

ISA bus slave NS32016 processor board

By Dave Rand and George Scolaro, 1985.



Keith

Follow project

Like project

Join this project

1.4k

views

3

comments

12

followers

12

likes

Description

Details

Files ¹⁴

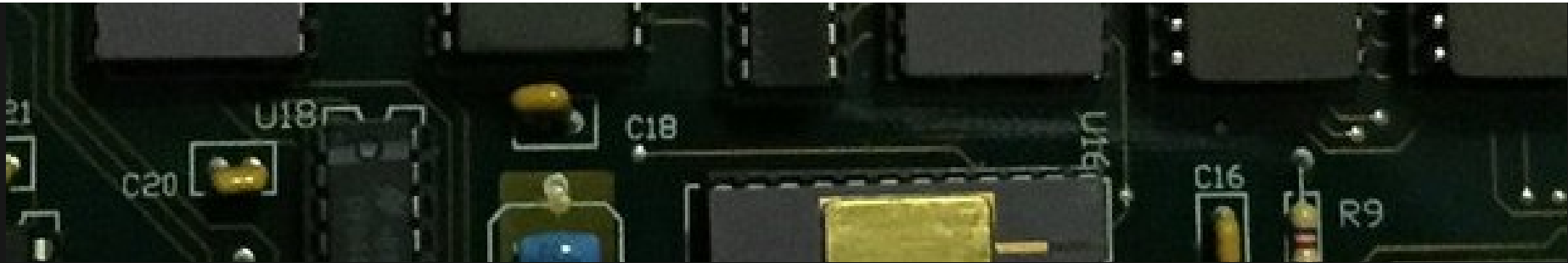
Components ⁰

Logs ¹³

Instructions ⁰

Discussion ³



[View Gallery](#)

👁 1.4k 💬 3 👤 12 ☠ 12

TEAM (2)

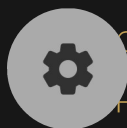


Keith



Andrew Lynch

[Join this project's team](#)



Original source

Harston 32016 Tube page



Andrew Lynch reproduction

- ★ DP84412 datasheet
- ★ DP8419 datasheet
- ★ 32000 cross assembler restoration

🕒 ONGOING PROJECT

NS32016

ISA BUS

16-BIT

This project was created on 03/09/2023 and last updated 2 years ago.

DESCRIPTION

<https://archive.org/details/ns32k-archive> seems the internet's biggest archive of NS32K material. There is software, and a scanned magazine with articles about building an NS32016 board called the Public Domain 32000. It includes circuit diagrams. It uses 256 kbit DRAM chips, and PAL16L8 chips with logic equations (yay!) but also uses a pair of rare DRAM controller chips (boo!). I have OCR'd and HTML'd it, and uploaded it for people to download and read. In another language if your browser has translation abilities!



DETAILS

The PD32, was published in Micro Cornucopia number 32, (October/November 1986), page 6.

Supports the 32016. Has 512k bytes of DRAM modules.

It is a complete processor board.

It has no bus interface.

It communicates through an 8-bit bus, but not a latch. I don't know how that works - do they wait state until the other side has read the bus.

The article talks about interfacing to the Z80 bus.

I'd prefer to get it talking to the STEbus or an FTDI USB FIFO module.

It comes with all the PAL equations and software to implement UNIX.

I could not find the DRAM controller chips on eBay, so I presume they are effectively extinct in the wild.

The design could be adapted to use modern SRAM chips. A pair of 512k x 8 bit chips would provide double the original RAM.

The byte-wide communications interface could be adapted to use Jonathan Harston's single-channel Tube API, and thus allow Acorn PANOS to run. But that's a job for people with more time than me.

I'm not that motivated to make a PD32, my hobby is mainly capturing electronic design details for posterity and people who want to make them. I don't have time to more than a few pet projects, nor do I have space to store stuff.

I had a quick go at assembling the ROM source and the PAL equations but failed. They need major syntax edits and checking.

FILES

PD32 V1.0 Gerbers.zip



Files for manufacturing

Compressed - 1.16 MB - 02/15/2024 at 20:23



Download

PD32-schematic.pdf

schematic

Adobe Portable Document Format - 766.96 kB - 02/15/2024 at 20:23



Preview

PD32-PCB.pdf

PCB layout

Adobe Portable Document Format - 682.59 kB - 02/15/2024 at 20:23



Preview

PD32-FRONT.jpg

3D render front PCB

JPEG Image - 194.45 kB - 02/15/2024 at 20:23



Preview

PD32-BACK.jpg

3D render back PCB

JPEG Image - 194.71 kB - 02/15/2024 at 20:23



Preview

[View all 14 files](#)

PROJECT LOGS



Tech data for build your own PD32

Andrew Lynch • 02/16/2024 at 14:31 • 0 comments



I added tech data so you can build your own PD32 or at least the reconstruction of it. There are schematics, PCB layout, two BOMs, 3D rendering images, and Gerber files.

If anyone decides to make their own PD32 using the supplied Gerber files please post here with your results. Especially if any changes are

necessary and/or you find improvements.

Would really like to see your results of a PD32 build. Thanks!

Cross-assembler recovery

Keith • 02/11/2024 at 01:26 • 0 comments

There is a Definicon cross-assembler but it is an MSDOS executable binary so I can't run it on my Linux PC nor can I compile it for Linux because there is no source-code.

There is C source code for one in Dr. Dobbs' Journal. The OCR text is not good because all the OCR tools I can use for free try to automatically recognise text in columns, with no way to specify single-column operation. Which mangles the order of source code scans.

Several days of tedious salvage work later, I've got the 32000 cross-assembler to the point where it compiles, but it crashes with a memory segmentation fault. Probably trying to do something that used to be okay on MSDOS but not on a modern OS.

I don't have enough spare time to get it working, but there are many capable coders out there, and I am sure one of them could get it running fairly quickly.


I notice that most of the code is doing lexical analysis, splitting the source into lines and words, turning words into symbols and numbers.

I have just started learning Python, and it has vast libraries for doing this kind of work. If I were to write one from scratch, I would start with Python to make the job easier and quicker.

PCB trace routing

Andrew Lynch • 02/07/2024 at 14:24 • 0 comments

The PD32 PCB routes in in FreeRouting as a 2-layer board, however, based on what I've read the NS32K chipset seems rather sensitive to signal quality. A 2-layer board is less expensive, but if a 4-layer design is more likely to work, then I think that's the way to go.

ns there were some articles and/or letters in MicroCornicopia after the original PD32 article suggesting PCB quality caused a lot of problems to the initial roll out in 1986. They also used an improved 4-layer board to improve signal quality, so I think it is prudent to do the same.

The good news is the PCB easily routes as a 2-layer board so transitioning to a 4-layer PCB should be no problem. Just add two internal copper layers, one for GND and the other for VCC.

[View all 13 project logs](#)

ENJOY THIS PROJECT?

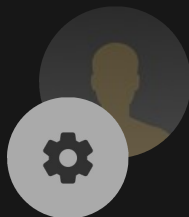
Share



DISCUSSIONS

Log In or become a member to leave your comment

[Log In/Sign up to comment](#)



Steve Jones wrote 02/14/2024 at 10:54

Interested! I see most of the things I was going to mention over on the RetroBrew forum, except for Andrew Voropay's wiki page (<http://wiki.sensi.org/dokuwiki/doku.php?id=ns32ktoolchain>) about using old versions of GCC and binutils for cross-compiling.



Andrew Lynch wrote 02/05/2024 at 17:25

Great mailing list for NS32K enthusiasts

~~http://wiki.ccs.neu.edu/view.php?id=ns32k/pcb32/pcb32/~~
I would like to build a reproduction of PD32/pcb32/However, this is an enormous project and will require multiple hands to make it work. I would like to assemble that team to do it.

I've captured the design as close to "as is" to the design described in the original MicroCornicopia article as I can make it. I think it would be easy to make some updates to the design, however, making modifications to a design which has not been through initial build and test is quite hazardous. Premature optimization is the root of all evil in my experience.

The PD32 hardware and software information exists (thanks to the generosity and wisdom of Dave Rand and George Scolaro), all we need are PCBs (easy to manufacture) and at least a couple of people willing to build and test. Preferably with NS32016 experience but at least the capability to bring up a new design to working condition and work around any errata from the original article design and/or introduced during the translation to KiCAD.

Also, a skilled software person would be needed to reconstruct the software available for the Host PC side and the Unix System V software. Please note, the Unix System V software is likely IP encumbered so that will affect any future distribution. Long term, that can be mitigated by using Minix or other free/open-source operating system. Plus, we'd need some documentation once we had working hardware and software.

In short, this is a huge project and way more than one person can reasonably do. I have brought the PD32 PCB hardware to the table and willing to share in build and test. However, I do not have actual experience with bringing up NS32016 hardware, so we need some help if this is to work.

If you want to help out, I have a few tasks that can be done before we even have hardware.

1. pin audit of the KiCAD (PDF) schematic against the original article. I have done this already, but another set of eyes would help ensure the best capture possible. Involves literally printing out the article schematic and the KiCAD PDF and going through connection by connection, pin by pin, checking them off to ensure they match. There are some discrepancies by the nature of the translation (intentional), but we are looking for the unintentional differences (errors) which would hamper build and test and successful bring up of the board.

2. datasheet audit of the components against the KiCAD footprints. The KiCAD design relies on an NS32K footprint library, and it likely has errors. Literally go through the datasheet for each part and compare to the parts in the KiCAD (PDF) schematic to ensure the pins match. I have seen this before where the datasheet and/or KiCAD footprint library have errors that get propagated into designs and turn up at the worst possible times. Major risk mitigation step.

3. Review and analysis of the 3D rendering of the PCB. Step back and look at the PCB. Are there things about it that



don't make sense? Like the bypass capacitors, crystal oscillators, jumpers, missing labels, etc. All of those can have glaring errors obvious after build and test but maybe with a close inspection we can capture beforehand and increase the likelihood of a successful first time bring up of the PD32 prototype.

Further areas where expertise is needed:

1. Reconstruction of the toolchain. The design includes source code (ROM.A32) for the onboard ROMs. Find what toolchain the A32 source code is associated with and post toolchain step by step instructions. Probably something obvious but I have no clue. We need even and odd binaries to burn to EPROM (or EEPROM or Flash) to boot the system. We go nowhere without the ROM images. Super important!
2. Reconstruction of the PALs into GALs using PALASM (or GALASM). We need JEDEC files to convert the PLDs into something we can use. PALs are difficult to obtain these days (not impossible), but the design should be converted to use more common Lattice or Atmel GAL22V10 and/or GAL16V8 components. Especially the NS DRAM controller uses a component which is unobtainium (DP84412) but can be reconstructed with a fast GAL22V10. (I am guessing a 10ns part will work) Super important step. Like the ROMs, we are going nowhere without GAL JEDEC files.
3. Reconstruction of the Host PC software. Source is included but for what compiler? C of some kind presumably. Maybe GCC? The code would have to be translated to a modern C compiler (preferably SDCC or some open-source equivalent) and the binaries tested on target hardware. Again, super important step.
4. Reconstruction of the Unix System V software. This speaks for itself. It is a huge step and really the whole point of the exercise.

If you would like a challenging and historically relevant project to work, this is your opportunity!

Post here for replies and comments, questions, etc.



SIMILAR PROJECTS