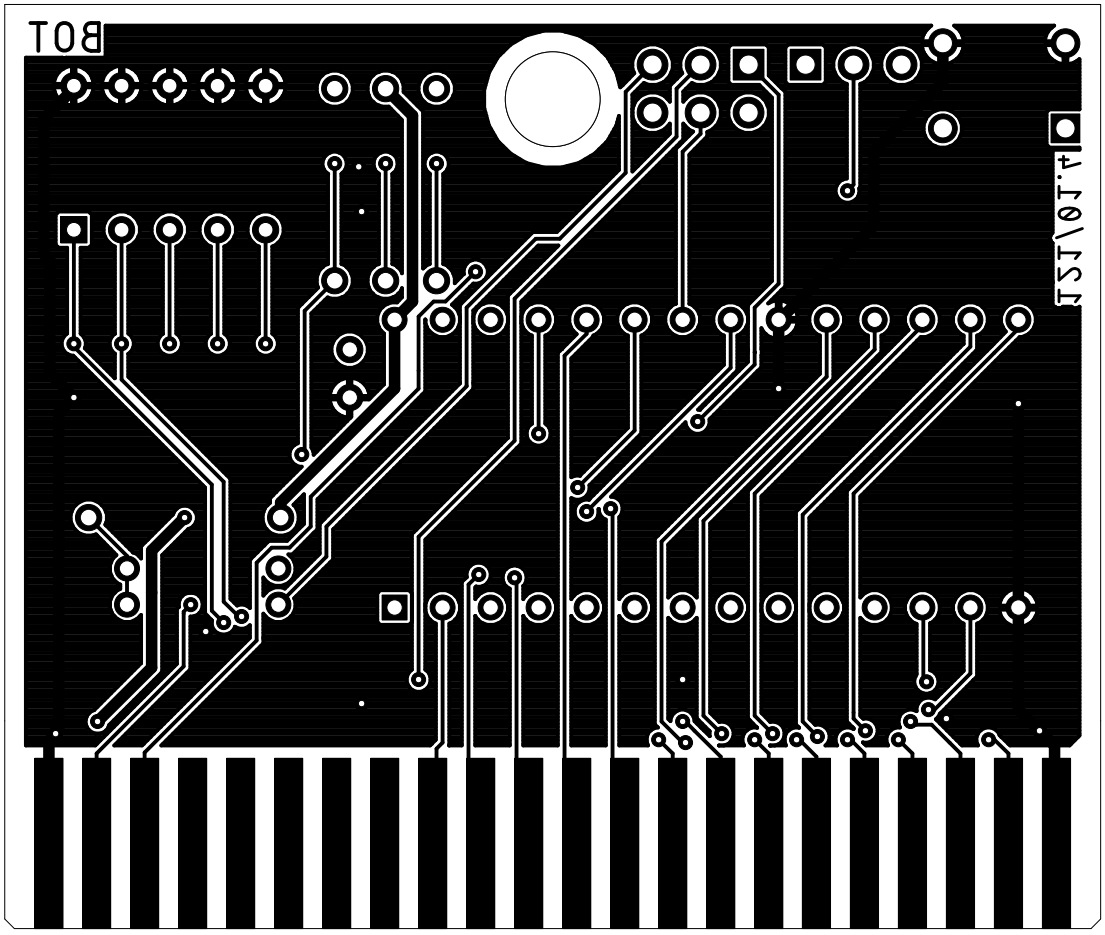
A

Z

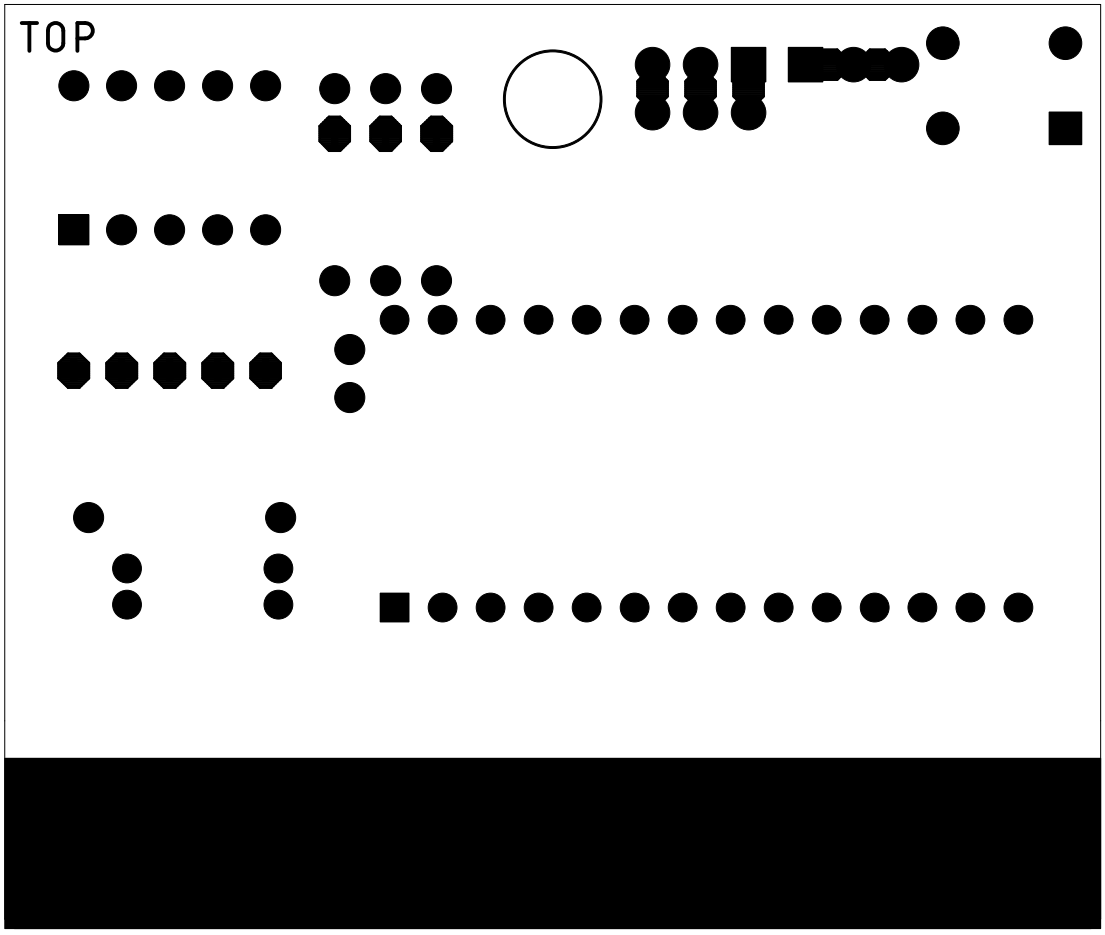
BWACK	Doc.-No.:121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 11:10		Rev.: 1.4
top		



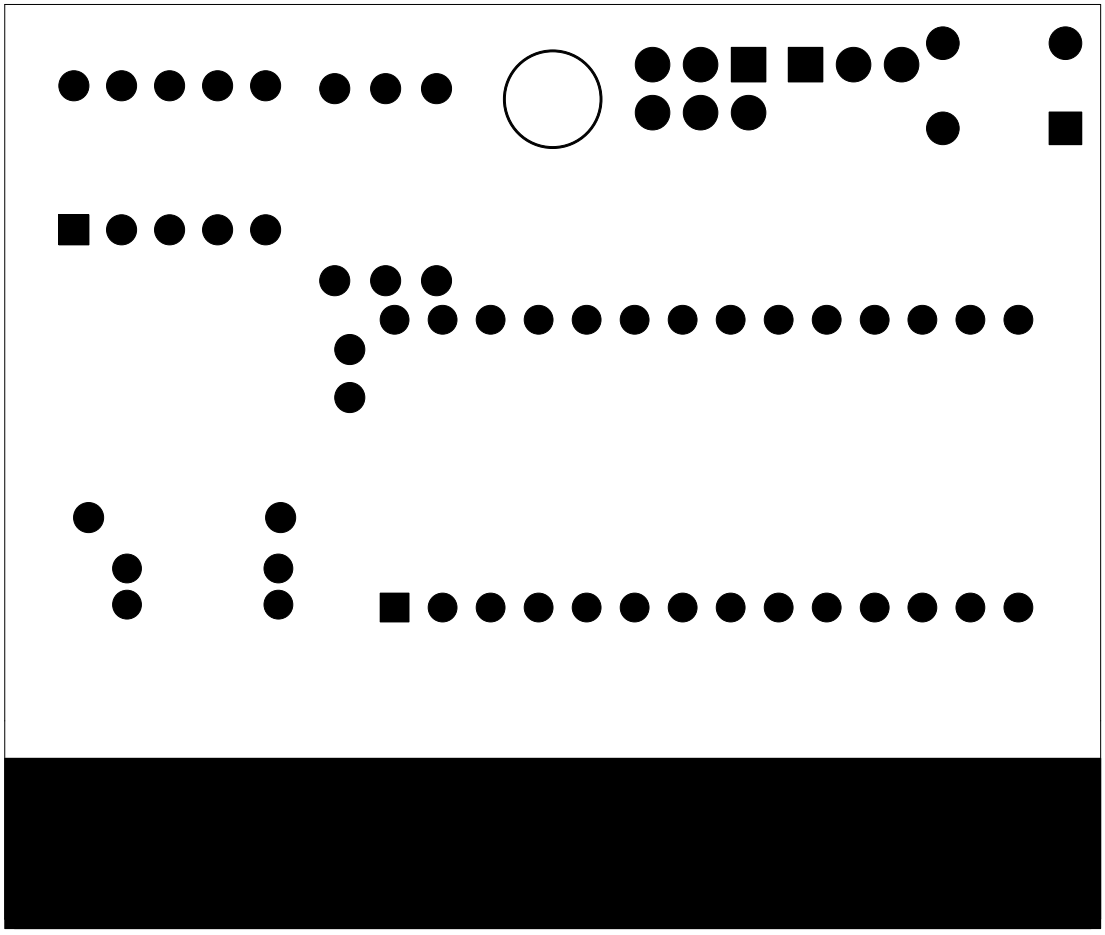
BWACK	Doc.-No.: 121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 11:10		Rev.: 1.4
bottom		



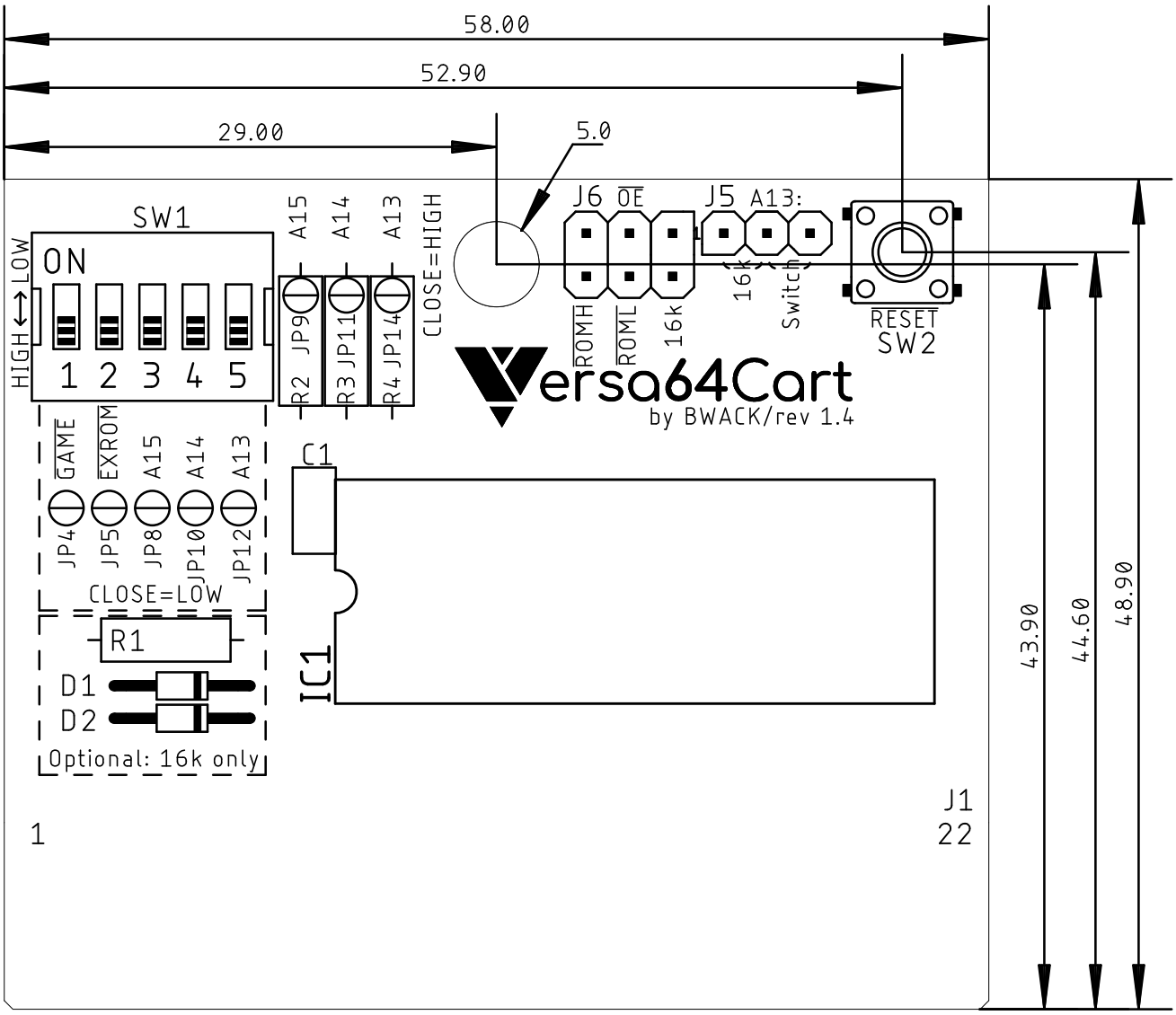
BWACK	Doc.-No.:121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 11:10		Rev.: 1.4
stopmask component side		



BWACK	Doc.-No.:121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 11:10		Rev.: 1.4
stopmask solder side		



BWACK	Doc.-No.:121-2-01-01.4	
	Cu: 35µm	Cu-Layers: 2
Versa64Cart_v1_4		
25.06.2019 10:52		Rev.: 1.4
placement component side measures		



Versa64Cart v1.4

Functional Description

J1 are the contacts, that fit into the Commodore C64 expansion port. The pin numbering is identical to those of the C64. IC1 is the EPROM, which can be socketed or not. C1 is the blocking capacitor, that should help to keep the noise from the supply voltage out of the IC.

The configuration is achieved with the DIP-Switch SW1, jumper J5 and jumper J6. SW1-1 and SW1-2 set the signals $\overline{\text{GAME}}$ and $\overline{\text{EXROM}}$. These signalize the C64 the mode of the cartridge. The switches SW1-3 to SW3-5 are responsible for the (manual) bank select: The configure the three most sufficient address bit of the EPROM, since those are not asserted by the C64, which only addresses 8kB blocks.

For 16kB cartridges, a further address bit (A13) is required. Since this kind of cartridge uses both chip selects ($\overline{\text{ROML}}$ and $\overline{\text{ROMH}}$) the signal $\overline{\text{ROML}}$ is used for this purpose. It will be low, while the lower 8kB are addressed and HIGH, while the upper 8kB are addressed.

The 16k cartridge configuration requires to combine $\overline{\text{ROML}}$ and $\overline{\text{ROMH}}$ to one chip select signal. This function is provided by the wired AND circuit D1, D2 and R1. R1 is pulling the RH&RL signal HIGH. In case one of those signals is low, the RH&RL is driven LOW.

The required chip select signal can be selected with J6.

In 8k mode, the address bit A13 is set by the DIP-switch. Jumper J5 has to be set to the "Switch" position.

While $\overline{\text{GAME}}$ and $\overline{\text{EXROM}}$ utilize the pull-up resistors inside the C64, the address signals A13 to A15 require Pull-Up resistors on the cartridge board (R2-R4). There are solder bridges on the component side if the Versa64Cart, that can be used alternatively to the DIP-Switch to pull each signal low. In case the Pull-Up resistors R2-R4 are not populated, additional solder bridges can be used to set A13 – A15 to a HIGH level (only those, that are set to a HIGH level). It is obvious, that closing the HIGH bridge and the LOW bridge of a signal at the same time produces a short circuit.

SW2 is the reset switch. A pull-up resistor on the C64 mainboard is utilized here.

Versa64Cart v1.4

Testing

Test Description

The test was executed with a Versa64Cart v1.2 (the differences to v1.3 and v1.4 are not functional), a C64G (Mainboard ASSY 250469) and an EPROM type ST micro M27C512 10F1. The programming of the EPROM was done with a TL866II Plus programmer.

The following binaries were loaded into the program memory of the programmer, programed to the EPROM and verified "ok":

Binary	Binary #	Start Address	End Address
Neutron.bin	1	0x0000	0x3FFF
Deadtest.bin	2	0x4000	0x5FFF
586220plus_0_4.bin	3	0x6000	0x7FFF
LalaPrologue.bin	4	0x8000	0xBFFF

The EPROM was inserted into the socket of Versa64Cart, which was then configured, inserted into the C64, the C64 was switched on and the selected binary was run.

Test Execution

Binary #	A15..13	$\overline{\text{GAME}}$	$\overline{\text{EXROM}}$	J6 $\overline{\text{OE}}$	J5 A13	Observation	Testing
1	LLL	L	L	16k	16k	working	ok
2	LHL	L	H	$\overline{\text{ROMH}}$	Switch	working	ok
3	LHH	H	L	$\overline{\text{ROML}}$	Switch	working	ok
4	HLL	L	L	16k	16k	working	ok

Test	Observation	Testing
Does the Cart fit into the slot?	yes	ok
Does the Cart move inside the slot?	The Cart moves 0.9mm from left to right in total. The contacts of the cartridge connector do not leave the contact pad.	ok
Comparison of measures v1.2 vs. v1.1	Both PCBs have the same dimensions. The mounting holes are aligned. The port contacts are aligned, the contacts of v1.2 are wider.	ok
Reset button	Binary #1 can be restarted by pushing the reset button. Some software (e.g. LalaPrologue.bin) does not reset well after pressing the RESET button. It depends on that software, not on the memory bank.	ok
Running cartridge with a breakout board, which adds 90mm to the bus	Binary #1 is running ⇒ The bus length is not critical	ok

The critical signal is the $\overline{OE}$ pulse for the EPROM when the 16k-mode is selected. It can either be too wide or the low level is above 0.7V (the upper margin for a TTL low level).

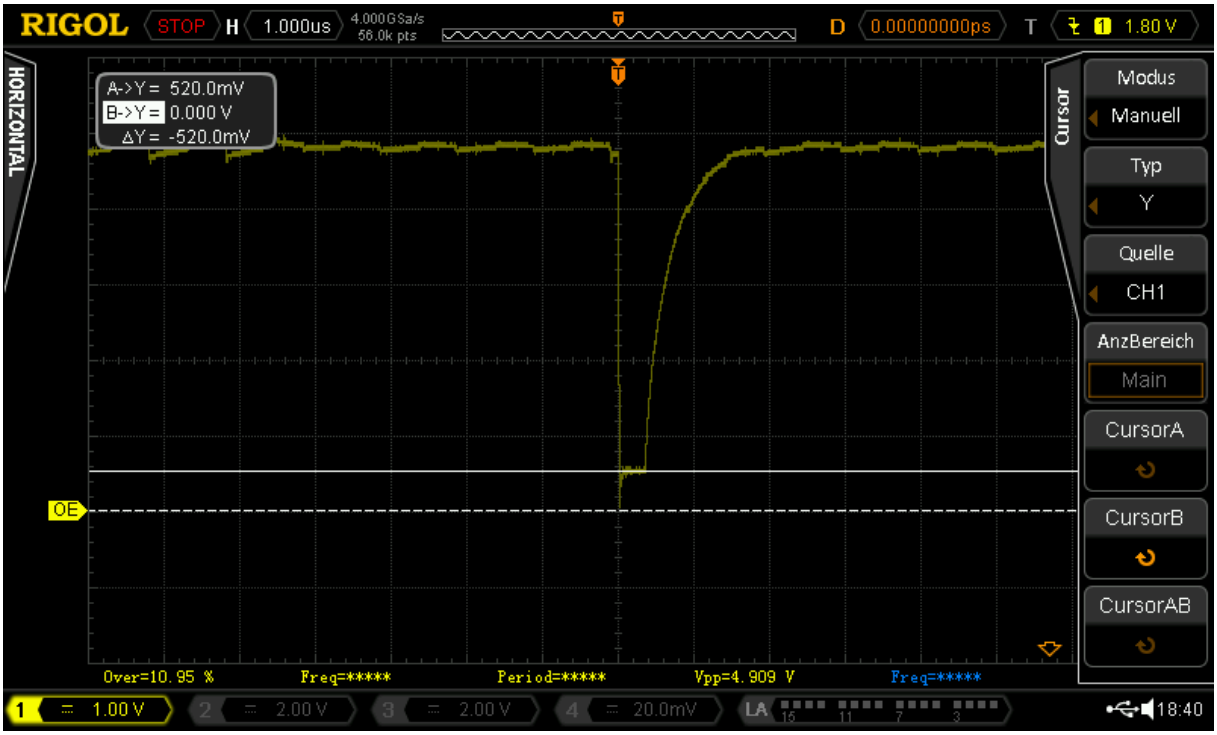


Figure 1: The /OE pulse (low level = 520mV)

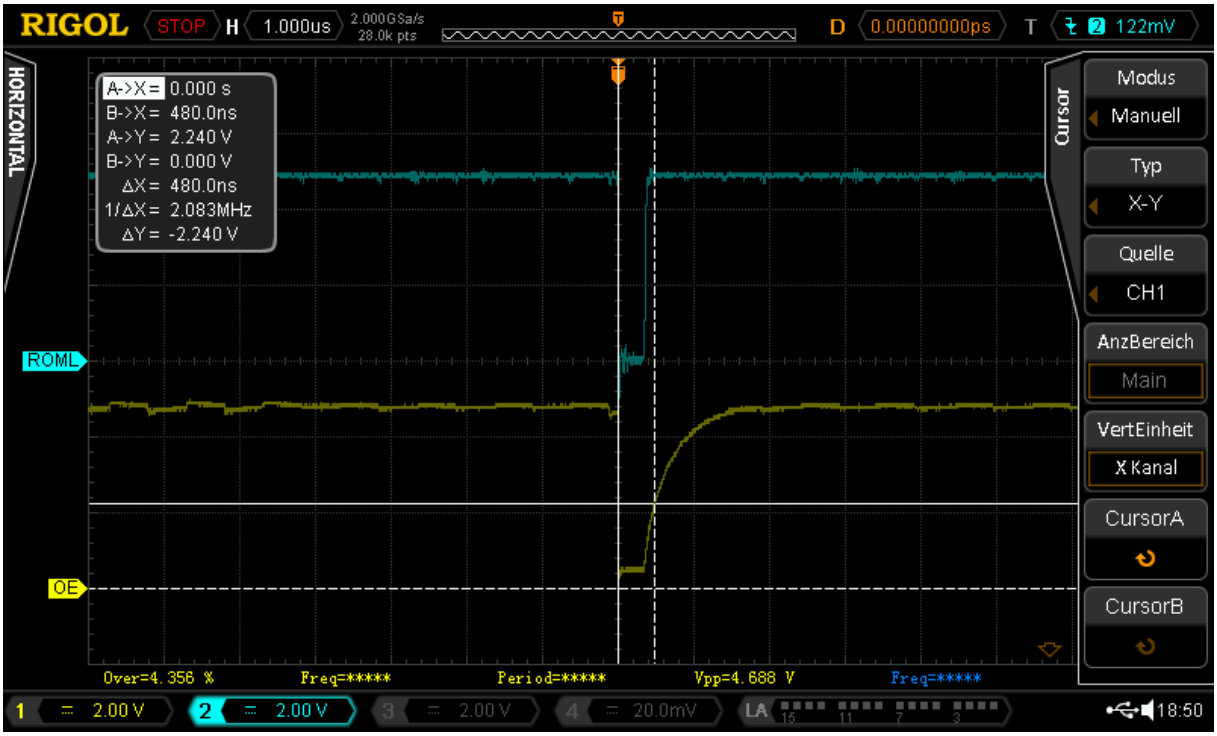


Figure 2: /OE pulse (pulse width = 480ns)

Test	Observation	Testing
Low level of the 16k- OE pulse below 0.7V	The low level is 520mV	ok
Pulse width of the 16k-OE pulse less than 500ns (?)	The pulse width is 480ns	ok (?)

Test	Observation	Testing
Neutron.bin & LalaPrologue.bin were programmed into a 27C256, A15 was set HIGH	Both games started and played properly.	ok
LalaPrologue.bin was programmed into a 27C128A, A15 & A14 were set HIGH	Game started and played properly.	ok
1541diagcart.bin was programmed into a 27C164A, A13..A15:HHH, $\overline{\text{GAME}}$: H, $\overline{\text{EXROM}}$: L, $\overline{\text{OE}}$: ROML	Software started and worked properly	ok

Conclusion

- All extra address lines (A15 … A13) can be asserted
- $\overline{\text{GAME}}$ can be configured properly
- $\overline{\text{EXROM}}$ can be configured properly
- $\overline{\text{ROML}}$ works as a chip select
- $\overline{\text{ROMH}}$ works as a chip select
- The 16k $\overline{\text{OE}}$ works properly, the timing cannot be proved ok, due to a lack of information.
- All binaries are running (16k, 8k Game, 8k Exrom)
- EPROM types tested: ST M27C512, M27C256B, M27C128A, M27C64A

The Versa64Cart is fully functional.

Versa64Cart Rev. 1.4
Bill of Material Rev. 1.40

Pos.	Qty	Value	Footprint	Ref.-No.	Comment
1	1	121-2-01-01.3	2 Layer	PCB Rev. 1.4	2 layer, Cu 35µ, HASL, 58mm x 48.9mm, 1.6mm FR4
2	1	100n	C-2,5	C1	ceramic capacitor, pitch 2.54
3	4	10k	R-10	R1, R2, R3, R4	resistors
4	2	1N4148	DO-35	D1, D2	diodes
5	1	27C512	DIL28	IC1	EPROM. Also possible: 27C256, 27C128 and 27C64
6	1	socket	DIL28	(IC1)	IC socket for the EPROM
7	1	Pinheader 3x1 pin	1x3p	J5	2.54mm pitch (0.1") pinheader (option: 90°)
8	1	Pinheader 3x2 pin	2x3p	J6	2.54mm pitch (0.1") pinheader (option: 90°)
9	2	Jumper 2.54mm	2.54mm	(J5), (J6)	Jumper to be placed on J5 and J6
10	1	DIPSW-05V	DIPSW-05V	SW1	Standard DIP-Switch with 5 switches (switch side facing up)
11	1	JTP-1130	JTP-1130	SW2	Standard 6x6mm tact switch, e.g. Namae JTP-1130 or any other

Rev. 1.2 → 1.3

Pos 1 new board revision

Rev. 1.3 → 1.4

Pos 1 new board revision