ABOUT THIS DOCUMENT

This document has some comments about building the device. But the program debugger pc2n64prj.exe used in this document was for previous version device. Probably that program does not work anymore.

LIST OF TOPIC

COMPONENTS

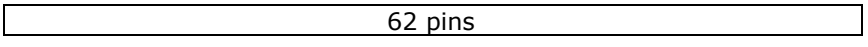
RefDes	Value	Type	Description
C1- C7	0.22	"CHIP CAP100"	"Chip capacitor"
C8	100x16	"POLCAP"	"Electrolyte capacitor"
P3		CON50 (female)	"For N64 Cartridge Connector"
D1 – D4	"1N4001"	"DIODE"	"Diode"
"P1"		DB25M	"PCB-mounted Pin Connector"
"P2"		EDG50M (male)	"Card-Edge Connector"
U4	"5V 1.5A"	"LM7805CTB"	"5 Volt Voltage REGULATOR"
R1- R38	33 Om	"CHIP RES"	"Resistor"
U1 – U2	"74HC245"	DIP20	"Octal Bus TRANSCEIVERS with 3-State Outputs"
U3	"EPM70128"	PLCC84	"In system programming device from ALTERA"
U5	"power"	"POWER"	"Connector for external BP 7,5 Volt"
U6	"SIMM72"	SIMM 72 connector	You can use EDO DRAMM 72 pin 60 or 70 ns
U7	"JTAG"	10 Pins connector	"Interface for programming EPM"
U8	40 mhz	"Oscilyator"	

If your power regulator 7805 has no heat sink and to hot, you can remove the diode (between n64's 12V and 7805) and use 7,5 or 8 or 9 V external power suply.

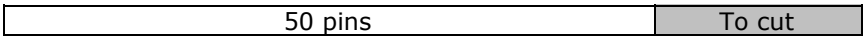
CONNECTORS

You can use connector for SIMM removed from old motherboard. You can sold SIMM w/o connector also. You can use the usual cable from scanner. If you can't find the connector for n64 cartridge - you can use ISA connector. You must to cut the 50 pins part and close edges by plastics plates.

1) Source ISA connector

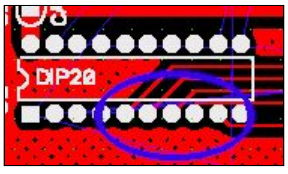


2) What you must to cut



3) Close edges

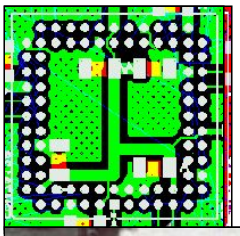
Be careful the size of pin's steps is not same like N64 cartridge. You must look at this fact. The each pin of N64 connector should have contact with pins of new custom connector.



You can use the PLCC84 socket for EPM7128.

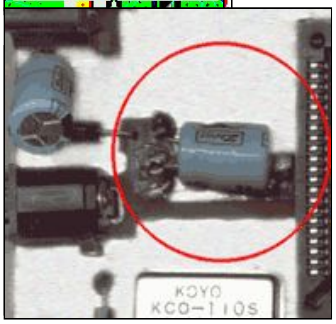
You can use the DIP20 sockets for 74HC245 chips. If your PCB is homemade do not forget make connection between top and bottom side of board before soldering another components.

PCB



The thickness of PCB should be correct for N64 connector's size.

The power and GND wires on the top and bottom sides of the PCB, should be connected by wire in several places. Course it is necessary for the homemade-PCB.



My PCB files is not so good, it was just enough for me. Course it has many untraced wires. All this wires I made by the prototype-wire. Look at right picture, I show there how I connect two sides of PCB.

The each pair VCC and GND of the PLD should have decoupling 0.1mF. And additional electrolyte capacitor 100mF should be placed too all PCB (Look at schematic). You can see this capacitor on the left picture.

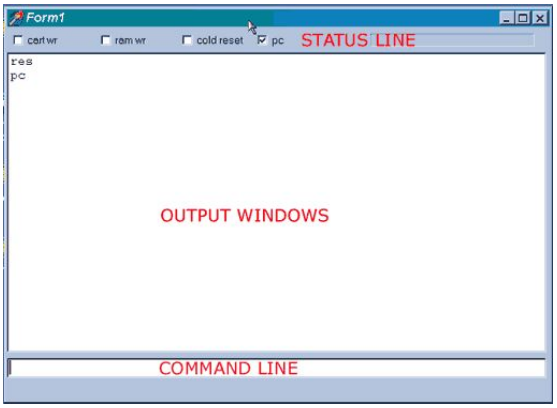
PROGRAMMING OF PLD

The "ByteBlasterMV" cable can program the PLD. The ISP is the "In System Programmable" abbreviator. The ISP has another name JTAG interface.

You must to use the binary file UPLOADER.POF for ISP programmer. Y can also to build own binary file in the CAD system for PLD. I used the "ALTERA MAX II+plus" which can be downloaded from www.altera.com for free. You can download the schematic of ByteBlasterMV cable also.

You can't use EPM7128LC-10 (w/o "S").

DEBUGING



You must to check all nets for proper connections. The next step is checking of 4 and 5 and 12 volts signals for shot circuit to GND. After that you can give a power and to control temperature of ICs. It should not be too hoot. Also you need to check voltage on the power lines.

You must set on the EPP1.9 mode in your BIOS settings. Then start the pc2n64prj.exe program. This program has help, just enter "**?**" command in the command line.

Switch off the power of n64 but switch on the power for device. Execute the command "res", all checkboxes should be 'off'. Then execute command "pc", after it check box "pc" should be on.

You can switch on power on the n64 and send any command "pc" or "stat", after that the check box "cold reset" should be on.

And now, you can show the dump of DRAMM. Execute command "d" for that. The random data should be in the RAM. Try to fill DRAMM by command's "f 0 0 0", and then dump it again.

```
D
00000000: 0000 0000 0000 0000 0000 0000 0000 0000
00000010: 0000 0000 0000 0000 0000 0000 0000 0000
00000020: 0000 0000 0000 0000 0000 0000 0000 0000
……..
```

Fill memory by 0xFFFF

```
F 0 0 FFFF
D
00000000: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
00000010: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
00000020: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
```

Fill memory by 0x55AA

```
F 0 0 55AA
D
00000000: 55AA 55AA 55AA 55AA 55AA 55AA 55AA 55AA
00000010: 55AA 55AA 55AA 55AA 55AA 55AA 55AA 55AA
00000020: 55AA 55AA 55AA 55AA 55AA 55AA 55AA 55AA
```

After it you can execute the "test**"** command, the program should test DRAMM and show errors.

```
test
Write increment
Write runing 1, now ALL 01
Write runing 1, now ALL 02
Write runing 1, now ALL 03
……
Write random
```

If no errors happen, you can try upload the game for example mario64.

```
Ld mario64
Loading: mario64.v64
<size>

send
…..
```

If no errors happen you can switch on DRAMM for n64 by "n64" command.

```
N64
```

And switch off then switch on power of n64. It should start a game … ☺

If game was not started, you can send "**pc"** command and then compare memory with source file for problems. Use command "**d"** the DRAMM dumping and "**dr"** for source file dumping. You can also use command "**cmp"** for comparing. If memory has problems, it is very bad ☹.

But if there was no problems, you can try use cart boot mode, In this mode N64 start game from daughter cartridge. After command "**send"** use command "**mode gm"** then "**n64",** N64 should start game from the ROM.

In a good result execute the command "**mode",** it switch n64 from ROM to DRAMM again. The sound should crash now, it is OK. But you can research the device by scope now, because the game N64 executes the game and access to DRAMM for sound data.