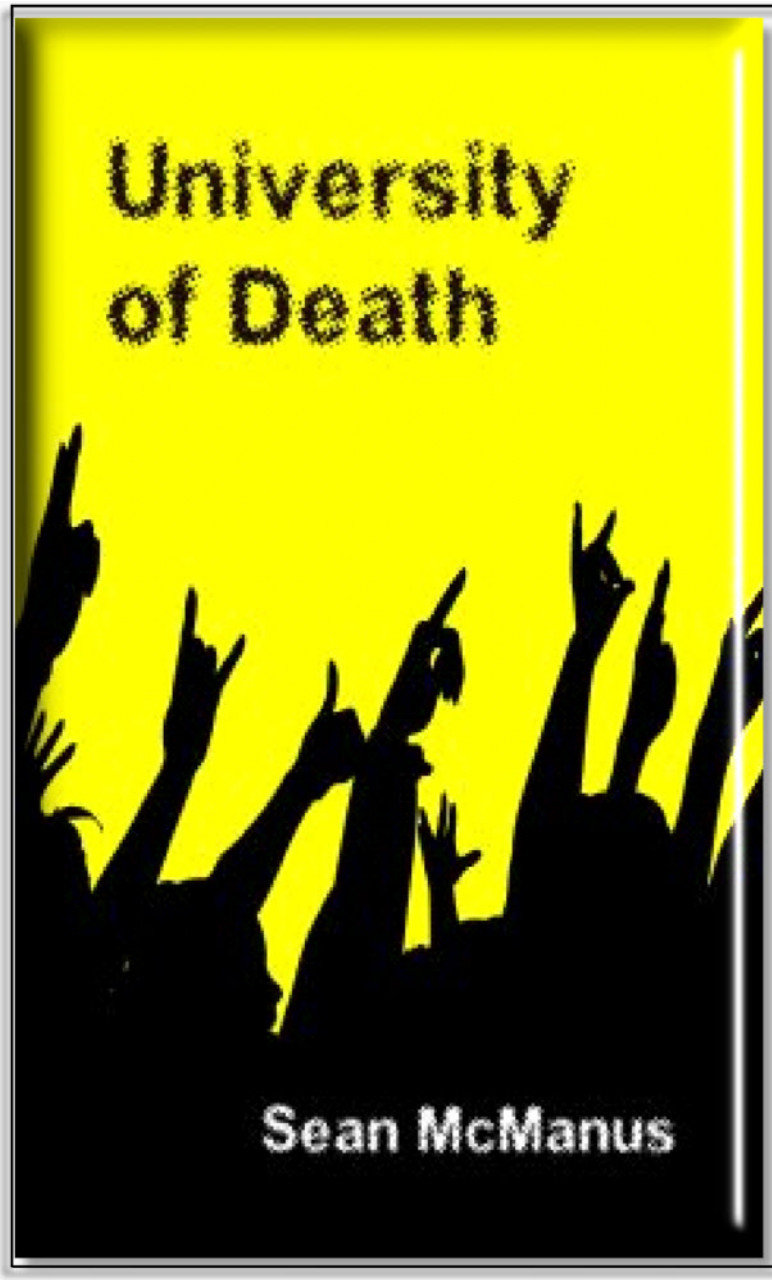
Commodore Free

Volume 3 Issue 1 Released January 2009

Free to download Commodore magazine
Dedicated to Commodore Computers
Available as PDF Text SEQ HTML and D64 image
www.commodorefree.com



CONTENTS

IN THIS ISSUE

How to help Commodore Free Magazine

HOW CAN I HELP COMMODORE FREE..
Ok the best way to help would be write something about Commodore
articles are always welcome,..

WHAT ARTICLES DO YOU NEED..
Well they vary, contact me if you have an idea but I am looking for..

Tutorials..
(beginners and Expert),..
Experiences..
with Commodore,..
Why I love Commodore machines,.

.**Interviews..**
maybe you have access to a power user.
News
Club meeting
General Commodore news
New Products

Editor

http://www.commodorecomputerclub.co.uk/
http://translate.google.com/translate_t#

Happy new year!

Translation
Right a few readers have asked about Commodore Free as a German, French and Spanish publication. I am one person editing the magazine, I speak only one language (and am still trying to master that correctly) I can't possibly translate the magazine to other languages on my own. I have found this website though, run by Google: http://translate.google.com/translate_t# where you can convert to and from many languages. I have tried the site with the TEXT version of Commodore Free and it did seemed to work ok, well for my schoolboy French. Something, however is lost in the translation, especially if you transfer back to English again, but this is better than nothing, and I am hoping you will be able to pick your way through the translations to your native language.

Now back to Commodore Free magazine
For anyone who knows me, and wonders why the heck I still load games from tape (and most think I am absolutely mad for continuing to do so), heck people tell me with uiec and mmc and all manner of sd card readers what's the point of using flaky tape devices? We have an interview with Tom Roger from the Tapes 64 website (a tape preservation website) http://tapes.c64.no/index.php So why do people load from tape, well for me personally tape was the first storage format I ever had, loading from tape is slow and this builds up anticipation, loading screens and music displayed during the waiting process really added to the game. I also personally feel that the time it took to load a game made the user get more involved with the game and play it for longer, if you have almost instant access to any game like from an SD card reader you will play a game for a few seconds, then think heck I could be playing xyz load xyz for a few seconds then think I haven`t played abc for ages then load abc and play for a few seconds. Of course loading from tape using a none turbo loading system and having to wait 20 minutes for something like Arcadia to load, well even I'm not that dedicated a user.

Also for this issue I have also had permission to reprint "On the Road the Commodore Scene meeting 2008 and a visit to AmigaKit by Robert Bernardo" Part of Robert's tour of the Commodore community. You may remember Robert visited "commodore Scene" well ok what was left of Commodore Scene it was myself, Shaun Bebbington and Allan Bairstow. At the time of the meeting Allan commented that meeting every 2 years was too much, we all agreed we needed a local club and have meetings more regularly, hence the emergence of the "Commodore Computer Club U.K." in the article I have included a picture Robert took showing myself in the centre and Shaun Bebbington and Alan Bairstow. Some of you may have already seen the article, but I know you wont have seen the picture.

Lord Ronin continues his beginners guide to using the Commodore 64 with a look at the final chapters of the User guide that was included with the machine, I know some readers have said "what is the point" but if you have been reading the article Lord Ronin does give some easier to understand information, he has also corrected some of the listing errors in the manual, I guess it was rushed with the launch of the machine. Some more information I have converted the manual to a 40 column document with ASCII art, this is being checked on Commodore 64 to ensure readability and then I intend to create a disk menu with the manual.

The news section sees some more innovations from Individual Computers with a flicker fixer for the Commodore 64, I would love to see this device working.

Also on the news of the Commodore Computer Club, we should have a working bank account (although I am writing this in the past tense) after various problems it seems the bank has excepted us as users, I can only put the problems down to the credit crunch as we gave them everything they wanted then they would say there was information missing and ask us to refill in the forms, with a bank account we sort of now look semi-professional.

I don't really know what to do for the best at this moment. All I can say is that Commodore Free may go more infrequent,

I have received various hate mails about Commodore Free, and at one point I thought about just stopping production and doing other things. But I did receive some emails from users expressing an interest in the magazine and wishing me good luck. I am paying rather a lot for web hosting, but I feel it does give me flexibility, and although its expensive the support is excellent. I don't really know what to do for the best at this moment. All I can say is that Commodore Free may go more infrequent, mainly because of the time to research and create the magazine, and the lack of time I have to myself and my Commodore.

I welcome comments
Regards Nigel
www.commodorefree.com

EDITOR	Nigel Parker
SPELL CHECKING	Peter Badrick
DISK IMAGE	Al Jackson
CONTRIBUTIONS	Lord Ronin

INDIVIDUAL COMPUTERS NEWS

5.10.2008: new products in fall 2008, Amiwest, holidays

Indivision AGA successfully introduced
After delivery of Indivision AGA has started about one week late this September, all units of the first production are already shipped to our resellers. We were prepared for a good demand, but are still surprised about the extremely high demand for a 24-bit flicker fixer. A second production run has already been started and it will be available in November of this year. Since our trade partners all have stock at this point, we don't expect a shortage in the coming weeks. One feature that many customers are pleased about is the HighGFX support of Indivision AGA. With the new flash update V1.2 alpha, we're extending screen modes to true HD720 wide-screen-resolutions with up to 1280x720 pixels, displayed with more than 60Hz refresh rate. Special thanks goes to André "Ratte" Pfeiffer, who adapted his HighGFX package especially for Indivision AGA.

Flicker fixer for the C64-
On September 13th, we have announced our new product for the C64 at Back In Time Live in Stockholm, Sweden. We have worked on this product for the past two years together with developer Peter Wendrich. The result is an extremely user-friendly cartridge that can be used without opening the computer. It is just plugged to the expansion port of the C64. Technical data in short:

- VGA-output with 60Hz refresh rate or more
- Turbo-function with full 6510-processor compatibility
- cycle-exact REU (memory expansion) with 16MByte ram
- MMC/SD card slot with MMC64 compatibility and 1541-emulation
- connector for PC-keyboard
- built-in help function "the book" with 16MByte flash
- freezer (compatible with Retro Replay)
- bright yellow case with blue buttons

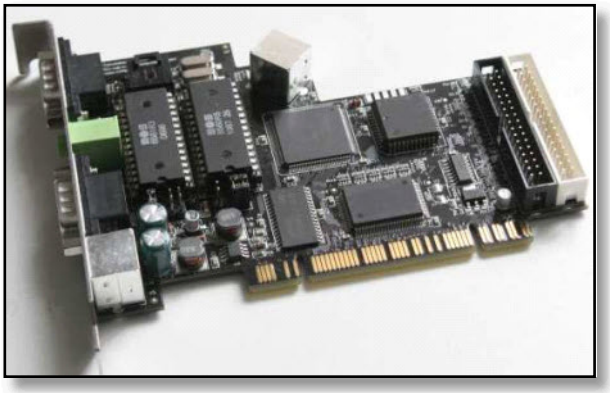
Chameleon will presumably be available in the first quarter of 2009. The price will be about 220,- EUR including German sales tax of 19%.

Micromys V3 available in November
The mouse-adapter Micromys V3 that we have already announced in December 2007 will finally be available in November 2008. We're proud to say that star-programmer Chris Hodges of Poseidon fame will be writing the Amiga-wheel drivers. His publications for the Amiga made him an excellent reputation in the Amiga market. He is filling a gap that Michael Schöttner has left after he could not start working for us because of family reasons. Another reason for the delay was the case: We could not find a vendor for a long time, but finally found someone who is willing to do the small quantities that we require. The picture below shows a first prototype that will be revised for mass-production. We decided to make a model with a cable in order to deal with space constraints on every desk. This design will allow installation on every narrow desk, even when a bulky USB-PS2 adapter is used.

Catweasel MK4plus available
The Catweasel MK4plus is available starting today. It replaces the Catweasel MK4, which has been sold out earlier this year. The main changes are cosmetic, and we have followed customer feedback in some places. The one easily visible change is that the new card is no longer low-profile PCI compliant. This feature of the old Catweasel MK4 was rarely used by our customers, so we decided to use the increased space for a better arrangement of the two SID sockets. These are more easily accessible now. Additional filters in the audio part are geared towards filtering noise from high-performance graphics cards and low-quality power supplies. Another novelty is the external audio jack and an angled internal audio connector for better accessibility. The Catweasel MK4plus is delivered only with an English manual at first, a German translation will be made available in November of this year. picture shows extras. Unit is delivered without SID-Chips.

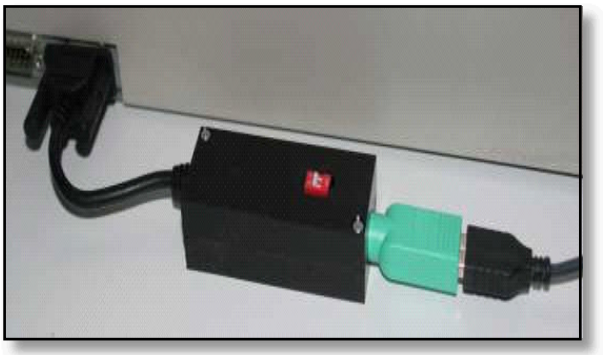
Lyra 2:
Keyboard adapter with new functions
We've been delivering the keyboard adapter Lyra for eight years now. It allows connecting PC-keyboards to Amiga computers. We're now delivering a new version called Lyra 2 that implements a few new features requested by our customers. Lyra 2 supports small-size keyboards without a numeric keypad, and multimedia-keyboards can be used with Guido Mersmann's tool MMKeyboard. The Amiga 1200 version now supports Amiga keyboards as well. The online manual gives an overview over the new functions. Just like the Catweasel manual, this manual is also available in English at this point. A German translation will be made available in November of this year. Lyra 2 is available from our trade partners starting today.

C-One goes Amiga
The FPGA-based computer C-One is ageing. Despite that, it marks a milestone in computer history, because it's the world's first computer that is



purely based on programmable logic chips that can re-program each other while the machine is running. In November of this year, we'll be providing an extension card that quadruples the FPGA space of the C-One. The board then has enough resources to run the "Minimig" core. C-One pioneer and FPGA-developer Tobias Gubener has invented a very special solution for his version of Minimig for the C-One: Not one, but two 68000-compatible processors are working in the FPGA extender. One of them does the job of the PIC controller on Dennis van Weeren's Minimig board, and the other replaces the physical 68000 CPU of the Minimig board. If you already own a C-One board, the extender can be bought for 99,- EUR from us. New boards are only delivered with the extender for a bundle-price of 333,- EUR. If you are interested in purchasing the extender only, please contact us in the coming weeks, so we can get an estimate of the required quantities.

Individual Computers sponsors Amiwest



On October 17th and 18th 2008, the yearly Amiga-show "Amiwest" took place in Sacramento, California. We don't only exhibit there, but also co-sponsor the show. Together with our retail partner AmigaKit from England, we provide enough money to grant free admission to the show. Tickets for the banquet on Saturday evening are 29,- US Dollars each. Up-to-date information about the show can be found in the show blog.

Holidays until October 27th 2008
We're on holidays starting from October 6th until October 27th 2008. Please understand that only limited email support can be provided during that time. Unfortunately, Vesalia is also closed for fall holidays at the moment, so the newly available products are not yet listed in their web shop. However, orders can be placed and processed.

All prices are recommended retail prices and include the German sales tax of 19%. Prices at our retail partners may vary. For mail ordering, shipping may be added. individual computers Jens Schonfeld GmbH

NEWS

Crazy Light Construction Kit 3.0

There is a new Construction Kit to make Boulderdash caves. With the following features:

- No One's Enemy Designer v3.0.
- Better scroll detection.
- Grafik optimization.
- Drive independent so you can use a 1581 or a hard disk.
- All editors are enhanced or completely rewritten.
- The ghosts and surprise explosions are replaced by falling walls, acid and boxes.
- Title screen with more colours, more text characters.

There is a instruction video available to see how the Construction Kit works. http://www.gratissaugen.de/erbsen/plans.html

XRoar (Amiga)

XRoar is an emulator for the Dragon and Tandy range of computers. The Dragon and Tandy are eight bit home computers based on the 6809 CPU. Initially produced in Wales in early 1980s, later it spawned some variants in other countries as well. Even if using a very powerful processor - the eight bit smaller brother of the Motorola highly successful 16/32 bit 68000 and 68008 family, the lack of good peripherals, non quality software and weak user base, made the computer unpopular despite being with reasonable specifications for the period it was introduced. Nevertheless with XRoar you can experience the Dragon computer and tinker with the software available for it.

The Amiga version uses ASL file requester for the selection of the virtual diskettes, cassettes or cartridges. There are 3 video modes available for selection. The YUV overlay video output mode displays on resizable window, which can be used directly on Workbench. The emulator supports Joystick via Amiga Input, have sound output and recognises several formats for diskette images, snapshots and tapes. XRoar is a new development, under constant updates but is available already for GP32, Nintendo Dual Screen, Linux and PSP. The Amiga version contributes to its wide availability. Because of legal reasons the original firmware ROM images are not included in the archive, but only a freeware replacement with very limited functionality. The replacement gives the possibility to load software, without the need of original firmware ROM image files. If you don't have any firmware images, you might like to try this one. It doesn't do anything beyond trying to boot OS9 (Dragon version), and only a few cartridge images work with it (mostly CoCo software - Dragon carts seem to always contain many more direct jumps into ROM). If you do have images, but want to try this one anyway, save it out and use "-extbas FILENAME" to load it instead of the normal Dragon 64 image.

Popular software titles for the Dragon computers include games
- Manic Miner,
- Frogger,
- Chuckie Egg,
- Jet Set Willy,
- Donkey Kong,
- Cassette 50,

operating systems
- OS9,
-Dragon DOS,

productivity software
- Dragon Graphic Studio,
- File Master, Rainbow Writer,
- 6809 editor assembler.

There is software for Dragon in the Others section. Not being that popular, nevertheless, the XRoar emulator providing Dragon emulation with enhanced functionality compared to the other Dragon emulators for Amiga, gives great entertainment for the people enjoying the retro computers and the software written for them.
http://hirudov.com/amiga/XRoar.html

The 8Bit Philosophy a Commodore 64 Symphony Released

A documentary about c64 musicians and Commodore remixes is available for download now, there are two versions available:

Long Cut (40 mins, MPEG 2, 1.7 GB):
http://8bit.scenesat.com/The.8Bit.Philosophy.long.cut.version.1.0.16.9.MPEG2.progressive.mpg

Short Cut (Bit Live 2008, 20 min, MPEG 2, 1GB):
http://8bit.scenesat.com/The.8Bit.Philosophy.Short.Cut.1.0.16.9.PA

UN- NEW

This email arrived into the Homestead mailing list, the sender wanted to UN NEW a basic listing, I thought with the Lord Ronin series for anyone following the Commodore manual this may be of use. The NEW command, as you are aware from the tutorials in earlier issues of Commodore Free, clears a basic listing from memory, so what happens if you want the listing back, or typed new in error? How does it work though?

-----Original Message-----
From:] On Behalf Of Cameron Kaiser
To: dunric

Subject: Re: [Homestead] UNNEW command in ML in BASIC 2.0?
> Is there an UNNEW command in ML via BASIC 2.0? Perhaps via a BASIC Loader?

This is my usual incantation:
poke2050,1:sys42291:poke45,peek(34):poke46,peek(35):clr

personal: http://www.cameronkaiser.com/
Cameron Kaiser * Floodgap Systems * www.floodgap.com

Plus/4 World Closed re-housed and re-opened

The Official (Re)Opening Of Plus/4 World! - 2008-09-26
Our previous host, emucamp.com is unfortunately gone forever. Let us take this opportunity
to thank the good folks at ztnet.com for their generosity, we appreciated having a good
home. However, it's now time to move on. In a truly incredible move, Mike has come to the
rescue and offered to give us a new home! It's very cool to finally have our own domain name

:-) And (obviously) here it is: you are on Plus4World.com! Also big thanks to everyone who
offered their help with finding a permanent hosting solution for us. Obviously things are
still not 100% (there's tons of stuff to upload), and we might have minor glitches here and
there. If you find problems, report them, or better yet, volunteer and try to help out. If
you like this place (and from the incoming emails it appears there are many of you who do),
be sure to thank Mike. Cheers! http://plus4world.powweb.com/

C- ONE Website News

From: cone_cores
Subject: [cone_cores] C-One.net

Just to let everyone on the group know, www.c-one.net has these features to offer the C-One
community:

- Wiki
- FAQs
- Downloads
- Picture Gallery
- Forums
- Newsletter
- Blogs

The forums are the only dedicated forums for the C-One. I've created several categories and
posted a few "seeder threads". Go check it out!

Commodore Free Magazine.

Items are always wanted to help produce Commodore Free magazine, feel free to contact me with any news / Reviews / Or features for the magazine.

Trans Logic 2

Skoro from the group Assassins has released a new game entitled Trans Logic 2. This is a sequel to Trans Logic. You have to slide the stones so that the left screen equals the right screen. The game is made for the PAL C16 or Plus/4. The game can also be played with the VICE emulator. Music was composed by Simon (SLD) and the code work was created by Varga (Skoro).
http://plus4world.powweb.com/software/Trans_Logic_2

VICtoria Gold Edition

by Orion70 and Mike for 16K expanded VIC 20

VICtoria Gold Edition is the latest update of the turn based strategy game which tasks the player with being the Emperor of Rome. This latest version requires a 16K expanded VIC and uses some of that space for some very nice graphics. It can be downloaded here http://it.geocities.com/orion010870/VICtoria_gold.zip along with a cover image and manual (please note that it's hosted by Geocities so the file may not always be available due to their bandwidth restrictions). "VIC=toria GOLD Edition is a turn-based strategy game for the VIC-20 expanded with 16K RAM. As the ruler at the dawn of Roman power, your task in centuries to come is to conquer all the known world, region by region. It won`t be easy progressively stronger empires, barbarian invasions, and civil war will keep you occupied in the struggle for your own survival."

Edge of Disgrace by Booze Design

http://www.youtube.com/watch?v=yFdjWSaDlIo
http://www.youtube.com/watch?v=0b4uGv-9xpw&feature=related

Ok remember I kept harping on about a plain vanilla Commodore 64 entering the charts with minimal sound processing, well I am sure if the excellent music track from this demo was released, it would sail to number 1. The music is superb, I was tapping along, the actual demo itself is equally impressive. However not being a demo techy I can only say you need to watch it for yourself.

Commodore Documentaries

Okay, for those who liked the VIC-20 documentary, there is a 2/3rds finished C64 documentary. Here are the videos so far:

Commodore 64 Documentaries

Chapter 1 - Hardware
high res -
http://www.youtube.com/watch?v=qJNXGW80U0s&fmt=18
low res - http://www.youtube.com/watch?v=qJNXGW80U0s

Chapter 2A - Games
high res -
http://www.youtube.com/watch?v=BKFGpuzhcbM&fmt=18
low res - http://www.youtube.com/watch?v=BKFGpuzhcbM

Chapter 2B - Games
high res -
http://www.youtube.com/watch?v=RyZFHZdEZaY&fmt=18
low res - http://www.youtube.com/watch?v=RyZFHZdEZaY

Chapter 3 will cover operation of BASIC, GEOS, and touch on terminal programs.

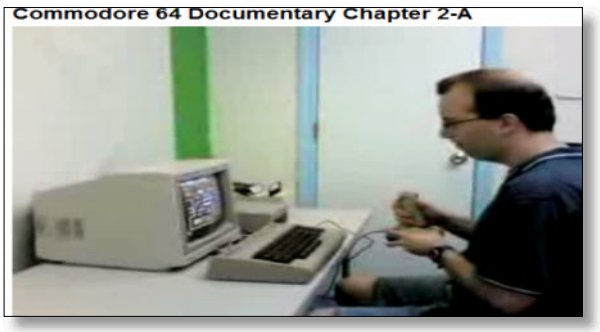
The Vic 20 documentaries

The Vic 20 documentaries are available still from YouTube Vic 20 Documentaries

Chapter 1 hardware
http://www.youtube.com/watch?v=tV7A_PwXEX0

chapter 2 games
http://www.youtube.com/watch?v=YJoQYgoPgVQ&feature=related

Chapter 3 basic operation
http://www.youtube.com/watch?v=hjlI_NmYVpQ&feature=related



Creating TAP Files
http://tapes.c64.no/index.php

For now, this article won't go into deep details about the transfer process, since this is documented in other places. However The recommended way to transfer tapes is using a real C64 datasette. For PC-users, this is best done with mtap by Markus Brenner. You can download the application at his web site at http://markus.brenner.de. Another pretty common way to transfer tapes, is through sampling it with a sound card using a HiFi-tapedeck. DON'T DO IT!!!! It's a TERRIBLE way of dumping tapes, and the error-rate is VERY high. It's quite high even using a real datasette.

Okay, you have everything you need now (I assume that you've read the instructions found in the mtap-archive found at Markus Brenner's site), and if you follow these simple, but time-consuming steps, you will soon be making the best TAP-files that can be made.

Another pretty common way to transfer tapes, is through sampling it with a sound card using a HiFi-tapedeck. DON'T DO IT!!!

1. Dump the same title several times.

Yes, it IS time-consuming, but it is the only way to make sure you get all data is pulled out from the tape. Often when reading these old treasures, a few pulses may be misread for different reasons. Especially on lower quality tapes. This is critical, because you may not always get any clues that the tape infant WAS misread. Even if some pulses are wrong, it may load and appear to be working. Some loaders have implemented checksums. A checksum is a value that is created by adding the read bytes into a sum. At the end of the file, the final sum is compared to the sum that is expected by the loader. These loaders aren't as demanding, but they're not 100% foolproof either.

The positive thing is that noisy, worn out tapes, USUALLY produces random errors. Using a TAP-scanner and cleaner (in this text, I'll be referring to my good friend Stewart Wilson's FinalTAP, since it's probably the best TAP-tool released at the time being), we can detect a lot of things. We can detect loadertype, if a checksum is present, and if it's OK. FinalTAP also creates crc32 values of the data. Both single files and the entire data. This is very useful since we then can compare different dumps of the same tape.

As a rule of thumb, you should have at least two identical dumps before you can assume the TAP is OK. I recommend at least three dumps for loaders not containing their own checksum since they are more vulnerable. It's not likely that you have two identical TAPs which are faulty, but you can never be 100% sure. It's better to do one extra dump just to make sure the data is OK. If you can get the same title from several sources (assuming the other sources has the same version as you), that's even better. If two dumps from two different sources, even different sides of a tape turns out identical, you're 99,9% sure that your dump is completely error free.

Sometimes, the content from two different sources may vary slightly. This could be very confusing for the inexperienced, and also sometimes for the more experienced TAP-maker. That doesn't HAVE to mean it is an error (unless it fails to load, or an internal checksum indicates so). It's actually common with very small differences in "unused memory" in certain loaders (Novaload being an example of this). The software companies had written the game including random garbage bytes present in RAM at the time and those bytes may differ two different recordings, even on the same tape. This is unfortunately something we cannot do anything about, but I would advise you to try locating the differences to see if it can be any harm. An article on this subject will come later.

2. Time to clean up!

Well, you now have a functional tape? Congratulations. Now we want to clean it up. As we've mentioned, old tapes are noisy and the need to be freshened up a little. FinalTAP also has functions for that. Actually, that is FinalTAP's main purpose. By pressing the "optimize"-button, FinalTAP flattens out the signals of recognised files and removes any noise that it may detect. This has at least two advantages. You get files that are excellent as master tapes for writing back to real tapes again (these tapes will actually turn out cleaner and nicer than the originals you have), and the files will compress much better, which is nice if you want to share your tapes on the net.

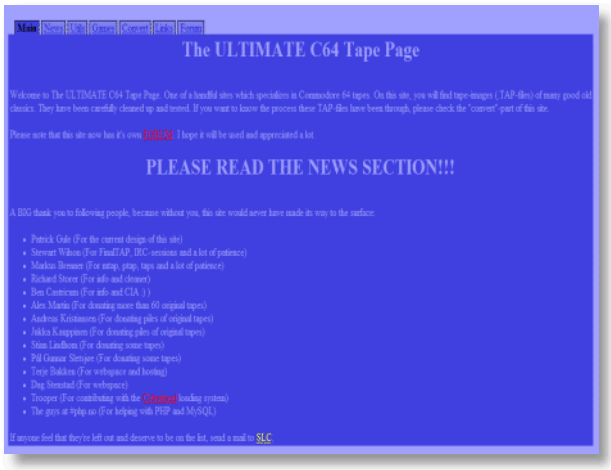
Due to the way FinalTAP works, it may happen that not all noise is being removed entirely. This is because it does NOT touch anything it doesn't recognise. This is your guarantee that FinalTAP won't damage your TAP-image. In these cases, manual hex-editing is necessary if you want a perfect TAP. This is not recommended for inexperienced TAP-makers, because you could end up damaging the TAP if you don't know exactly what you are doing. The leftovers from FinalTAP will not harm the tape in any way, and the tape should still load fine on a real C64.

You should be aware of the fact that FinalTAP does not support all the different loaders out there, and probably never will, but it supports all of the most common loaders. You should also make sure that 100% of the TAP is detected. In some cases where some noise hasn't been removed, it will say it detected 99% after optimizing/cleaning. An unprocessed TAP may have down to 97% detected (This is the lowest value I've found myself so far).

3. ALWAYS save ALL your raw dumps

This is very important, because you can always clean a TAP, but never go back IF the process went wrong. The raw files can also be useful for research and developing utilities for those into that.

I hope by writing this article I've managed to give you a clue on how to succeed in making the best possible TAP-files for the community. If there's something you feel is not answered here, or you need some other help, feel free to contact me



Interview with Tom Roger (Tapes 64)

http://tapes.c64.no/index.php

Commodore Free
Please introduce yourself to our readers

Tom Roger Skauen
Hello. My name is Tom Roger Skauen and I was born in 1979, which makes me 29 at the time of this interview. The Commodore 64 have always had a special place in my heart, ever I received my first unit, sometime in either 1984 or 1985, I don't remember exactly. I should see if I can find the invoice at my father's place sometime as it would be nice to know when this madness really started. Apart from Commodore 64-stuff, I do some hobby based programming on PC, compose some music every now and then, and.. nothing really in particular. If anyone really wants to know anything about me, just contact me. I shouldn't be hard to find, and friendly letters are always answered, even if I may be very late sometimes :)

CF. How did you first become involved with Commodore machines

TRS. My mother took part in some sort of computer school back in the early/mid 80s, so we bought a Commodore 64, mainly for her to write programs in BASIC at home, which was the preferred language at that particular school. Being a kid that loved any electronic equipment, I of course fell in love with the Commodore 64 immediately.

CF. What was the first machine you owned

TRS. That was the Commodore 64.

CF. Can you explain why games were distributed on Tapes

TRS. This seems to be a mainly European phenomenon. Probably because it was a cheap way of distributing games, and cassette players for Commodore 64 were way cheaper than a disk drive so virtually every C64 user owned one. In USA, there were almost no tape games at all. I'm afraid I cannot provide any real explanation for why this trend never hit Europe, but probably because disk drives were more expensive in Europe? Hard to say.

CF. Do people still uses tapes on Commodore machines

TRS. Yes. For different reasons. Some people (like me) do it because they still find an odd pleasure in watching the loading screens and listening to the music a while before the game play can begin. Some because they prefer to play originals (also like me), and tapes are far easier to duplicate than disks. There are also people who own a C64 that they play some games on every now and then they still don't even own a disk drive. So yes, we can safely say that tapes are still being used on Commodore machines, at least the C64.

CF. What was the websites motivation

TRS. Not sure, really. It started out like a small project that myself and a guy from Australia did. Our goal was to just dump our collections and share them with everyone, but at some point, more people got involved and the archive kinda exploded. At one point it became too much for me to handle alone, so I let Peepo do the work for a while, and never really returned to it myself, sadly. Even if the site is rather idle, I've not stopped working with tapes, so stay tuned...

CF. Is the site still active and currently maintained

TRS. It is not currently maintained, but all the content is still there and there's still lots of downloads. At some point, something might happen, but I don't want to say anything about what and when, because I do not want to give out promises I don't know I will be able to keep. But there are plans, if it's of any comfort.

CF. I still prefer to load games via the tape format, as the suspense and loading music and title screen for me add to the game, sometimes playing the game is actually an anticlimax would you like to comment

Store them in a room with no direct exposure to sun, and preferably in a room where temperature doesn't vary too

TRS. I totally agree. There are quite a few games with rather cool loading tunes and/or loading gfx, but games themselves being utter crap. In some cases, like Rambo, it's working out very well. The only annoying part with Rambo is that the game itself is so short. But the loading tune and loading picture really builds up expectations nicely, and the game itself is very well made and looks good. Platoon is another game where it really adds to the game, not to mention some of the classics like Last Ninja 1 and 2. Last time I played Last Ninja 2 was from tape, and I really enjoyed it all.

CF. So some countries they just had disk drives and didn't bother with tapes, do you think they lost out or gained:-)

TRS. Being the tape lunatic I am, of course I think they lost out :) But I'm quite sure that these people think that WE were the real losers, people who had to wait for ages to load a game, and at the time while C64 was still hot, they were probably right. Even if I fancied loading the few original games I owned myself back in the days, I'm quite sure I'd happily trade that away for much faster loading. I got a disk drive in 1989, and I have to admit that I did not care much for tapes after that, until the nostalgia caught up with me in the end of the 90s.

CF. How can people look after taped games

TRS. Store them in a room with no direct exposure to sun, and preferably in a room where temperature doesn't vary too much. Sudden changes in temperature is very bad for tapes. A cool and dry room is preferred, but not everyone has such a room available. In general, keep them away from as much excessive heat, differing temperatures and moisture as possible and they will probably last for quite some time still. I'm amazed to see that I have tapes from early 80s that don't show any signs of wear or ageing at all.

CF. What is the best method to align tape heads for perfect copies, and to ensure the games still load correctly

TRS. By using a tool that displays the signals, such as Recorder Justage. Align by using a selection of different games from different publishers. When you have an alignment that reads them all good, there shouldn't be much problems. In some cases it may be very difficult to align the tape head to read both sides of a tape perfectly. This is probably not a defect, but something distributors did to improve the chances of loading side 2 if your cassette player was too misaligned to read side 1. In case of aligning a deck for dumping, you should probably check every tape for optimal results and do small adjustments if you can gain anything from it.

CF. I presume that motors wear out on and the bands connecting the motors to the drive mechanism will wear out, does this affect performance of the tapes

TRS. I'm not really sure. The only problems I've had myself is that heads are worn out and tapes that gets "stuck" because they're too moist, and motor isn't able to pull it. (A so called sticky tape will be slowed down by anything that touches it, and the heads will slow it down the most, sometimes even stop it completely).

CF. Also with the wear and tear does this make archiving difficult

TRS. What makes archiving difficult is mostly a tape that hasn't been cared for. Apart from that we haven't run into many problems on wear and tear issues. Some tapes are more prone to sucking up moisture though. I've seen this on many Gremlin Graphics tapes and some Ocean tapes. In many cases there are tricks to get the content dumped anyway, but sometimes I've sadly had to give up.

CF. how long do you think tapes will last before they degrade beyond use

TRS. To be honest I have Absolutely no idea. But if stored properly, there is a fair chance that tapes may actually outlive the hardware from what I've seen today.

I am very well aware of the DC2N-project, having one of the prototypes in action with excellent results so far.

CF. I guess archiving is important, is this the reason for the website

TRS. The reason for the website was basically because I wanted something to do and to organize my stuff, and because I also wanted to share my work with others.

CF. Is archiving the covers a goal of the website

TRS. Maybe not so much for the site in its current form, but scanning covers is an important part of archiving and will also be done, yes.

CF. Are all the TAP files on the website given from users or have you compiled these yourself

TRS. All .tap-files on the site were made by a small group of people. They are all verified and cleaned by either me or Peepo. Nothing is collected from other websites.

CF. If our reader has a selection of Tape files how can he send them to you

TRS. He should make sure he has preferably two dumps of each tape side, then contact me at slc@c64.no.

CF. Do you know about the DC2N project and do you plan to archive tapes in this format as well as TAP files

TRS. I am very well aware of the DC2N-project, having one of the prototypes in action with excellent results so far. I have dumped about 90% of my own tape collection with the DC2N, but even if I keep the DC2N-files, they will probably not be spread. There's really no need for that, they are kept only as a raw source of the tape in case it should be necessary one day, and uploaded to c64tapes.org which is another tape project, but so far more focused on archiving than downloads. A site absolutely worth taking a look at.

CF. What is the best maintenance to perform on a Datasette

TRS. No idea. Fix it when it breaks down, and leave it alone when it's actually working =)

CF. How would you align the heads? can you explain this, and why you would need to do such a process

TRS. This is already answered to some extent. The only real way to do this is by using an alignment tool, and it's needed whenever your cassette player starts choking on about any original you feed it with. Over time, the head MIGHT drift a little, but if it has never been tampered with chances are you won't need to now either. In case you do, use a proper alignment tool that displays the signals graphically. There are really no other good ways to do this.

CF. Because tapes were slow people invented "fast loading systems" do you know how many of these systems exist and can you explain briefly how they worked?

TRS. There must be hundreds of different systems out there, but many are based on Paul Hughes' Freeload. When Paul Hughes examined some of these Freeload-clones, he found that they were line-by-line-identical with his own work. For how they work... There are two reasons why the turbo loaders are so much faster than the standard ROM loader. The ROM loader isn't really as slow as it seems, but all data is stored twice. In addition to this, one byte is represented by 20 pulses. One pair for each bit, then one pair for a parity bit and one pair to decide if it's done loading, or if there's more. So technically, all data is stored 4 times + some extra overhead. There's no wonder why this loader is so slow. In addition, turbo loaders usually has shorter pulses than the ROM loader (some have longer, but still load faster). Turbo loaders mostly also use two pulse lengths (one length for a 0-bit and the other length for a 1-bit). The standard ROM loader uses three different lengths. Some turbo loaders also uses more than 3 pulse lengths, but this is very very rare. This is a HUGE topic, so I'm not going to go into more details here.

CF. Utilities and information are listed on the website, about creating tapes, do you think archiving is important

TRS. Absolutely. This is an important piece of computer history, and everyone involved in this in one way or another is doing a really important job. We are at least trying to do our best to create a archive for enthusiasts who DO appreciate this, and there are quite a few of them out there judging from the download statistics from tapes.c64.no.

CF. Do you have any question you wished I had asked but didn`t

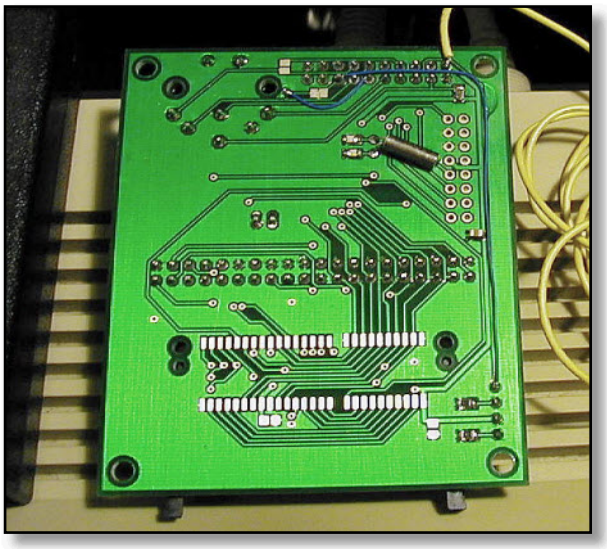
TRS. Not really, but I'd like an opportunity to thank all enthusiasts and people out there who has an interest in this. In fear of forgetting someone, I just want to also thank everyone I'm working with on various tape projects. I really appreciate the existence of this community and I'm very happy to be a part of it.

THE BRAIN INNOVATIONS "MICRO IEC"

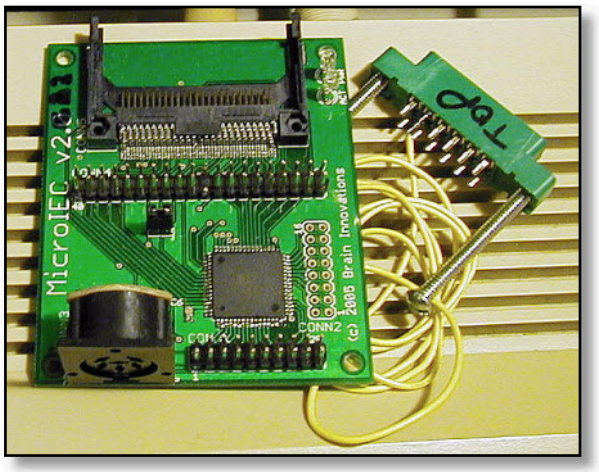
(AKA: uIEC)
review by Larry Anderson
http://www.portcommodore.com/
Updated 09/16/2008

SUMMARY
WOW! If you've been kicking yourself for not getting a CMD HD - you will be kicking yourself if you don't get a uIEC (even if you have a CMD HD), its compact, energy efficient, stores a ton of stuff, silent, and cards/drives can be plugged into a PC to transfer files without special programs, and it is very affordable - just can't get much better than that.

MY PAST & LEADING UP TO THE UIEC
I have followed and/or bought the latest in C= storage, partly because I had ran a BBS for 15 of those years. Each time a 'better' drive came out that I could afford, I had to get it. I remember the 1541 (big expense back in 1984 for me) later I had a pair of 1541s replaced by the MSD-SD2 (a dual drive, even had a PET IEEE-488 interface!) then I jumped to a 1571 with the extra storage capacity on one disk. One of the biggest jumps was to the 1581, a whole 800k on a disk! add into those RAM units such as the Commodore REU and RAMLink. Then finally I though I had reached the pinnacle with a CMD-HardDdrive (My CMD 20 megabyte drive cost a whopping $419 new back in '94) Problem with all those drives; was as I started using other systems for getting files; it was really hard to move data to and from those other systems to the 64 BBS. Usually this involved 'calling' the BBS or 64 w/terminal over modems or a null-modem connection from the Amiga, Mac or Linux computer and uploading the files, while it worked it was really darned slow and cumbersome and meant stringing extra wires here and there.



A few years ago non-volatile solid state storage started to come out, notably the Micro MMC, which had promise in that you could put files on the card directly from a PC, but unlike all the other drives did not offer a 'DOS' for the Commodore-64 to easily access the files (just a specialized boot loader menu to launch game images 'ROMs') Not too useful if you have a BBS, or do other work requiring storage on the 64. A little after that Jim Brain had announced his work on the uIEC, a device to use solid state Compact Flash cards on the 64 like you did with a hard drive on the CMD HD - meaning no need



for special programs in to the 64, plug-in to the IEC port and go. The big news was the possibility to seamlessly access files on the card on a PC as well, using the popular FAT file systems so popular with these cards. Thanks to our patience, support and Jim Brain's diligence a new contender, the uIEC is now a reality. Though there are now other similar storage devices: 1541-III, Ultimate 1541, etc. Unlike those, this one is readily available in the US and the price is very reasonable,

WHAT IS THE UIEC?
Generally speaking, it is a very compact disk drive for the C64, C128, VIC-20, C16, Plus/4 with no moving parts and lots of storage capacity... but it is way more versatile.

There are two versions of the uIEC, one with just a compact flash card slot, and a larger one that also includes an IDE drive connector, The uIEC CF/IDE is the one I am using. The smaller uIEC CF does not have traditional IEC connector as it is intended for mounting/wiring inside a commodore.

HARDWARE FEATURES
- Compact size - the large CF/IDE model is only a modest 3" x 2.5" - About 4" long w/CF card inserted, slightly larger than a C64 game cartridge!
- Uses 5v DC power - which can be tapped from the C64 (this one was provided with a cassette port connector to get power)
- Supports IDE Drives and Compact Flash cards (uIEC supports IDE/IDE, IDE/CF, or CF/CF if you have an IDE->CF adapter for the second card. uIEC/CF supports single CF card)
- One IEC serial port (Commodore 64 style disk interface) interfacing for a 2nd port is provided, but not wired in.
- 2 LEDs (power & activity)
- 20 pin header for special switches or other interfacing
- Supports FAT12/16/32 partitions of any legal size though support for >137GB drives needs more testing.

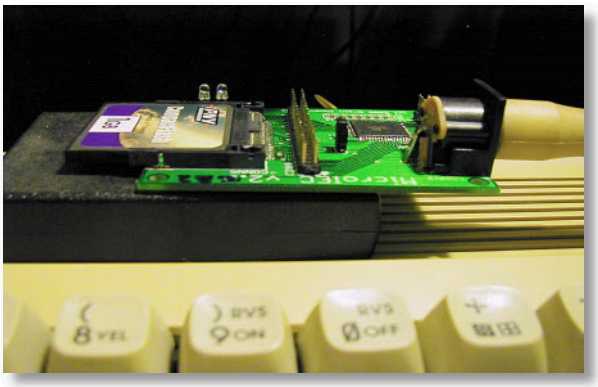
SOFTWARE FEATURES
- Uses SD2IEC DOS which is a popular DOS for a number of solid state drives for the Commodore (MMC2IEC, SD2IEC, and uIEC).
- Stores data on the card using the popular FAT file system (no special format all CF cards are already formatted for this), this also

means data on the CF is readily accessible on a PC without any special access software.
- The uIEC DOS can access files on the card directly or through many popular disk/image formats (i.e. it can use a .d64 image as if it were converted to a disk) others include P00.
- Support for some popular fast loaders Turbo Disk, Final Cartridge III, JiffyDOS) and at least plays well with many others like Action Replay and doesn't seem to crash when using Super Snapshot, but it doesn't speedload either)
- Files/disk images can be stored into sub directories so you can organize your content if you have lots of stuff.
- Pretty easy use and navigation when using a DOS wedge (like JiffyDOS' @ commands)
- Supports FAT Long filenames
- Transparent support for PRG/SEQ/USR file extensions, with REL support planned.
- Supports partition-less cards/drives, or up to 4 primary partitions or 3 primary and 12 extended partitions. (Email Jim if you have more than 12 extended partitions on a drive.)
- Supports read and write of D64 images.
- Block level disk access supported on D64 images
- Most CBM DOS commands (Scratch, Initialize, Rename, etc.) supported.
- CBM general config commands (U0, U+, U-, UI, U9, U:, UJ) supported.
- CBM block level commands (B-R, B-W, UA, U1, U2, UB) supported when in D64 image.
- CMD-style partition ($=P) support
- CMD-style subdirectory (MD,CD,RD) support.
- CMD DOS Commands (G-P, G-[shift-P]) support.
- Long form CMD directories ($=T:*, $=T:*=L) supported
- 1581-style/CMD-FD/HD-style wildcard matching supported ($:JIM*RAIN)
- JiffyDOS fast loader equipped (PAL and NTSC support). Can be enabled or disabled via DOS command.

Transparent support for P00/S00/U00 files, with R00 support planned.

Pricing for the unit are (shipping additional):
uIEC : $75.00 - the CF/IDE model (unit reviewed here) plus shipping
uIEC/CF: $50.00 - the CF only model (intended for internal C64 mounting) http://www.jbrain.com/vicug/gallery/uIEC?page=1 see # 1882, 1880, & 1875 Production just starting so there may be a waiting list. From: Brain Innovations, in Iowa Email for availability/order details: brain [at] jbrain.com

APPEARANCE & HOOKING UP
The unit is a tidy little package, a PCB with just a few tiny circuits on it - very easy to handle (you can readily grasp the long edges of the board wile plugging and unplugging the card and cables. There are four screw mounting holes on the edges With the IEC connector it may be a tight fit in the 64, the standard (v1) model will not have the IDE connector which makes it quite smaller and is better suited for internal mounting in a 64. Being fresh connectors the fit is tight, so one must be careful plugging and unplugging the serial cable. Also the guides on the CF slot can lead you off when inserting the card (this is where making a case with a good CF guide could help.



For starters I use my PC to put a few files on the CF card, some D64's, a couple zipped d64s, and some .prg files (straight to the file system - not inside an image), and a few folders as well. Plugging in the cassette connector was a tad tough it was just a cassette connector without housing (using a couple 6/32" machine screws made it easier to grip without bothering the wire.) My unit had a connector different then the final version Jim is going to produce so those should be easier to handle. If you have an SX64 or want to mount the uIEC inside a c64 case you could tap power from other points (like the joystick port)

POWER ON
At first I though something was wrong after I turned on the machine, the LEDs lit and it did nothing, commands didn't work - after a couple checks I discovered the boot process is a bit longer then I imagined, about 30 seconds, this is an issue with using a CF card, which Jim is working to resolve. Though during this time it is so eerily silent, and so small something you have to get used to no squeak of the 1541 or whine of the HD spinning.

Software Updates
Another part of the bootup is checking for a EEPROM update file, unlike many devices uIEC is flash-upgradable so if there are some bug fixes or improvements, the device can be updated without a trip back to the factory or buying a replacement.

OPERATION
Initially I discovered the unit was using device 10, so I at first was only able to play with single file programs or ones that were load device aware... After reading the documentation http://snowcat.de/cgi-bin/gitweb.cgip=sd2iec.git;a=blob;f=README;hb=HEAD I found how to set the device number and save the settings in the uIEC so I could use it as device 8. (more on that later) Listing directories went as expected LOAD"$",8 (or @$ for DOS wedge users.) displayed the contents of the card at the root level loading up one of the PRG files was quick, and JiffyDOS just works so far. Referring to the documentation enter sub directories or mounting .D64 files uses the CD command, i.e. (without a wedge)

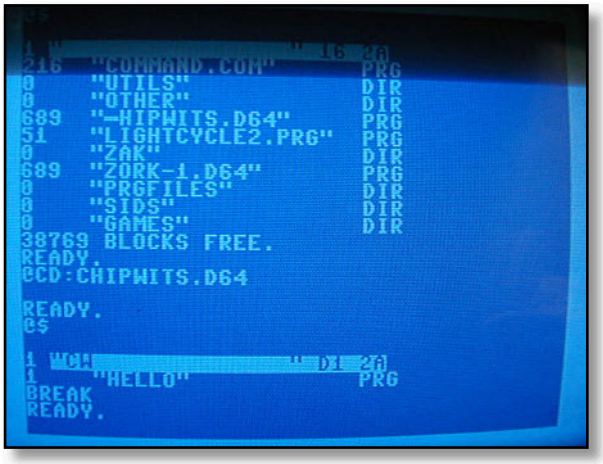
OPEN 15,8,15,"CD:GAMEDISK.D64":close 15

or for people using a wedge:
@CD:GAMEDISK.D64

gets you into the GAMEDISK.64 image, there it then looks like you are in the 1541. To get back out of an image or back a directory (without resetting the 64 & the unit) is done by doing CD with a left-arrow symbol (<- Above the CONTROL key on the 64/128).

@CD:<- (left arrow symbol)

Resetting the computer returns the uIEC to power-on state which is normally the root directory of the card. You can also set up partitions on your CF or hard disk and use the CMD style partition commands (CP) to jump between partitions. Running single files works without a hitch, game disks without software fastloaders also work fine. programs with fastloaders (stereo sidplayer 10 for example) need to have the fastloader disabled or bypassed (this may be near impossible for some programs without acquiring a cracked version). note: this result is pretty much the norm on non-Commodore 1541 drives. Using some cracked single disk games was also rewarding. One program I usually have fits with loading, Editor Assembler, also worked fine with uiec.



One note:
when you are not in a d64 image, in the FAT file system, upper and lower case letters are treated as the same thing unlike Commodore DOS (My game, MYGAME, and my game, would all be thought of as the same file) which can be a convenience (to load files without having to deal with case issues) or an unexpected annoyance (if there is a need for similar file names with difference cases - though I don't think I've run into any myself) In d64 mode the file case sensitivity operates as expected.

But What About Multiple Disk Games?
Yes! you can work with multiple disk images without having to go back to BASIC. You will need to have a disk change button or two connected to the uiec (there are pins assigned for that on the board: 15 and 16 of connector 1, for forward and back, connect the other lead of each momentary-on switch to pin 1) First, after copying the disk images into the card or drive, you make a file called a "swap list" which is a text file containing just a list of the image file names (and paths) you need in the "set" (example a swaplist file called PACGUYLIST or whatever you wish to name it) the file includes these lines representing the names of the disk files in the disk set:

PACGUYDISK1.D64
PACGUYDISK2.D64
PACGUYDISK3.D64

Then instead of CDing to the first d64, you issue the command:

@XS;PACGUYLIST

This command opens the file and automatically selects the first disk image on the list. Once you have the game going and it asks for the next disk you push your diskchange switch and uiec queues up the next on the list (or previous if you have a 'back' switch) If you go to the end of the disk list it starts again at the beginning and visa versa.

TRYING OUT UIEC WITH AN IDE DISK DRIVE
On the CF/IDE version, besides the CF slot there is a header to plug in a PC drive. CMD HD the drives used SCSI interfaced hard disks which can be troublesome to locate a replacement. On the uIEC the hard drive interface is IDE (aka ATA, not serial ATA). If you do use a hard drive you will need to use a separate power supply to power the drive. You can get a simple USB to IDE kit which includes a compatible power pack pretty reasonably on eBay, this is what I used for my testing, or if you get a big external USB drive enclosure you could put the uIEC in it along with the drive. (and not use the USB part of the unit) I quickly formatted and set up an old drive with some d64s on the PC and then went to the 64. Hooking up the drive you need to make sure the red line on the cable is at pin 1 of the pins on the uIEC (there is no plastic guide on the circuit board like PC motherboards have, but it does indicate pin 1 on the PCB text.) After that plug in the drive power and turn on the 64, and very quickly the uIEC is ready! Only a second or two of wait at most instead of 30 with CF, sure seems a bit zippier, but also a lot more bulk with that big drive (relatively) and cables to deal with. Attempting to use the IDE and the CF at the same time only resulted in drive errors (should work, will have to check, maybe it's a master/slave select issue?) From Jim Brain's Posting you should be able to use two hard drives (master slave, like on the PC) Some of the planned expansion for the IDE part is uIEC support for reading CD-ROMs which will be very nice.

A LITTLE BIT ON CONFIGURATION
As mentioned you may at times need to set uIEC as device 8 so you can load some d64 images of games OK. To get to device 8 you use the command (in BASIC): OPEN 15,8,15,"u0>"+CHR$(8):CLOSE 15 - To set to 8; this is similar to how you do a soft device change on the 1571 or later drive. to save the device settings it so it remembers when reset (and other settings you may wish), just enter: @XW now every time it powers up it will start with the settings you want. Pretty darn easy and no programs needed. Though I found it took a couple tries to get the settings to catch. But once they were set everything works as expected.

MORE TECHNICAL EXPLORATION
Some of this is more a sysop view of the uIEC, so bear with me if you get lost, this is for the other techies and Sysops out there.

DRIVE IDENTIFICATION
When trying CMD's Fcopy, the uIEC is identified as a 1541, thus keeping Fcopy from accessing partitions or subdirectories. Jim is working on the best way to provide compatibility with such tools.

PARTITIONS AND SUB DIRECTORIES
Partition navigation works like a CMD in that if you partition the drive the partitions are 'drives' to the device, partition 0 is the current partition, etc. Currently there is no partitioning software for the uIEC on the C64, you can use PC utilities to do the job presently. A C64 partitioner/formatter is in development. One thing I never got into was subdirectories on my CMD Drive, and from what Jim reported they work similar on the uIEC. To move things in and out of

disk image file you need to be 'in' the image but you then you can access outside files; but only if they are from a separate partition. There is work on making access of image contents more flexible but it is a whole lot better then just access only within an image. This could make development of a C64 based .d64 image utility much easier! 15 channels can be opened at the same time. So no problem opening multiple files.

NATIVE MODE
The "Native mode" of the drive is the FAT file system which technically is different than what many are used to with 1541s and CMD drives. Part of which is in native mode you don't have that 16MB content size limit. But then again, you don't have tracks and sectors not in an image, so disk editors, directory editors, and utilities that may rely on direct access to disk blocks will fail in native spaces (Lynx is an example) For utilities that rely on such access you can use disk images. Also being FAT it goes by FAT file naming conventions, which means letters are not case sensitive and the characters / \ ? : , are in the list of no-no characters in native mode file names, of all those I think / would be the most common. Someone informed me that there is a file extension mode called x00 mode which should allow for using all the characters in Commodore file names, to initiate you would use the command @XE2, I haven't tried out this mode yet. Relative file support in native mode is operational but in alpha as it has not been tested in length as of yet. As more BBS guys get their hands on these I'm sure they will put them through the paces.

GEOS SUPPORT
I don't use GEOS/Wheels/Wings so I can't really comment on those. From what Jim said he needs help from a GEOS guru to figure out the GEOS fastloader to make it compatible.

REALLY TECHNICAL
For more information or to help with the open source sd2iec DOS refer to the sd2iec project home page. http://snowcat.de/cgi-bin/gitweb.cgi?p=sd2iec.git;a=summary

CONCLUSIONS
Overall WOW! This is the best thing I've seen since the CMD HD, plug it in and it works - no special software, minimal conflicts. But even way better as it works "cross-platform" its conveniently small, and very affordable. For sysops, I think it is generally OK and getting better (unless you don't already have mass-storage, then it's really good), at present it would make a great U/D, text file and/or programs drive(s), Though the lack of some of those characters (which may not be an issue, see "NATIVE MODE" above) in the file name will be an issue to resolve on some BBSs (i.e. Image BBS uses / in some of its system and program files) so you may have to do some BASIC updates to the system and implement a message/group/UD name filter to reach compatibility. Lastly REL file usability will be a factor if it is to become a total BBS drive (at least for the BBSs that rely on REL files.).

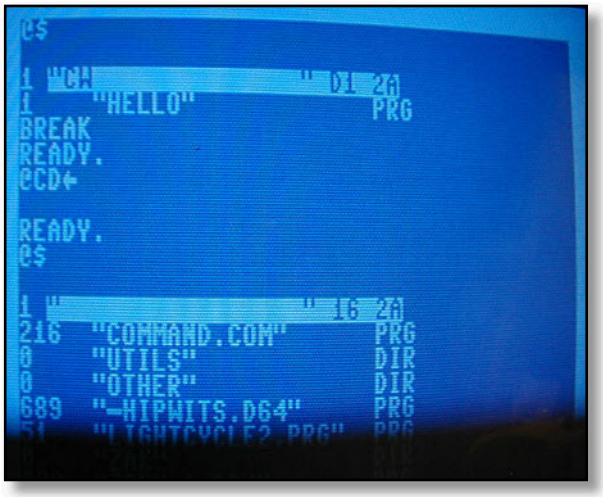
Detractions
- The 30 second start up when using CF cards - this should be fixed soon to just a second or two.
- The cassette/power wire is a necessary thing, though I wish it weren't

Always Room for Improvement
None of these would keep me from recommending anyone to get the uIEC, but these could make it better (at least to me):

- A utilities disk it would be great to have a 64 program to formant and/or partition CF cards and hard drives.
- A menu/navigation program - so you can quickly navigate all the directories you will have on this thing, it's is something that someone who got a UIEC is probably is working on right now.
- A "swap 8 button" to quickly (temporarily) swap a device 8 drive with whatever device # the uIEC is at the push of a button (I use that feature a lot on the CMD HD)
- A two IEC port version (with the 5V lead I need to keep it close to the 64 or route an IEC cable back from the other drives and end the chain with the uIEC, a ready made IEC pass-through port would be nice in the next version.
- An LCD readout (not really necessary, but it would boost up the cool factor, or better yet if you could navigate directories using it... that's probably asking for too much at that price point though...)

ADDENDUM:
Jim Brain has announced there is going to be an even smaller version (!) of the uIEC CF, using an SD card instead of a compact flash. This will be replacing the UIEC CF version. Thanks to these people for corrections: Ingo Korb Greg King

COMMODORE FREE
I would like to thank Larry for Permission to reprint his review; an updated version can be found here http://www.portcommodore.com/uiecreview.php



MossyCon5

By Lord Ronin from Q-Link

April the 5th 2009 is the date slated for MossyCon5. A small get together of C= users from three or more states. The first three years were held at Mohr Realties Games, a small little game {RPG} shop that also promotes the Commodore line.

OK in truth it is the headquarters of the Anything Commodore Users Group CBM Reg Number 447. This is a small shop and a small amount of people arrived; but after the amount of demos that have grown so have the attendees. There wasn't enough room for the pipe smoke, so for MossyCon4 in 2008, we moved to a Pizza place, and they gave us the space for free! OK some promo adverts in the press releases helped sweeten the deal.

So we decided to hold MossyCon 5 at the same place. This year we are expecting: more music work from Steve Jones, a Display of C= items from Robert Bernardo of the Fresno California Users Group, along with an Amiga One with the new OS 4.1 installed. Perhaps a demo of the Amiga 4000 Power Tower. Well that's on the assumption we can get it set up right with the monitor.
The first idea came from Steve Jones of Prophet 64 fame, He suggested that we do a game challenge: this idea grew from using the Adventure Construction Set and a 30 minute adventure, Into a gigantic hell storm. First the ACS is still being used, But the time limit has been removed, then the idea of one theme for a game was deleted. As ACS will create Sword & Sorcery, Spy/Mystery (great for horror too) Along with Sci-Fi.

That idea expanded; some people like text adventures, so adventure writer was added to the mix. Then some don't like ACS and prefer Dungeon Creator found in Load Star by David Caruso II, then some other game makers that I do and others that I don't have in my personal collection where tossed into the works.

The end result is that we hope that there will be many games produced for the micro con. The Idea is that we want to have some 1541 disks of the games, but we also we want to have them in a playable format from a CD This is a task that I hope to have completed at the time of MossyCon 5. One other part has also introduced, I admit it was by myself the dice gamer. Making adventures for dice and paper games, using Geos as the tool for writing and for maps.

Don't get too scared, I have found a way using post print 3.8 and GeoDos to put the files onto disks readable from a Linux system, then on the Linux system convert the files to a .pdf format. This PDF file works well on the reader that myself and several others users have for testing. In fact it is how I made the 20 page Cyber Space edition of our newsletter.

Right now there are three known games in ACS, half of one in Dungeon Creator and assorted bits in other game makers, along with a nearly finished adventure for the out of print High Colonies Sci-Fi game. Taking up about an entire 1541 disk side, in uncompressed form. Now then, as we have readers throughout the world, and most won't be attending MossyCon5, it would be nice to have this available as well, but I don't think there is space, I don't need the competition at the bar gang [VBG].

Anyone in the C= world can contact me about submitting their work, I'll go over in detail the simplistic rules for the game making, then when completed the work is sent to me and through the prayers to the great C= Headed goddess. Your work will be on the CD at MossyCon5. All credit is yours and the CD isn't for sale either, a complete labour of love. So if this is of interest to you. Feel Free to contact me .... lordronin@vcsweb.com. I'll get back to you, though a bit slow at times in the e-mail these days.

In the Beginning 11

By Lord Ronin from Q-Link

Having gone through how we make more than a one sprite active, there is a bit more to it than we have covered last session. We will move ahead to some vague things for me, then back to multiple sprites. As well as fixing the problem of our sprite not going all the way across the screen.

Down to line #12 from our programme. The line Says poke 2042,13. Right we know poke is put something into the computer. 2042 is a location. Looking back at that chart, the one that goes from 2040 to 2047, for the 8 sprite memory locations. We can see that 2042 is the number for sprite #2. That part is pretty easy, I lose it at the next part. That ,13 part. The manual says "13th area of memory." and continues with the information that a sprite takes up 63 sections of memory. Said just about that way. Yeah I get lost here. OK it helps a little bit with the following. As they talk about doing all of that three series adding up stuff to make the data for the sprite. 21 rows, three series each row is 63. Got that part figured out. What's next is that each one of those series. I mean the 128 64 32 16 8 4 2 1 series on each row is 1 byte of computer memory. therefore each row is 3 bytes of computer memory. So an entire sprite is 63 bytes of computer memory. Impressive that we can do so much with so little space. Even at this lamer beginner wanna-be level of a novice. Thought that the,13 part was the section under the registers in the last instalment. Where the book says 2-15 are the pairs for sprites 1-7. But 13 doesn't fit any pairing for sprite #2. Can't help you, my members or myself on this part.

line 20, we have a for and next loop, In that we have 0-62, remember 0 starts so there is really 63 units., hmmm that is the same number of things you put in those data statements. 63 for the amount of numbers inserted in the data statement, 63 for the bytes of memory used for a sprite. Here too we have the read q part. Read is the command to go look at the data. Going out on a limb here in my understanding. Q then would be a variable for that information, as the next part of the like poke 832+n,Q. Fits as a 63 time around variable that is read and poked into the memory of the machine. Again the book says this loop is poked into the 13th block of memory. Starting at location 832. Again I have no idea of this 13th block and why or how they picked location 832. Only that it works in the lessons here.

line 40 we are pokeing v+4,x and at line 50 we are poking v+5,x. That 4 is the controller for sprite 2 in the X co-ordinate. While the 5 is the same for the same sprite save that it is the Y coordinate. Doesn't make too much sense by itself. Remember though we have line 30 which is the for X = 0 to 200. Add that value of X to both the +4 and the +5. Then there will be a replacement across and down the screen. Making the diagonal movement. Trick here is that it is moving so fast, since the data is being read fast and the computer runs fast. You see the balloon float diagonally across the screen, this is the same sort of trick used for years to make animation. Show, move, show again, Just do it so fast the eye doesn't see the changes.

Well there is a note to look at the back of the book for the list of the registers. All 46 are listed with some very minor explanations to them, that isn't a part of this series but personally maybe that is where some of the raster stuff is done and where the controls for a light pen can be accessed. As they are listed in that area as are several sprite things, and yes that 23 and 29 part from last instalment is right for what they do with a sprite.

We may have the greatest PC ever created. But we can't do multiple objects in one memory section, So each sprite has its own 2 sets of memory sections to make it and to move it on the screen. Now they want you to try the following. Add this line to the programme...

```
25 pOv+23,4:pOv+29,4: rem expand
```

Run the program and your balloon just got taller and wider by around twice as much. Right that 23 makes it taller and the 29 makes it wider, ok and the 4 is still playing around with the 2nd sprite. Play a bit with just pokeing one or the other. But leave one of them out. See what your balloon looks like now. <groans are accepted at the sight> Note too that it still starts from the same section of the screen. Next they want you to modify some lines...

```
11 pOv+21,12
12 pO2042,13:pO2043,13
30 forx=1to190
45 pOv+6,x
55 pOv+7,190-x
```

..... Oh sorry I jacked out for a little bit and did a few of the type in things for this part....

The extra lines above did some changes, like line 45 and line 55, you note that line 11 changed the ,4 to a ,12. As we talked about earlier this will mean that sprite 2 and sprite 3 are now activated. Line 12 adds a new poke command, and that is 2043,13. Looking back on the charts, the one for the sprite locations. 2043 is the number for sprite #3. I'm still at a loss on the ,13 and where and why they use find and use it.

Off of that rant for a moment and back to line 30 there is a change in the writing of the for part. Instead of 0 to 200 it is now 1 to 190. Not sure why this is done. But it works on the screen.

At line 45 we poke in v+6,x. Line 55 we poke in v+7,190-x. That 6 & 7 must be or the new sprite. But we have a reverse of something here. 190-x is gonna make it do what?

Run the newly modified program and lets see what happens, what I had was a cyan big balloon going as it had before. now a little purple one is going in sort of the opposite direction, bottom left corner of the screen to the top right. OK we have two sprites moving on the screen at two sizes in two directions. Time to complicate things even more.

```
11 pO v+21,28
12 pO2042,13:pO2043,13:pO2044,13
25 pOv+23,12:pOv+29,12
48 pOv+8,x
58 pOv+9,100
```

Before you run this one, I'll lay odds that you can see we have added another sprite to the mix. You should see that in line 11 with the higher number, you will also see that in line 12 there is a third poke for another sprite location. This is going to be a balloon because of that ,13. This is obtaining the information from the same location as the other two sprites. At line 25 we do something a little different, we poke the expansion codes into two of the balloon sprites. Seeing that it is the same number of 12,that means it is the 4 and the 8 values, or sprites 2 & 3. 48 is a similar line, to what we have seen before, this line controls the horizontal direction. Line 58 is close to what we have seen before, understanding that the 9 will deal with this sprites vertical movement, hmm no variable or negative number here, What happens in this program?

Give it a run, you should see the balloon as before on the screen, except that both of those two balloons are now bigger. The third sprite, is also another balloon and is coloured green, this sprite went directly across the screen as far as it could. That may be what that line 58 is supposed to make it do?

"Additional Notes on Sprites" is the next part of the book, there are questions, not all of them raised are covered above.

Starting off is changing the colour of the sprite, repeating the command of v=53248 as the way to set the video. We are told that to change the colour of our first sprite, or "sprite 1" to type in pokev+40,13 to make the sprite a light green, That 40 is the register code, for sprite #1. The only problem is, that we haven't turned on sprite #1 in our program. Next part of the ,13 is the light green colour, so for your experiments, a refresher. 39-46 is for sprite colour 0-7. Pokeing v+40 is the register number for the sprite #1. If that was a poke v+39 it would be for sprite #0. 41 for sprite #2. 42 = sprite #3, 43 = sprite # 4, 44 = sprite #5, 45 = sprite #6 and finally 46 = sprite #7.

There are charts that will tell you the number between 0 and 15 for the colours, the easiest way to remember that in this part of programming. We start at the colours listed on the keys, though not all models have the second set of colours listed. Black is 1 on the keys so remember control and 1 to change to black for text and the cursor, as well as in print statements? Well here we start with 1 on the keys, but in programming we start at 0. So Black = 0. Light Grey is the 16th colour. Since we started at Black as 0 then Light Grey, the last colour number would be 15, simply stated; just subtract one from the key readings, so the white key is number 2, this would be 1 in the code list, Cyan is 4 on the key and therefore 3 in the code. Yellow is 8 on the key, so it is 7 in the code. Takes a little bit of time to be comfortable with this trick, I still at times have to do a count and subtract one, When I deal with the second set of colours; The ones that we get with the C= key.

Next they tell us that we may have noticed that the sprite didn't go all the way to the way to the right hand side of the screen. Yeah we did see that one, the reason for this is that our value for that specific register is the maximum that it can be, 255. While the screen is 320, they say dots for their illustration of a 320 dot wide screen and 255 dots for the direction.

We know it can move across the screen, as we have seen it in programmes. How is it done then? Back to that registers and description chart; register 16 is the Most Significant Bit <MSB> and it appears that it is for the X or horizontal coordinate. What we do is poke the value of the sprite into the memory at this position of 16, doing that for sprite #2, it would read pokev+16,4. That will take it past the 255 location on the screen and move it from 256 to the 320 location.

We will have the programme in just a moment, What I want to present at this point is the fact that there are only 64 spaces on the horizontal that the sprite needs to move all the way across the screen. Keeping that in mind, lets do the programme. First off this program is only for one balloon. Pretty much keep just the DATA statements from your previous balloon work, Kill off the other lines and then type in the following.

```
10 v=53248:pOv+21,4:pO2042,13
20 forn=0to62:readq:poke832+n,q:next
25 pOv+5,100
30 forx=0to255
40 pOv+4,x
50 next
60 pOv+16,4
70 forx=0to63
80 pOv+4,x
90 next
100 pOv+16,0
110 goto30
```

Right then, we have new and old stuff in here, so let's take a look at the lines.

Line 10 is a tad bit different, sure we have the turn on the chip with the variable v, then followed by the turning on the sprite and the sprite being the 2nd one <value of 4 remember>. Next we poke into the sprite our data values. What is new then? Well you have used all those commands on one line. Using that : symbol to separate different commands on one line, rather than write a mess of separate code lines.

line or 20. Has that part that reads all 63 parts of the sprite, doing that 0 to 62 bit for Q with the READ command. Line 25 has the "Y" coordinate for the sprite, here we see a 100 rather than the previous "X". Line 30 has the "X" But see that it is 0-255, line 40 we have the semi familiar poke for the second sprite and the horizontal coordinate. Note that it is "X" and that is generated in line 30 for 0 to 255.

Pretty much what we have done already, line 50 is a next and must be for that for in line 30. line 60 we have something new, pokev+16,4. That one is for making the MSB or Most Significant Bit trip. Poked that into the v variable and that, 4 is the value for the second sprite, the one that has been working for us through out the prg.

Line 70 is the rest of what we read about just a little bit ago, Here we have for x = 0 to 63. That is the next 64 spaces on the screen to make it go all the way to the right hand side of the screen.

OK but we have to tell the sprite to do that, and that is done in line 80. Where we have pokev+4,x. Same as line 40. But here the x variable is the 0 to 63. See how that works? First the x variable is the 0 to 255, and then it becomes the 0 to 63. This takes us the rest of the way across the screen.

Final part of that piece is line 90. Where we have the next for the for-next loop. Since there isn't any variable in use past the x one, there isn't a need to tell the for next loop to do a next x. Computer has that as implied, we didn't give it any other variable to work with at that time.

I have a bit of a problem understanding the next part, perhaps I missed something or I am just too literal.

New at line 100, looks kind of familiar, here we poke v+16,0. Well we got the poke part and the v variable part, just started to understand the +16 part. You remember turning on the MSB. What the frell does that 0 mean? A big nothing? Sort of, remember back a bit when we were doing the chart stuff. Like the sprite drawings? There it was a 1 meant something was there or turned on, while a 0 meant that there was nothing there or turned off, on this line that 0 means that we are turning off that MSB part of the sprite. That allows the whole thing to start over again, with line 110 in the goto command.

I have a bit of a problem understanding the next part, perhaps I missed something or I am just too literal. But here is what they say in regards to defining multiple sprites, we may need additional blocks for the sprite data, OK I can follow that part. As I suspect that this is the beginning of how to have more than 8 sprites. Remember that 8 are all that can be active at one time; you can have others in the "wings" waiting to be used though. They then tell us that we can

use some of the "Basic's RAM by moving Basic, before typing or loading your programme type:" Get to that in a moment. I have heard about moving Basic for things like new fonts and the like, I don't know how it is done and have not done this myself, I needed to clarify that before the code line of...

POKE44,16:POKE16*256,0:NEW

This sets up to use blocks 32 through 41, these are memory locations 2048 through 4095, "To store sprite data." Right 64 bytes for each sprite, I am not sure of the block part or the 1044 memory locations that exist in the above. Suffice that it works and gives you more areas for your sprites. You may remember that 2040 is the memory for sprite 0, making sprite 8 at 2047, this starts one off at the spot right after sprite #8. Anyway feel free to play with some of the numbers in the program above and see what happens.

I Barely grasp the concepts presented on the 832. Just to say that is the place to put the first sprite you use

In a recent lesson, we played with the colour of the balloon, placement on the screen and the speed of its movement; we also made some interesting effects.

I Must tell you that I tried to gain the information on the vague parts I mentioned above, I found the information on that 832 part and the 13th block of memory, all in the Programmers Reference Guide. Editorial comment time: Great book, but not for the rank beginner, there is tonnes of information, and it's Not written at the level of the users manual. Good book to have a copy of in some form though, But the book assumes a bit more comprehension than we have at the moment.

I Barely grasp the concepts presented on the 832. Just to say that is the place to put the first sprite you use, they then go up by 64 for the different parts. Don't try it now though. Sprite #4 causes a problem in screen location, now that,13 for the 13th block of memory. Not real up on it, but can say that if you have a second sprite, and here I mean a different one than the other(s). You would use the 14th block of memory and so forth. Not the scope to do that in the users manual or in this series, in attempting to explain the higher level of sprites, Besides I have to learn more as well.

Add at this point sprites are more than we have seen here in the series and the user's book; I haven't talked about controlling them with the joystick or keyboard, nor anything about multicolour sprites. I haven't even talked about multicolour custom fonts either, none of that is in the users manual. I point it out now to show you there is a vast amount more to learn and to play with on the C= than presented here.

Now back to the book and their beginnings on binary math. We actually have been doing some of this without knowing it with those charts for the balloon and sprite locations, that on and off part. Well we aren't going to spend a lot of time on this part; I had it and failed it in college. First we have some terms to explain; well the book starts out that way in this area. BIT is the smallest part of information that the computer can store, basically something is there or not, the on/off thing. If there is something there, the value is 1. If it is empty then the value is 0. Right like those chart things earlier for the sprite creation and location stuff. The book goes into a BYTE next, this is a series of BITS, since we are an 8 bit machine; A BYTE for us is composed of 8 bits. Hmm wasn't the number of slots in that sprite row, and say isn't that the same amount of sprites that can be on the screen active?

OK now I have to add some information that isn't in the book, because you may have heard the terms in regards to the C= in the past. NIBBLE, isn't in the book, the term comes from crackers and hackers, so I was told by some of them. NIBBLE is 4 BITS, yeah half a BYTE. This is one of the early copy systems, called a nibbler. Sorry we had it before the Futurama show, but then again Matt tosses in a lot of C=64 things in that show. Back on topic; You may find in your disk collection that you have a tool called a "nibble copy" or a "nibbler", That will copy a disk at 4 bits at a time, this Broke early copy protection.

Another word that you have heard in here in the early part, and will see on your disks, is a BLOCK. Commodore measures things in BLOCKS, as I said before. We have been doing things with bits most recently; I wanted to let you know that these measurements of block and bits/bytes will be tossed around a lot, and Used by users, programmers, coders, manufacturers and more. Gets real confusing at times, when they start tossing out KiloBytes, Bytes and blocks; For the record a C= block is a 256 bit unit. Roughly 4 of them will make 1 kilobyte.

Saying all of that trivia, here is your next thing to play upon; On the regular screen, save the balloon stuff if you wish, I want you to do some simple math problems. We are going to raise a number, What I want you to do is type in ...

? 2^0 <and then press return>

Now then just cursor up and over the 0 and change it to a 1, then a 2 a 3 a 4 a 5 a 6 and finally a 7. You will see that it goes the same 1 through 128 values we have dealt with before. I just think that this is easier than making the chart in the book to show you the same smegging thing. Now there is a program to type in and I strongly suggest that you save this one.

```
5 rem binary to decimal converter
10 input"enter 8-bit binary number:";a$
12 iflen(a$)<>8then?"8 bits please...":goto10
15 tl=0:c=0
20 forx=8to1step-1:c=c+1
30 tl=tl+val(mid$(a$,c,1))*2^(x-1)
40 nextx
50 ?a$;"binary ";" = ";tl;"decimal"
60 goto10
```

Line 30 is the one that most of the group here, self included, frell up. Forgetting to do)) after the 1 and before the *. The programme will take an 8 character binary number, like 11111111 and tell you what the decimal value of it is, and for here it is 255. There are a couple new commands in this one that are lightly discussed in the book, VAL is for VALue, Gives us the actual value of the character as a numeric form. MID$, also called "mid-string", and takes a look at each character in the string, from left to right.

C variable in that string in line 30 tells the program what character to work upon as the program goes through the loop. There is a lot more to the usage of these two new commands. Not all that needed for this level of understanding. Last part of that line raises the number to a power of 2. That x-1 just keeps it in track, Starting off at 2^7 and each time through the loop dropping by one, till it reaches 2^0. Locale group lesson is to count on this thing from 0 to 255, in decimal, by typing the binary values, sounds tough I know; and it took me two hours the first time, however there is a pattern. If you have that chart at hand, the one that starts at the left with 128 and ends with 1 on the right, You can see the pattern a lot faster.

20 pO54277,9 <sets the attack & decay>
30 pO54276,17 <determines the waveform, or type of sound>
40 fort=1to300:next <duration of the sound>
50 readа <reads first number in data statement on line 110>
60 readb <reads 2nd number in data statement on line 110>
70 ifb=-1thenend <turns off at value in line 900>
80 pO54273,a:pO54272,b <pokes first data number as the hi frequency and 2nd number as lo frequency>
85 pO54276,17 <starts the note>
90 fort=1to250:next:pO54276,16 <play and stop the note>
95 fort=1to50:next <release time>
100 goto20 <loops back for new note>
110 data17,37,19,63,21,154,22,227 <musical note vales, listed in book>
120 data25,177,28,214,32,94,34,175 <Each of these pairs is one note>
900 data-1,-1 <turns off hi/lo and ends the prg>

Sound is not a something that I am not comfortable with at this time. Not because I can't read music to some degree, no because there is a lot of complicated things here for the beginner;

Page 79 starts on sound creating, the book ends at 103, starting all the appendices. Lot to cover and we aren't going to make it at all; No reason too either. If you have followed through to this to this point, you understand that there is a lot more to programming than we can cover in these instalments'. Just for Basic, let's not mention other forms of programming languages. You may have also thought that you are not interested in programming; in either case there are other sources than just my lame drivel. I'll try to remember to talk on that in the last part.

Sound is not a something that I am not comfortable with at this time. Not because I can't read music to some degree, no because there is a lot of complicated things here for the beginner; and the fact that not all of the 64 user manual examples work on the later SID chip, Like this 128Dcr I am using.

What I am going to do is lay out some type in things that did work on this C=PC. Leaving off a lot of explanations and just having you see that you can make sounds on the system. Big books cover this topic better than I or this manual can/did.

Like I said earlier on, and for sprites as well; there are programming tools that will do this work for you. One I did was just put the note on the staff and select the musical instrument, sound effects may be a bit harder for games, or not. The book tells us about the ADSR; The Attack Delay Sustain and Release of the sound, the waveform control and hi/lo frequency. First three settings are generally done just once for the programme, Hi/Lo is done for each note and waveform is the start and stop for each note. That said, the first type in thing in the book failed on the 64c and on the 128Dcr in 64 mode: But the next one worked, and they did some additives for different sounds in the instructions, OK type in the following... without my comments

5 rem musical scale
7 forl=54272to54296:pOl,0:next
10 pO54296,15 <that sets the volume to the highest level of 15>

Not a great explanation. They take a few pages to explain things, run the program and you should hear an 8 note musical scale. Once tired of that, change line 85 to read pO54276,33 and 90 to fort=1to250:next:pO54276,32. You'll get a sort of harpsichord sound.

This is just one of the three voices and a couple of the different waveforms that can be used. Scare you now with the added information that there are those that have added a second SID chip, or a cart and do this in Stereo. Most popular music player for the c= 64 is the Stereo SID player, OK one last one that worked, A sound effect to try out. For a gun shot

10v=54296:w=54276:a=54277:h=54273:l=54272
20 forx=15to0step-1:pOv,x:pOw,129:pOa,15:pOh,40:pOl,200:next
30 pOw,0:pOa,0

Continued Next Month

On the Road the Commodore Scene meeting 2008 and a visit to AmigaKit

by Robert Bernardo,
Fresno Commodore User Group, http://videocam.net.au/fcug

Another long-distance trip through England in June this time from Haywards Heath in southern England to Birmingham in the middle England and from there to Preston in northern England in order to attend the Commodore Scene meeting. Fortunately, there was a direct train from Haywards Heath to Birmingham, and so, I did not have to go through London, thus saving me time and trouble. It was on a high-speed, modern train with electrical ports at every pair of seats and food service in the centre car. The 3+ hour trip was not so bad, since it brought me past countryside I had not seen since 1995. There was the green, rolling countryside of Oxfordshire zooming past the windows, the canal boats plying their way in the canals, the quaint farmhouses and villages laying in the distance, the sky glowing blue and intermittently cloudy.

I arrived into Birmingham at about 1 p.m., too early for MicroMart newspaper writer Shaun Bebbington to come and pick me up; he was still at work in the town of Crewe. When I got to the well-policed train station, I made my way up and out into the shopping centre, the Pavilion. I picked up a sandwich from the Marks & Spencer Simply Food store and walked around the quite enormous mall. Going outside, I found myself on New Street, a pedestrian street which ran through the city centre directly to the photogenic Victoria Square and its regal buildings and imposing statues.

Photos of the square, stamps from the nearby post office, 35mm film from the Tesco store, and I still had many hours to kill before Shaun would see me at 7 p.m.. Hey, how about a movie? Indiana Jones and the Crystal Skull in the Odeon Theatre. I telephoned Shaun and told him my plans. Hauling my luggage up and down the steps of the theatre, I planted myself in a great position to see the movie. After my enduring half an hour of commercials and previews, the movie started. Movie review 3 out of 4 stars!

Because of the half hour of commercials and previews, the movie ended later than what I expected, and Shaun was already waiting outside the theatre for me. We took the bus and then walked the rest of the way to his girlfriend's house, the place where I would spend the night. Lisa, who was ill with stomach problems, did not at first meet me when we arrived. Shaun and I went to the nearby convenience market to buy medicine for Lisa and frozen pizzas for all of us. Later that night, Lisa joined us in conversation about travel which invariably wandered off and on to Commodore computer talk.

The next day Shaun and I took an early morning train out of Birmingham and toward Preston. The one-hour ride brought us to the larger- than-expected Preston train station and to Commodore Free editor Nigel Parker who was waiting for us. After a few minutes, Commodore Scene's Allan Bairstow drove up in his Chrysler minivan, and all of us piled in for the drive to Nigel's house, which was quite some way from town.

At Nigel's house, we met his wife, Suzanne, and his young son, Robert. After a quick tour of the modern premises (what a kitchen! Hey, an exercise room!), we men then trundled up the steep stairs to the loft office. It barely fit all four of us sitting down, so crowded was it with hardware and software plus goods ready for eBay sales.

I passed out Ghiradelli chocolates from San Francisco. The meeting ran for many hours, four to five hours as I recall. Though Allan had to leave after the first hour, we merrily carried on. The talk was wide and varied. We covered such things as Allan redoing his garage and thus his C= goods, the Maurice Randall situation, my video of California Commodore and Amiga clubs, my video of 1541 Ultimate creator Gideon Zweijtzer, the Behr-Bonz VIC-20 Multicart (a PAL one which I gifted to Shaun), various solid state card solutions such as the MMC2IEC, MMC64, MMC64-Retro Replay; clones of the FD-2000 and SuperCPU, the U.K's Dave Elliott and his former C= hardware, Commodore Gaming and Commodore Int'l., the U.K. scene including a new U.K. C= club,(EDITOR now officially named Commodore Computer Club U.K.) and C64 noters such as Noter-writer (because Nigel wanted to run a presentation at work and did not want to use Powerpoint on a PC). (EDITOR I am still on the look out for a "powerpoint" style application for the C64 not to bothered about graphics or sound but text in different sizes and fades from one page to the next with the press of the space bar or similar to advance to the next slide)

To keep up our energy, Suzanne was kind enough to bring us sandwiches and tea.

To keep up our energy, Suzanne was kind enough to bring us sandwiches and tea.

The last hour or so was spent in the back garden, out of the hot and stuffy loft. We watched as Nigel's young son played, and talk continued over such subjects as the C64 DTV and Jeri Ellsworth. Early evening and it was time for the drive back to the train station.

Another night at Lisa's house. She had made a fine salmon dinner, and afterwards, I talked to her curious teenage son, Kristien, about life in California. Hans Dussel of the HCC Commodore in the Netherlands had given me many DVDs filled with .pdf's of European Commodore magazines; I spent much of that night trying to make copies of those for Shaun.

In the morning, Shaun gave me a carrier bag in which to carry my goods, including an early PAL VIC-20 which he had gotten out of storage. Then he brought me back to the ever-busy Birmingham train station. Travel hint we discovered that it was cheaper to buy multi- legged tickets (from town to town to town) from Birmingham to Cardiff instead of one single ticket. It was a long but scenic ride to Cardiff for what was to be a brief 4-5 hour meeting with Matthew Leaman of Amigakit.com

I had never visited Cardiff, my last time in Wales being in 1995. As the train travelled nearer to that destination city, we passengers gazed out over the wide Severn estuary that led there. The weather was cloudy with areas of drizzle. I counted the cities in great anticipation of my arrival in Cardiff. Finally, the train pulled into Cardiff Central, a small station in comparison to that of Birmingham.

The signs greeted me in English and in Welsh. I pulled out my travel mobile phone and called Matthew. He would be over right away and

told me to meet him at a parking lot to the side of the station. When I got out, I was thrown for a loop. Which side parking lot? There were several. I waited in one for about 25 minutes. No Matthew. I called again. He repeated that it was a side parking lot. I went to another side parking lot. After waiting some more, I called him again and described the nearby landmarks to him. He didn't recognize those landmarks. I went back to the original side lot and waited. Another call and more description of the landmarks. Finally, to my relief he appeared and mentioned that this was not the side lot that he was thinking of.

Nevertheless, it was good to see him, our last meeting being at the AmiWest Show 2007. He drove to Cardiff Bay area. He was going to treat me to lunch, and we walked through a shopping mall in search of a good restaurant. Finding nothing there, we walked out onto the plaza itself. The weather had cleared up a bit, and the sun was shining through broken clouds, a wind blowing from the Irish Sea.

As we walked, he proudly pointed to several signs which had familiar-looking television actors and remarked, Cardiff, the home of Doctor Who. He pointed out the Cardiff opera house, used as background for a Doctor Who episode of a few years ago. Surprised, I told him I remembered that episode and the surrounding plaza that was used in the episode. After looking at a few restaurants that faced the sea, we decided on an upscale pizza restaurant. Waiters in uniform welcomed us into the bright, cheery, modern restaurant a sign that this place was going to charge the big bucks (or should I say the big pounds).

Matthew and I talked of many things the Commodore and Amiga clubs in the U.S., my travels around Europe, his AmigaKit business, and future Amiga products. The pizza came a large combination for me (or at least the closest the restaurant could come to a combination and it still wasn't gigantic American-style). Between bites of pizza and downing drinks, we happily conversed.

We could have stayed longer, but time was running out; I had to make my late afternoon train. It was off to AmigaKit, and I discovered it was in an industrial park. Matthew led me through the hallowed doors of his establishment and into the main reception area of the office. There I found his jolly assistant, Dave Markey, who I had previously met at AmiWest. He was very glad to see me, and it was good seeing my instant friend. The reception area was bare, save for a desk for Matthew to do his accounting and several Amigas that had been sent in for repair or upgrade. The more interesting area was the back room; here Dave had a couple of workbenches on which the computers would be repaired / upgraded. He was working on an A4000 desktop with PPC board, its innards flowing out. Spare parts used in repairs were stacked on the floor to the side of the workbenches. However, those two rooms were not all. In the grand tour of the offices, Matthew led me to the warehouse, a giant room across the hall. Behind its dark-colored door was treasure!

Amiga hardware and software... more and more... some piled in neat stacks, others on shelves. Boxes of unopened, new AmigaTech A1200s stacked to the ceiling. Boxed games. All kinds of adapters and cables. Used hardware and software. CDTV controllers (over 3,000!). Even Commodore 64 software that Matthew obtained from the defunct High Street Micro in Crewe. Matthew offered it all, and it was so tempting. I had to steel myself. My mission was to get a new, formatted hard drive for one of my A1200s, a package of AmigaOS 2.1, and for friends, one or two NTSC CD-32s. No, not the solid state hard drive. No, not a new tower to replace a ramshackle A1200 tower I had. I had to consider that the things I bought must fit in my suitcases for the airline journey home.

O.K., an 80-gig two-and-half inch IDE hard drive for the A1200 first. Matthew got cracking to it. He was going to format and install OS 3.9 on it. Meanwhile, I wandered around, looking at the hardware on the workbenches, talking to Dave, and taking photos and video of the areas I was permitted to record.

Prepping the 80-gig drive took some time, and my train deadline was coming up. I asked Matthew about the NTSC CD-32. He showed me a big box with unwrapped CD-32s jumbled inside haphazardly. There were 4 big boxes with a total of over 300 CD-32s. They had come from the CBM warehouse in the Philippines. Oh, so this is where those last CD-32s ended up, I said. Matthew cautioned me, Only one out of thirty is NTSC.

Uh-oh. With so little time left, Matthew couldn't test a whole slew of CD-32s. I reached into the box and drew out what I hoped would be the lucky NTSC one. He took it over to a workbench and hooked it up. After a few minutes, he gave me the bad news; it was PAL. asked him about AmigaOS 2.1; he didn't have it readily available. No time to look for it. I grabbed the drive, an empty box for the AmigaOne board I obtained a few days earlier, and an unusual, new-in-box Sega SG Fighter joystick unusual in that it seemed very similar to a flight joystick. I paid for the goodies with the good, old Visa credit card. Then Matthew and I rushed out of his establishment in our dash for the train station.

Traffic seemed slower on the way back. I had missed the train which would take me through Reading and to my destination of Haywards Heath. I had to take the next train which took the longer route through London. I would have to change trains, and then I would catch a southbound one to Haywards Heath. There was a traffic jam at the train station. Finally, Matthew edged his car in, he helped me unload my cases and bought goodies, and we shook hands.

I'll see you at AmiWest, Matthew, I said. He promised to bring the AmigaOS 2.1 with him. Then waving good-bye, I walked into the station to await my train.

Interviews Sean McManus

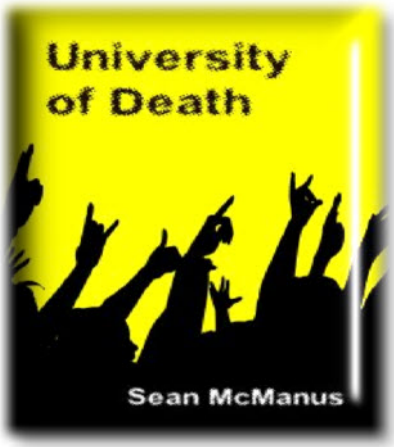
Sean McManus
creator of a new novel involving the Commodore 64
http://www.sean.co.uk/index.shtm

Commodore 64 co-stars in music industry novel

The Commodore 64 co-stars in a thrilling new novel that satirises the music industry. ‘University of Death` by Sean McManus reveals what happens when a major record label builds a program that creates and markets perfect pop songs, tailored for each listener`s taste. At the heart of the system is a Commodore 64 churning out random melodies.

Commodore Free
Free Please introduce yourself

Sean McManus
I'm a writer and keen retrogamer, based in London. I've just published my first novel, University of Death, which is all about the plight of the music industry and is named after the band at the centre of the story. The book explores how fans relate to their favourite bands, how businesses use technology to manipulate consumers, and what would happen if the music industry disappeared overnight.



The story has a cast of famous people in cameo roles, and a special guest role for the humble Commodore 64!

CF. How does the Commodore fit in?

SMM. In the story, there's a computer program that's inventing random bands, including their music. A major record label is using spyware hidden on fans' computers to sell them this fake music. When I started writing the book, that idea seemed far-fetched, but then Sony BMG was caught putting anti-piracy software onto Celine Dion CDs that was classified as malware. All of a sudden, that bit of the story didn't seem quite so unlikely.

Anyway, in the middle of this fictional-bands system is a Commodore 64 that has been churning out random melodies for twenty years. There were a couple of reasons for choosing to put a Commodore there, and not something more modern. One was that using a Commodore 64 meant that one man with no particular IT expertise could keep it running without any hassle. Today, you'd need a team of consultants and a big pot of money to create even the simplest original software, which would make it hard to keep the whole scam secret.

The other reason was that I just liked the aesthetics of the Commodore. I know I have an emotional response to classic machines from the 80s, which it's difficult to have with anything post-Windows. I'm guessing others feel the same the Commodore is a style icon.

The Commodore is also widely respected for its music capabilities. As it turns out, that didn't matter for my story the machine in my book doesn't make a sound itself but it was easier to believe that someone from a record company had been amazed at the synth-like sounds of a Commodore, than it was of a Spectrum or Amstrad. And that this had sparked the whole dastardly plan.

CF. What was your first machine?

SMM. Well, speaking of Commodores, the first machine I owned was an emulator. I grew up with Amstrads, and came across Commodores mainly through friends. We'd play Commodore classics like Yie Ar Kung Fu, Wizball (which was only good on the Commodore, really) and Beach Head (which features in my story). I remember friends showing me the latest demos as well the Commodore had a really lively demo scene, with the music being particularly impressive. There's still a great Commodore music scene today I went to a Back in Time event last year and it was the maddest and greatest thing I'd seen all those people dancing to the music from I-Ball.

CF. What machines do you have today?

SMM. Today, I've got a couple of Amstrads, a couple of Spectrums and the C64 TV game joystick. That is a superb invention it's a shame it's been discontinued. It would have been nice to see more games get a new lease of life that way. I also come across new games through online emulators from time to time, including Lazy Jones which I played online after reading a story in Retro Gamer about it.

CF. Tell us about your contribution to Commodore Format magazine?

SMM. As the 8-bit scene was winding down, I did have an opportunity for a fleeting moment of Commodore glory when I wrote a review of a new emulator for Commodore Format magazine. At the time, I was writing type-ins and tutorials for Amstrad Action and Amstrad Computer User magazine and I came across a Commodore emulator. It was at a time before emulators were popular. Most people who cared were probably still playing the real thing, and it was too early for nostalgia to kick in.

The internet wasn't widely available, but you used to have shareware cafes in Germany where I was living at the time. You'd pay to copy a floppy worth of software from their vast shareware archive. That's where I found this C64 emulator, which ran in DOS on my 386 laptop. I pitched a review to the editor of Commodore Format and it became one of my earliest published freelance articles.

CF. Will there be a follow-up to this book?

SMM. Highly unlikely. The story as it stands is well-rounded and has a beginning, middle and an end. It doesn't really need a sequel. I've been thinking about another aspect of popular culture I'd like to explore in a book though, so there might be a completely different novel at some stage in the future. It took two years to write this one, though, so I need to muster the energy and gather all the ideas I need first.

CF. Where can people find out more?

SMM. You can download the first two chapters of 'University of Death' at my website at www.sean.co.uk. The Commodore doesn't enter the story until a bit later, but it gives a good flavour of the tone and introduces the main characters. There's also more information on what inspired the book, and reviews from magazines like Record Collector and Metal Hammer on the site. The book's only available at www.lulu.com, and you can find the direct ordering links easily through my site. Thank you for your interest!

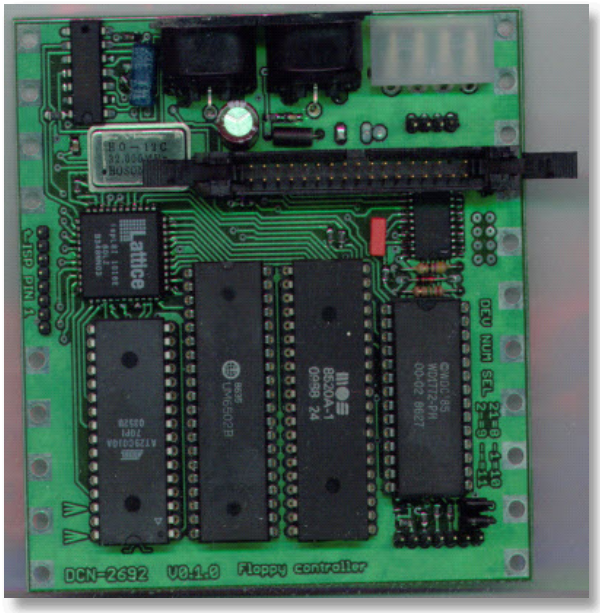
DCN-2692 floppy controller board

http://www.iki.fi/mkl/dcn2692/

DCN-2692 floppy controller board
http://www.iki.fi/mkl/dcn2692/
DCN-2692 is suitable to use with Commodores as a 1581 "replica"
This document was last modified on 30-dec-2007 ...

Don't use the buggy first revision of the original 1581 ROM with a missing sei instruction that will lead to data loss!
WWW for "DCN": http://www.iki.fi/mkl/dcn.html
The above link is outside of this page you are currently viewing. There will be links to related information, such as data of the ICs used. Also follow the above link for other floppy controller projects & information. Currently, there are pictures of an early prototype of an 1581-clone.

I don't have any of these controllers for sale at this moment. If you plan to build one, please check all the necessary parts are available (and at a reasonable cost.) WD1772, for example, is rare.



Information about the board
The controller is intended to be used with a low cost PC HD floppy drive unit.
Not all PC HD floppy drives are fully compatible. For example, the JU-257A427P (I have rev. F) may fail. It does not respond fast enough to the RW-head track-to-track stepping commands, and therefore the RW-head ends up on a wrong track, which leads to data loss at worst. By changing the code on the ROM on the board, the WD1772 Floppy Disk Controller could be instructed to use the slowest setting for head-step. I have not tried this modification in practise. It is also rumoured, that the cheap drives of today are not of very high quality, and also not designed to be used with the obsolescent, hard to find, Double Density floppies.

The floppy cable from the board to the drive is direct, without a twist in the cable. Number 1 pin of the cable and connector is near the edge of the board. Odd numbered pins are ground, except for pin 3. Number 3 of the connector is a key pin, and should be removed, if the cable has a filled location, that does not accept a pin. In a Commodore/Amiga DD-floppy mechanism, a few pins are different. I have included some information on pin outs EDITOR COMMENTS "I have copied the document out below FILE: floppypinoutinfo.txt)
FILE: floppypinoutinfo.txt
HTTP://www.iki.fi/mkl/dcn.html

Notes:
The DD-drive used in 1581 has a slightly different pin out than "PC-drives".
The same type of drive that are used in 1581 are also used in the Amiga 500.
"PC-drives" have disk change signal on pin 34, and 1581 drives have it on
pin 2. 1581 drive has "ready"-signal on pin 34. Pin 3 can be a key-pin and removed from the pin header connector.

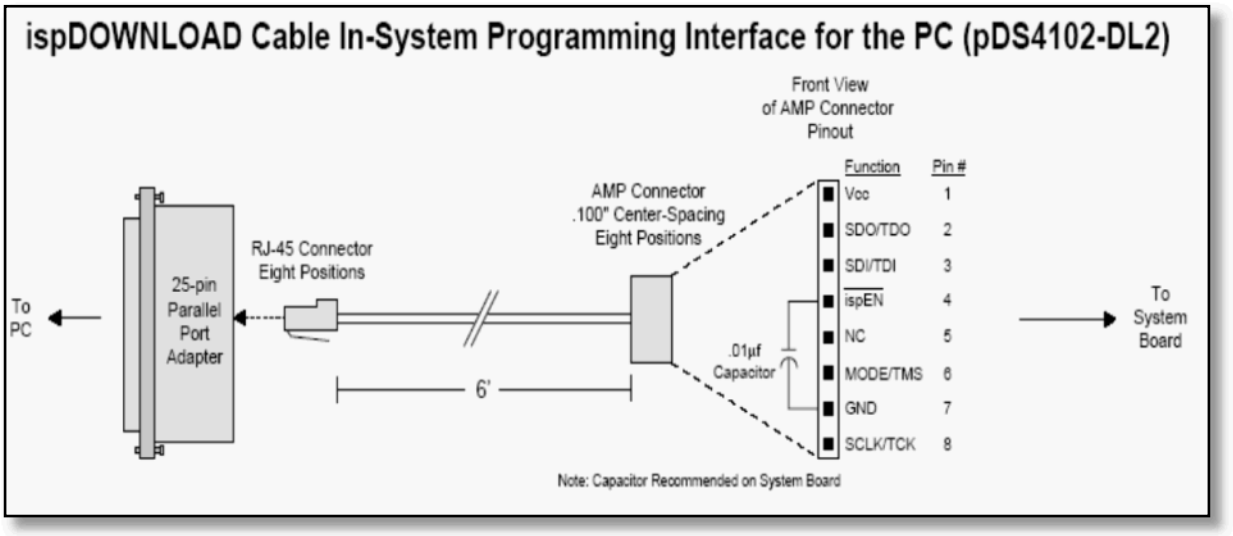
----- This 1581 PCB-connector info is collected from schematics page 16 -----
----- of "Service Manual, 1581, 3.5" Disk Drive, June 1987 PN-314982-01" -----
Pin 1 not connected in this schematic Other odd numbered pins 3 trough 33 are grounds

```
_____________
/             \
|1           2| /Disk change
|3           4| n.c.
|5           6| n.c.
|7           8| /Index pulse
|9          10| /Drive select grounded
|11         12| n.c.
|13         14| n.c.
|15         16| /Motor on
|17         18| /Step direction
|19         20| /Step
|21         22| /Write data
|23         24| /Write gate
|25         26| /Track 0
|27         28| /Write protect
|29         30| /Read data
|31         32| /Side 0
|33         34| /Ready
\_____________/
```

This is a cut from "Pin outs for various connectors in Real Life"
Notice that the signals that are active low are not marked as such
6.5) Floppy

6.5.1) Floppy Disk Controller IDC-34 Male

pin	assignment	pin	assignment
1	GND	2	Density Select
3	GND	4	(reserved)
5	GND	6	(reserved)
7	GND	8	Index
9	GND	10	Motor Enable A
11	GND	12	Drive Sel B
13	GND	14	Drive Sel A
15	GND	16	Motor Enable B
17	GND	18	Direction
19	GND	20	Step
21	GND	22	Write Data
23	GND	24	Floppy Write Enable
25	GND	26	Track 0
27	GND	28	Write Protect
29	GND	30	Read Data
31	GND	32	Head Select
33	GND	34	Disk Change



The activity LED on the drive unit will normally always be lit, because the drive select signal from the board to the drive unit will be normally always active, as in C-1581, where the drive mechanism doesn't have a LED of it's own. If a primary/secondary floppy drive selection signal was implemented in the programmable logic device in this design, the LED on the drive would indicate which unit is selected at that time by the controller. To use HD-floppies on a HD drive instead of DD-floppies, You can put adhesive tape over the HD/DD-recognition hole. For some reason, without that trick the HD-floppies do not seem to work. I do not know however if this could cause any reliability problems.

Anyways, You shouldn't use HD floppies on an DD only drive.

Jumpers and headers on the board
To select the device number for the Commodore serial bus, You can use the jumpers at the top right hand corner of the board.

Jumper1 is the one which is closer to the board edge

Device number	Jumper2	Jumper1
8	Closed	Closed
9	Closed	Open
10	Open	Closed
11	Open	Open

Next to these jumpers, there is also a pin header, this has the device select signals and outputs to connect two LEDs. The LEDs can be connected between +5V and the output. Series resistors for the LEDs are on the board. The Power LED in the CBM-1581 has two functions: it is normally lit to indicate that power is on, and in case of error or something special like that, the led blinks from dim to bright and then back.

I have omitted the resistor that keeps the power led dimly lit. This resistor is added to the board in version 0.1.0. The resistor can be added to the circuit even if there is no place for it on the board. It is connected between ground and the cathode of the power led, or pin 5 of the header. The value of the resistor could be something between about 100 to 1000 ohms. Pin outs for the header are, from the top right corner of the PCB:

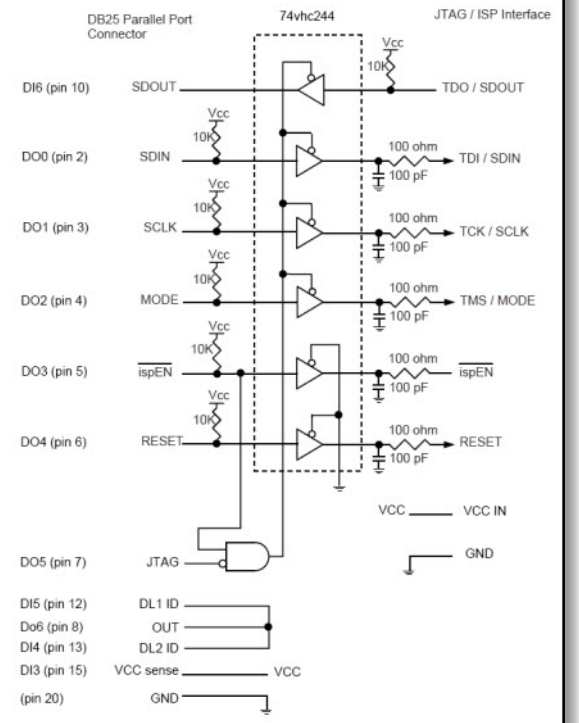
1 GND
2 Vcc (+5V) (Anodes (+) of LEDs can be connected to this)
3 DEVN0 (device ID number select bit 0)
4 DEVN1 (device ID number select bit 1)
5 Power LED (Cathode -)
6 Activity LED (Cathode -)
7 GND

Powering the controller
The supply voltage should be 5 volts +-5% (4.75 - 5.25) at a max current of about 1 ampere (could be much less actually). The drive mechanism needs roughly that (it takes more current when the motor starts to spin than when the drive is idle.) At the input there is a fuse, this will prevent smoke coming out of somewhere, in case of short-circuit. You could use a resettable poly switch type fuse, but the fuse needs to be fast acting to avoid burnt PCB traces. It is also possible to replace the fuse with a wire link, but then the protection will be lost. Across the supply voltage, after the fuse, there is a zener-diode or a transient voltage suppressor diode on board, which starts to conduct if the supply voltage goes too much over 5.25 volts. (at 5.6 - 6.8 maybe) or below 0 volts with respect to ground. This part can be omitted, but then the protection is missing as well.

8520A vs 6526A CIA interface adapter IC:s
MOS/CSG CIA:s 8520 and 6526 are quite similar devices. 8520 does not have the same "TOD"-clock that 6526 does have. 8520 seems to have a more powerful output at some of the pins, where as outputs from 6526 are capable of sinking only 3.2 mA (min.) or sourcing 200 uA (min.) 1 mA (typ.) That is the reason why there are some extra buffers from the signals from CIA to Floppy-connector. The original C-1581 uses a 8520A. The letter A in 8520A or 6526A means that the chips are rated for 2 MHz. Links to more information can be found here http://www.students.tut.fi/~leinone3/dcn.html

WD1772 vs WD1770 floppy controller IC:s
I have only used this controller with WD1772. WD1770 is quite compatible, but it has a few differences. In the 1581 schematic it suggested that the 2 MHz PHI2 clock is connected to pin 19 of 8520A CIA, when WD1770 is used. This pin is the input for the counter, this is different between 6526 and 8520, so the combination of 6526 and WD1770 might possibly not work? On DCN-2692, the pin 19 is tied to VCC voltage level. AFAIK, Commodore started us-

Figure 5. Buffered 25-Pin Parallel Port Adapter



ing WD1772 instead of WD1770, when it was discovered that some of the WD1770:s were faulty, and these could corrupt data on disk.

ROM types
Atmel FLASH PEROM type AT29C010AP (128 KBytes) is used here. It is reprogrammable without extra programming voltages, and it is possible to reprogram it in-system. The ROM socket should also accept other types of ROMs, including 28-pin types, which should be inserted correctly to the 32-pin socket. It needs to be at least 32 KiloBytes in size to hold the original ROM content.

Ready signal
PC-drives differ from the standard. One thing missing is the drive ready signal. Standard drives assert this signal when the floppy disk is spinning at a steady speed and is ready to be read from or written onto. It should take about a half second after the motor has been turned on. If it takes too long, the 1581 will give an error, saying drive not ready. This signal is emulated by R15, R14, D1, C10, and a buffer in IC10. The component values were changed between V0.0.4 and V0.1.0. There is a slim chance that a PC floppy drive has the Ready signal somewhere on its circuit board.

Compatibility with Commodore 1581
This should be compatible. But I had trouble with cbm4win and my PC; copying files to dcn-2692 caused the transfer to hang many times, but this could have happened because of the slow transistors I used in the XA1541 adapter. There are known differences, but I don't know whether these cause any compatibility issues: The lowest 16 KBytes of SRAM is visible at the lowest 16 KByte range of the 6502. The C1581 has 8 KBytes of RAM, and it is (IIRC) mirrored to 6502 address space 2000..3FFF. The ready signal from the floppy drive is simulated so, that the CIA senses the simulated "ready" going low (=active) about 0.5 seconds after the CIA drives the "motor" signal of the floppy drive low (=active.)

Please note that I haven't tested the board with the C128 fast burst transfer mode.

CPLD files and download cable schematic
Some of the logic (e.g. glue logic, etc.) is placed on a programmable logic chip. CPLD stands for Complex Programmable Logic Device. Such an IC and can be user programmed it can be used instead of standard TTL/CMOS logic ICs, such as 74LS00 etc. The type used with this board is Lattice Semiconductor ispLSI1016E-80LJ (where J stands for PLCC package.) It should also be possible to use the ispLSI1016 or ispLSI1016EA version, and any speed grade will be fast enough. But it is needed to recompile the ABEL source for each version of the CPLD.

dcn2.abl ABEL sourcefile for the CPLD, date 20-May-2003
dcn2.jed bitfile for ispLSI1016E-80LJ download
http://www.students.tut.fi/~leinone3/dcn2692/dcn2.jed

```
MODULE dcn2

TITLE 'dcn2'
"This version is UNFINISHED, but provides basic functioning for a
1581 clone
"20-May-2003
"Projects on the web at http://www.iki.fi/mkl
"The target CPLD is Lattice ispLSI1016-E80LJ, or other PLCC-44
ispLSI1016
"use Slowslew pin attribute when compiling design
"This ABEL file describes three functionally separate modules. (In
one file to save(?) trouble)
"1. Logic that replaces various IEC-bus related logic gates and buf-
fers in the original CBM-1581
"2. Reset/Clock control logic that
"(i) generates a delayed system reset to CPU, CIA and WD from
master reset
"(ii) generates 2 MHz 6502 CPU clock PHI0, which MUST be run-
ning before reset is removed
"(iii) generates WD1772 clock, which is selectable between 8 or 16
MHz
"3. Logic to control chip selects, output and write enables for
RAM,(e)ROM,CIA,WD
"   and manage memory banks of 32 KByte RAM, and ROM (size
32, 64 or 128 KBytes)
"   and select the WD clock between 8 or 16 MHz
"   and select the floppy drive unit (Primary / Secondary)
"After master reset the signals are as follows:
"WD clock is 8 MHz (as in original 1581)
"Primary drive signal is active (low). The same signal will be high
for secondary drive.
"Lowest 16 KB part of RAM is present at 6502 address $0000-$3fff
"CIA is visible at $4000-$4fff (registers at $4000-400f)
"WD is visible at $6000-$6fff (registers at $6000-6003)
"HIGHEST 32 Kbyte of ROM (PROM, ROM, EPROM or FLASH)
is normally at $8000-$ffff
"SECOND LOWEST 4 KByte of ROM(ROM address $1000-1fff) is
FIXED at $5000-$5fff.
"A13 is copied to XA13 (the address pin on RAM/ROM)
"A14 is copied to XA14 (--""--)
"XA15 and XA16 are set to high
"XA13..XA16 outputs are for memory bank selection
"The configurable outputs and memory management get their input
from CPU data line D7
"Address line A12 participates in selecting the configurable registers
at $7000-$7fff
"The configuration data is fed SERIALLY from D7 using a special
protocol algorithm.
```

```
"^^^^above function not yet implemented

"-------------------------------------------------------------------------------
"IEC logic part of design PIN DECLARATIONS
"IEC is a name for the Commodore serial bus
        DATA_IN pin 22; "inverted IEC_DATA line state from IEC BUS
                        "(signal from 74ls14 inverting schmitt trigger gate)
        DATA_OUT pin 19;  "output to 7406 inverting open collector driver to IEC_DATA line
        ATN_IN pin 20;    "IEC signal...
        FCLK_IN pin 21;   "IEC signal...
        FCLK_OUT pin 18;  "IEC signal....
                "Note: IEC CLK line logic is not routed via this CPLD
"Signals from/to CIA
        DATAIN_CIA pin 4; "to CIA
        DATAOUT_CIA pin 5; "from CIA
        ATNIN_CIA pin 39; "to CIA
        ATNACK_CIA pin 6; "from CIA
        FASTDIR_CIA pin 7; "from CIA
        FASTCLK_CIA pin 10; "bi-directional from/to CIA
        FASTDATA_CIA pin 9; "bi-directional from/to CIA
"-------------------------------------------------------------------------------
"-------------------------------------------------------------------------------
"Control logic PART pin declarations
"inputs
        CLK pin 11;

        PHI2 pin 2;
        RnW pin 43;
        A12 pin 32;
        A13 pin 40;
        A14 pin 41;
        A15 pin 42;

        D7 pin 31;

"outputs
        PHI0 pin 44;
        nRESET pin 3 istype 'reg';
        WDCLK pin 16;

        nPRIMDRIVE pin 17;
        XA13 pin 26;
        XA14 pin 28;
        XA15 pin 38;
        XA16 pin 37;
        nWE pin 27;
        nOE pin 29;
        nCSROM pin 30;
        nCSRAM pin 25;
        nCSCIA pin 8;
        nCSWD pin 15;

"Internal stuff
        q7..q0 node istype 'reg';
        countteri = [q7..q0]; " set

"-------------------------------------------------------------------------------

"------- LOGIC EQUATIONS ----------------------------------------------
equations
"IEC logic equations (this part is asynchronous logic)
        ATNIN_CIA = ATN_IN; "the signal is only passed to another pin on CPLD
        DATAIN_CIA = DATA_IN; "same here...
        DATA_OUT = DATAOUT_CIA # (ATN_IN & ATNACK_CIA) # (FASTDIR_CIA & !FASTDATA_CIA);
        "DATA_OUT is controlled by many sources...
        FCLK_OUT = FASTDIR_CIA & !FASTCLK_CIA;
        FASTDATA_CIA = (!DATA_IN);
        FASTCLK_CIA = (!FCLK_IN);
        FASTDATA_CIA.oe = !FASTDIR_CIA; "when FAST-DIR is low, direction is towards the CIA
        FASTCLK_CIA.oe = !FASTDIR_CIA;  "and when high, towards the IEC BUS buffers.

"Control part equations
"All registers are reset to zero by the dedicated asynchronous reset input on the CPLD.
"Therefore, register asynchronous resets are not explicitly described.
"Clock & Reset output control
        countteri.clk = CLK; "clock is 32 MHz
        countteri:=countteri.fb+1; "counter function :-)
        PHI0=q3; "4 stages divide the frequency to 32->16->8->4->2 MHz
        nRESET.clk = q7; "asynchronous clocking...
        nRESET:='1'; " sets reset to inactive high
        "WDCLK selection logic below....
        WDCLK=q1;
"simple....
        nPRIMDRIVE = '0';
        nWE = ! (!RnW & PHI2);
        nOE = ! RnW;
        when ([A15..A12]==5) then [XA16,XA15,XA14,XA13] = [0,0,0,0];
                                else [XA16,XA15,XA14,XA13] = [1,1,A14,A13]; "ROM banking
        "chip selects
        nCSROM =! ((A15==1) # ([A15..A12]==5));" $8000-ffff + $5000-$5fff
        nCSRAM =! ([A15,A14]==[0,0]); "$0000-$3fff
        nCSCIA =! ([A15..A12]==4); "$4000-$4fff
        nCSWD  =! (([A15..A12]==6) & PHI2); "$6000-$6fff, phi2 is involved...


END
```

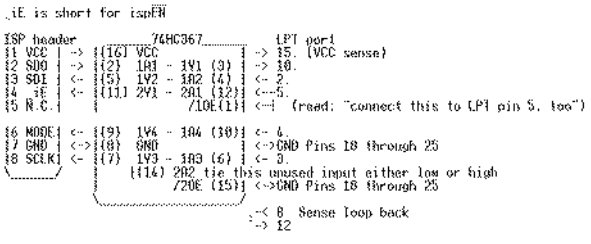
Future versions of cpld will have registers for extra control of ram, rom and the drive. I have made some advancement on developing that functionality, but it is not ready yet.

Lattice Semiconductor website http://www.latticesemi.com/
Download software that programs the PLD with a .jed bit-file (windblows pc and perhaps also x86 Linux). They might ask for registration, etc.
ispLSI download cable
(No guarantee there's no errors, but it should be almost ok.)
You could use small value (47-100 ohm) series resistors for signals and a decoupling capacitor (1-1000 nF) between VCC and GND.

_iE is short for ispEN

Images from from isp download cable pdf (june 2000.) header pins
http://www.students.tut.fi/~leinone3/dcn2692/ispdlckuva1.png
http://www.students.tut.fi/~leinone3/dcn2692/ispdlckuva2.png

schematic
Image from Lattice isp manual pdf(1996)



http://www.students.tut.fi/~leinone3/dcn2692/ispcable.png

Version history
PCB Version 0.1.2: I had one pce factory made.
PCB Version 0.1.1: I haven't built this one. 100 x 100 mm board size. PCB masks are in a single A4 size Postscript sheet. All holes are 0.3 mm "drill guide holes", for hand drilling aide. The schematic is in PNG format. PCB version 0.1.0: 6 pcs of these came from a PCB factory. A number of small changes since 0.0.4, for example 7406 IC is now in DIL-14 package instead of SMD SOIC-14.
PCB version 0.0.4: This is the same as version 0.0.2 with updated documents, eg. component values were added to the schematic. PCB version 0.0.2: 11 pcs of this board were made at a PCB factory.
PCB version 0.0.1: First prototype, PCB etched at home.
About the components, Some notesVersion 0.0.2 is the same as 0.0.4, but the schematic diagram v0.0.4 is updated. (It has component values.)

Please note that the BOM (bill of materials) files are not fully exact or complete, that means the raw BOMs are not perfect shopping lists.
The 47 ohm series resistors R10, R11, R17 and R18 are there to protect the CIA I/O pins from over-current, and maybe could be omitted and replaced with solder blobs. The 47 ohm series resistor R8 and R12 are like source termination resistors for the clock lines 32MHz and WDCLK. The value 47 ohm is only a guess, actually a good value would depend on the impedance of a transmission line. C1581 also used ferrite beads in series for these lines.

The power supply line ferrite bead (FB1, or L1 in some schematic versions) should be of a higher current type than the type used for small signals. If the ferrite gets saturated, it doesn't do a good job in suppressing RF noise leaving or entering the voltage supply connector. I have not actually made any EMI measurement if this component here has any significance, or could it be replace with something better or a wire link. The power supply connector does not appear in the genereated BOMs in the table. The "correct" type of 5.25"-drive/3.5"-HDD power input connector is the one which is normally mounted the other, and not the other side of the CDROM/3.5"-HDD/5.25"-floppydrive PCB. The 4-pin smaller (2.50 mm pitch) power connect on the board is not actually the same type that appears in the BOMs, but instead the type used with 3.5" floppy disk drives.
An 47 ohm SMD resistor does not appear in the generated BOMs. Its job is to limit the current in case there is a short circuit to ground when plugging in the CPLD programming cable. (This sometimes happens with my self built programming cable, which has the other row of two row flat cable connector connected to ground.)

The ispLSI programming connector header does not appear in the generated BOMs except for V0.0.4. It has an 8-pin 0.1" pitch header, where one no-connection pin can be removed, and actually should be removed, starting from V0.1.1.
74VHCT245 has been used here because its CMOS sensitivity allows use of small value (220 nF) capacitor with a larger value resistor (1.5 Mohm) in the Ready signal simulator circuit. Connector "Floppy" is an 34-pin IDC connector / dual row pin header, where one pin (number 3) should be removed.

Bugs
The series base resistor values (1 kohm) for BC847 npn transistor values are too small, because the 6526 type CIA typically sources only about one milliampere. So resistor values of R19 and R20 should be increased to 8.2 kohms. (26-march-2006) According to datasheets, the 6502 requires voltage swing to full VCC (=5 volts) at the clock input. But the ispLSI1016 only outputs about 4 volts at high logic level. The controller seems to work despite of that. Diode D2 in ready-signal-generator-circuit is superfluous and may be omitted.
There is a bug in the factory made v0.1.0 PCB:s It is a short missing VCC trace between two pull-up resistors R26 and R27 on the underside, near IC4(WD1772). The missing trace is added in these V010 postscript files below (V0.1.0a in table), so it is a slight bit different from the factory made ones in this good way. Copy paste from BOM-file for v0.1.0: IC7 (74LS14D) could be replaced with 74F14D, which have "stronger" outputs, which might be needed when driving the signals to two drives on the same cable, instead of only one drive on the cable. Usually each drive has 1 kohm pull up on the signals. For a logic TTL-low, an input must sense at most 0.8 volts, and that implies also a current. If there are two drives on a cable, the combined pull up resistance would be 0.5 kohm, which requires more current sinking capability from the signal driver.
Usage You may use these documents to build a board, but you will responsible that you can build it safely, and not burn your hand with a soldering iron and not become blind with dangerous PCB manufacturing chemicals etc. And of course there is no warranty that it will work, or work reliably.

COMMODORE FREE

I would like to thank Mika Leinonen" for the permission to reprint this information in Commodore free magazine and although Mika no longer makes the board due to the shortage of parts especially the very hard to find WD1772 Chip
Here are the notes Mika sent me about the board

" Hello, I was busy and almost forgot to answer, but now i do. I haven't produced more of these, because WD1772 is very rare. I could make some boards with socket, where the user would have to hunt the rare IC. magazine reprint is ok,"

The item was 85 Euros plus postage,

REAL LIFE USAGE

The board seemed to function without problems on the many drives I uses, from multiple manufacturers, I didn’t seem to have any compatibility issues, I am sure now with SD card readers there is less demand for s user to load items from floppy disk even if this is a 3.5" drive unit. However at the time of purchase for me it seemed the only cheep way to get a Commodore 64 into a PC case and load games from some device. My Idea was purely a station to load games and to that end the device worked well (although slow) I tested the device with fast loader carts although they seemed to create no speed advantage. I, Alan and Shaun tested the unit with geos and wheels, remembering this unit doesn’t have any form of speedup chip so no jiffy-dos which is a shame, the unit would format disks in Geos and wheels and read from them; but every now and again would cause both Geos and wheels to crash unrecoverable.

Interview with Robert Hurst

http://robert.hurst-ri.us/

Commodore FREE:
Firstly, please introduce yourself to our readers.

Robert Hurst:
I work as a senior information systems engineer for a major Boston-based healthcare system, which is part of the Harvard School of Medicine. I am responsible for keeping their core clinical systems operational 7x24x365, which is now attainable since we made the move to Red Hat Enterprise Linux with high availability clustering last year. With that success, we are now moving off of commodity racked servers and into highly available IBM BladeCenters, for production and disaster recovery.

CF: What was the first VIC-20 game that you wrote? And was it in machine language or BASIC?

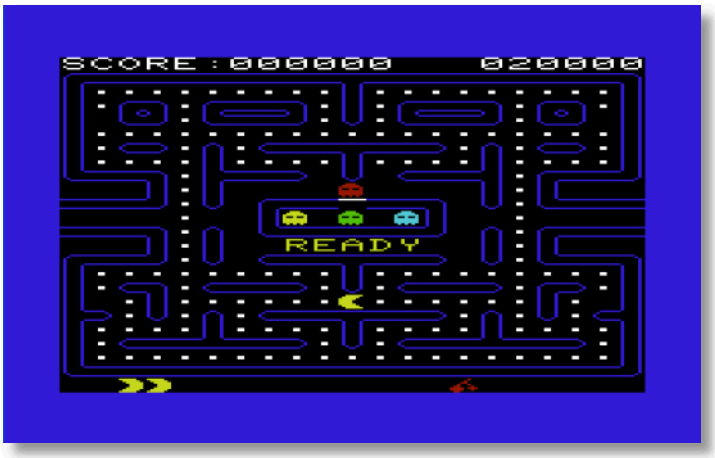
RH: It was called "Assassinate the Teacher" written in BASIC. I was a Junior in high school taking a computer math course using WANG microcomputers. I had just gotten my VIC 20 and begged my teacher to allow me to demo this friendly home computer to the class. He gave me permission and this game was showcased at the end of that demo. I was glad he had a sense of humor! Actually, he was really impressed that $300 could buy so much power, with color and sound, compared to $5000 for each microcomputer. The year after I graduated, that computer lab was re-fitted with thirty Commodore PET 4032.

CF: Was the VIC-20 the first computer you owned? Did you have any other Commodore computers?

RH: Yes. And I remained brand-loyal with C16, C128, Amiga 2000, 3000, and 3000T
for about 16 years. Emulation has made it possible to resurrect my software library and
to collect favorite titles. Recently, I bought a brand new Plus/4 from ebay, just because.

CF: How much more difficult is it to write a game in a machine language monitor?

RH: Interesting question. I would say the "ease" of using today's assemblers compared to a machine language monitor is roughly the same ratio -- as using back then -- a machine language monitor was used to poke byte-values from BASIC. I would recommend the use of a machine language monitor for real-time debugging purposes only.



CF: Have you ever thought of releasing your games on real media?

RH: Absolutely! I thought I had an "in" with a national pizza chain here called Domino's Pizza, whereas they could distribute my 'Pizza Delivery Man!' game on a tape cassette housed in a miniature version of their pizza box. Unfortunately, my representative that met with executive marketing at a local convention came back unsuccessful. Years later, I would port it for an Amiga magazine and sold it to their monthly diskette edition for $400.

CF: Are you going to produce more VIC games, or any other 8-bit machines?

RH: Undecided, frankly. Now that I have a good grasp on today's assemblers and debugging using emulators, I am tempted to tinker some more. As a former educator at the technical college level, I see potential educational value in this for would-be programmers, if this process and its tools could be packaged with associated training guides. I am disappointed with today's programmer apprentices -- "the good ones" always seem to have at least learned from some part of past practices, and are markedly better from it.

CF: How quickly did you manage to "pick up the code" from Quikman as it wasn't finished?

RH: A matter of days. It got profoundly "quiker" as I began commenting the original listing and using labels for subroutines, constants, and variables. Applying today's good coding style to legacy code is 'A Good Thing', which results in faster development and a better product overall. That programming style was just not feasible given the development tools of the time, combined with storage and memory constraints.

CF: Do you feel it's now easier to produce games for 8-bit machines because there are more tools readily available and therefore development time has been reduced?

RH: Without a doubt. Imagine using a machine language monitor and having to save hours of work to an uncertain piece of magnetic tape, which took about a minute to save and another minute to verify (and trust me, you did take the time to do the verify!) And when your code crashed *badly* by corrupting memory, you needed to re-load from tape back to your last point-in-time and redo work. Today, you save multiple versions to disk, compile-link-and-go, all in sub-second response time. Actually, the slowest part of the process is turning on or resetting the ROM-based VIC 20 -- which takes at least a full second to boot! But more importantly than those conveniences is the availability of modern 6502 assemblers. Moving blocks of code and/or data using a machine language monitor is very complicated. Editing a text file and compiling to produce a new binary is typically a cut & paste operation.

CF: Will you be revisiting any of you other old projects?

RH: Possibly, but more likely I would spend time doing something new. I was at a point with VIC 20 programming, back in 1984, where I could write 100% machine language programs, with smooth moving graphics and animations, on an unexpanded machine. I like that challenge, and I would like to see what else can be produced given that tiny footprint.

CF: Did you finish QUIKMAN as a nostalgic trip or as an ego trip because you had an unfinished project that was nagging away at the back of your mind?

RH: Definitely an ego trip, ha! Anyone who has worked with me knows I am never completely satisfied with my work... but only satisfied to a point where I feel my original objectives for doing it in the first place have been met. That is when I can walk away from a project without feeling any failure on my part.

CF: Have you had many comments about QUIKMAN

RH: Suprisingly, yes. I did not realize there is still a modest following for the beloved computer. That is pretty cool that the internet is allowing for that social network to exist from all parts of the world. Video game programming is not any different than being a toy maker or a comedian. You are crafting something for entertaining other people, and that accomplishment is your reward. Everyone's comments have been my reward, and it is always humbling to receive.

CF: Do you have any other "unfinished" or part complete games you will work on

RH: Not a game, specifically, but I do have this generalized "sprite" routine I wrote as a basis for writing VIC 20 games. I hacked its code down to be very efficient for Quikman's maze game, so now I would like to revisit that code to see if lessons learned to date can be applied back into it. If I do that, I am certain it will be implemented into another game.

CF: Do you have any other comments you would like to add

RH: I would like to thank the VIC 20 user community for their warm praises, the emulator project teams that made VICE and MESS possible, and the CC65 project for an excellent 6502 assembler.

QUICKMAN Programmers Delight

QUIKMAN: 1984 revisited
Posted on October 27th, 2008 by Rob

A couple of years ago, I was rejoined with my old Commodore VIC20 software library my brother had kept, some of which were programs I had written and saved using an ordinary audio cassette tape. I decided to try and resurrect the data, because I still have my old VIC20 and datasette drive. But if the data successfully loaded into the computer`s memory, how could I then transfer it over to a modern PC? This problem cannot be new to me, so after a few google searches, I learned of a PC serial cable and some DOS software that would allow me to hook up and use a Commodore floppy drive but not tape.

In steps ebay. I was able to purchase a new-in-the-box VIC 1540 diskette drive for $15 shipped. I then pillaged another $10 for a box of 5-1/4" floppy disks. Fortunately back then, I practiced the good discipline of recording on both sides of each tape, as well as to keep a master copy of all things relevant on an additional tape. So with a freshly cleaned datasette head and a LOT of load attempts, I was able to retrieve all of my saved programs from the 22-year aged tapes.

I forgot about the fun it was to format a 170kb floppy disk. Compared to cassette tape, floppies were amazingly fast. Today, you can download a copy of it faster than you can type RUN on the VIC`s keyboard. Seriously though, using the tapes and floppies was like experiencing it new all over again. And that was kind of fun, though I am glad not to be fussing over such clumsy media with severely limited storage capacity.

I have been able to enjoy the result of this tape librarian nightmare through the use of machine emulation software, specifically from the VICE Team. But a funny thing struck me this past week one of my programs is a game I wrote in 1984 called QUIKMAN. I named it QUIK instead of you-know-who, because earlier that year I wrote my first fully machine language program dubbed QUIKVIC quick as in fast, which it really was on a 1mHz 8-bit 6502 CPU. As it went for me back then, I had this one final week off during the last of my college days, and I decided to spend it writing this game. I believed then that there would never be another opportunity for me to create this game, because I was grooming to be a professional data processing programmer using mainframes my dream of being an arcade game programmer would die, but I needed to try and do this one last time.

I abandoned both my girlfriend and bathing that week, and spent 20-hour days in front of the parlor`s Zenith color TV with my little VIC20 and machine language monitor cartridge. The result of the game came out just fine for something that ultimately loads & runs on a machine with only 3.5kb of memory and 8-colors. As a matter of opinion, my recreation of this arcade megahit is far, FAR superior than what the licensed owners produced for the home computer market of that day. Still, I have always felt I could have done better If only I had the time, and perhaps even the tools. And now after 24-years, that feeling of an incomplete job has resurfaced.What triggered that gut reaction was my accidental discovery of a software project on cc65.org. It stirred up fond memories of my first C compiler, as it was also for the 6502 CPU powering the mighty Commodore 128. But this was not that product of that day. However, it sports a nifty 6502 assembler with preset configurations to compile for Commodore 8-bit computers, including the VIC20! I used to own Merlin 128, too, so I had some pretty high expectations from this tool.

After some light reading of its documentation, I became convinced that I could resurrect my QUIKMAN code into an original assembler source format that could be recompiled back into its original binary format. Turning back to VICE, I loaded QUIKMAN, virtually attached the VICMON cartridge, and had it virtually print (to an ASCII text file) a disassembled listing of its machine code and data.

Over the past 5 days, I have been massaging that listing into newly-formatted assembler source, worthy of today`s coding standards. The goal was to produce an assembler source version that would compile into a binary that was EXACTLY the same as the originally hand-coded machine language version. After my first successful pass at compiling, I simply could not wait to look for deltas I had to boot it up and see if the program ran. Naturally, I was disappointed when the screen turned blue and did nothing. I found the first bug and fixed it, and to my surprise and delight, a version of QUIKMAN was up and running. Way too cool!! I then had the chore to compare the new binary against the old one. It is really important to complete the first objective in making an assembler source that would compile exactly as the original. To accomplish this without too much effort, I turned to the use of two command-line tools: hexdump and meld. By issuing:

hexdump -C quikman.p00 > quikman.old
hexdump -C quikman.prg > quikman.new

I could then compare the two outputs with this graphical diff view:
meld quikman.old quikman.new It highlighted just a few differences, which had no real adverse affects on the program functionality, but I wanted it to be precisely the same. After a few more edits, validated by a clean meld view, the assembler source is now complete. Now I wonder how many more days I`ll go without bathing until I figure I am done with its next revision ?

Programmer`s Delight
Posted on November 1st, 2008 by Rob

Earlier this morning, I had the pleasure of scratching a 24-year old itch which resulted in the 2008 completed version of Quikman for the venerable Commodore VIC20. It was just the other week I wrote about the feat of converting a disassembled listing of my old game, reformatting it into suitable assembler source code so that if compiled, it produces the exact same binary of the original program file.

The days that followed was something I did not expect. With this new assembler, I found that I was able to recall what each part of the program did, reorganize the code and data in a manner that was much more memory efficient (a stock VIC20 only has 3583 bytes for loading a program), and remove all unnecessary data and NOP (no operation) instructions that were useful then for reserving bytes for future machine code. This internal house-cleaning did not change any part of the game, but it did free up 88-bytes and assure me that the assembly process was in a manageable, relocatable address state. That was all I needed to get motivated to make the changes I could not do in 1984 without these tools no excuses for me now not to get it done right! This is what got done:

- fix the character cell-based graphics to reduce flickering and color “noise”
- fix the number of lives remaining in the status bar
- fix a software interrupt that can cause a joystick poll to occur during quikman`s move
- reverse monsters direction after swallowing a powerpill
- any monster released from the cage remains aggressive, even during powerpill play
- add a startup demo mode
- improve quikman`s death sequence
- improve completed level sequence
= add a cheat key to advance to the next fruit level

A bonus feature from all the above fixes and optimizations allowed me to integrate progressive playing speed as the fruit levels advance. The player will require teenager reflexes to exceed the apple level without the use of the cheat key (ESC). I decided to make the demo mode look like an old Atari VCS home video game a steady flicker from its player / missile graphics while cycling the screen`s color palette. This was quite intentional, because that look definitely invokes an authentic retro-arcade gaming feel about it. The final result of this program is that it completely uses not just occupies all 3583-bytes of available RAM. Fortunately, it was enough for my skill and determination to complete my objective in making a quality Pac-Man clone for a stock Commodore VIC20. Too bad it took 24-years to complete, because there was no other clone of that era that I ever played which came this close to the original.

After playing a few rounds of my new found toy, an overwhelming mix of satisfaction and pride took hold of my senses which after a short time, I found myself a bit physically shaken as if I were nervous. Talk about a coding session hangover ha! The only other time I felt that, but it was more potent, was witnessing the birth of my daughter, Erin. It is that feeling from accomplishment in software invention that I describe as programmer`s delight.

P.S., this program runs perfectly on another fine machine emulator, mess, and uses a joystick. It uses the PC`s F4 key for the VIC`s F7 key (which is accurate as the VIC only had four function keys, but used SHIFT to yield eight) and the HOME key as VIC`s STOP key for cheating. From the command-line, simply: mess vic20 -quik quikman2k8.prg And you will need to type RUN after a few seconds from its READY prompt.

;Quikman for Commodore VIC20

written by Robert Hurst <robert@hurst-ri.us>

```
; Quikman for Commodore VIC20
; written by Robert Hurst <robert@hurst-ri.us>
; using Commodore VICMON (SYS 45056)
; original version: Fall 1984
;
; disassembled: 23-Oct-2008
; re-assembled: 27-Oct-2008
;
; to assemble this source using cc65.org project:
;   ca65.exe --cpu 6502 --listing quikman.s
;   ld65.exe -C doc/vic20.cfg -o quikman.prg quikman.o
;
; to run the binary using viceteam.org project:
;   xvic -memory none -ntsc -sound -joydev1 2 -autostart quikman.prg
;
; pertinent VIC20 symbols
JIFFYH          = $A0
JIFFYM          = $A1
JIFFYL          = $A2       ; jiffy-clock low byte value
SCRNPAGE        = $0288     ; screen memory page (unexpanded = $1E)
CTRLCSHIFT      = $028D     ;  keyboard  flag: control/commodore/shift
CASSBUFF = $033C            ; cassette buffer
VIC             = $9000     ; start of video interface chip registers
CHROUT          = $FFD2
GETIN           = $FFE4
;
; my symbol / memory map
PPILLTIMER      = $10       ; powerpill effectiveness timer
FRUITTIMER      = $39       ; 0 - 242
FRUITFLAG       = $3A       ; zero or non-zero, if fruit has been activated
PPILLFLAG       = $3B       ; just ate a powerpill this turn (0=no)
CHOMP           = $3C       ; pointer into sound effect for fruit and fleeing monsters
CHEWING         = $3D       ; flag whether quikman just ate a dot or not
DIGIT           = $3E       ; award points at this digit place
POINTS          = $3F       ; how many points scored
OLDDIR          = $40       ; direction sprite was last moving in
NEWDIR          = $41       ; direction sprite wants to take, if valid by MAZEMOVE
JOYVAL          = $42       ; last joystick read value
QMANDIR         = $43       ;  quikman's  current  direction (0=right,1=down,2=left,3=up)
LIVES           = $44       ; 0 - 3
MAZE            = $45       ; maze color: 3 (cyan) or 6 (blue)
FLASHPILL       = $46       ; powerpill blink counter (0-30)
BANNERTIMER     = $47       ; time for banner to display
BANNERFLAG      = $48       ; 0=author, 1=instructions
FRUITLEVEL      = $49       ; 0 - 12
DOTS            = $4A       ; 0 - 174
PENALTY         = $4B       ; $4B-$4E monsters are free-to-roam flag
;               = $4F       ; $4F-$52 monsters current direction (0=right,1=down,2=left,3=up)
;               = $53       ; $53,$55,$57,$59 monster's knowledge of quikman's "X" coord was
;               = $54       ; $54,$56,$58,$5A monster's knowledge of quikman's "Y" coord was
;               = $61       ; $61-$64 monster array for its next best move
;               = $69       ; temporary var
FLEEINGSCORE= $70           ; fleeing monster score: 2, 4, 8, 16
;               = $FC       ; $FC,$FD is screen cell pointer for sprite's "home" position
;               = $FE       ; $FE,$FF is color cell pointer for same
;
; program indirects (my sprite registers)
SPRITE          = $02A1     ; bitmask 0-7 controls sprite on/off
SPRITEX         = $02A2             ; $02A2,A4,A6,A8,AA,AC,AE,B0 each "X" coordinate
SPRITEY         = $02A3             ; $02A3,A5,A7,A9,AB,AD,AF,B1 each "Y" coordinate
SPRITECLR       = $02B2             ; $02B2-$02B9 color
SPRITEIMG1      = $02BA             ; $02BA-$02C1 low-byte of SPRITE image
SPRITEIMG2      = $02C2             ; $02C2-$02C9 hi-byte of SPRITE image
SPRITELAST      = $02CC             ; $02CC-$02DC keep last state of SPRITE registers
SAVEBACK        = $02DD     ; $02DD-$02FC keep what's under the sprite's 2x2 matrix
;
EXTRAQMAN       = $03E0     ; bonus quickman flag (0=unused)
;
; other constants
FRUITCELL       = $1F1B     ; screen cell address of fruit
FRUITCELLCLR= $971B         ; color cell address of fruit
;
; uses standard VIC20 (unexpanded)
                .segment "STARTUP"
;
; starting load address:
; LOAD "QUIKMAN.PRG",8,1
                .word       $1001
;
; 0 SYS4110
BASIC:      .byte       $0B, $10, $00, $00, $9E, $34, $31, $31, $30, $00, $00, $00
;
; Main entry point into the game
START:
                JSR REDRAW
RESTART:
                NOP
                NOP
                NOP
                JSR INITVARS
                LDA #$0E            ; black / blue
                STA VIC+$0F         ; background / border color
                LDA #$08            ; lock uppercase / graphic set
                JSR CHROUT
                LDX #$FF            ; set for $1C00
                STX VIC+$05         ; use programmable char set
                INX                 ; mute
                STX VIC+$0E         ; aux color / volume
                LDA #$80+$15        ;  set  for  videoram  @ $1E00 with 21-columns
                STA VIC+$02         ; video matrix address + columns
                LDA #$B0            ; $B0 = 10110000 = 24 rows + 8x8 height
                STA VIC+$03         ; rows / character height
                SEI
                LDX #<BACKGROUND        ;$4D
                LDY #>BACKGROUND        ;$10
                STX $0314
                STY $0315
                CLI
                LDA #$06            ; blue
                STA MAZE
@loop1:     JSR GETIN               ; get keyboard
                CMP #$88            ; got F7 ?
                BNE @loop1          ; try again ...
                BEQ RESET           ; game starts!
;
;($104D)
BACKGROUND:
                LDA JIFFYL
                AND #$07
                BEQ @skip1
                LDA DOTS
                CMP #$AE            ; All 174 dots been eaten
                BNE @skip2
@skip1:     JMP MAZEPAINT
@skip2:     JMP EFFECTS             ; which will return here (JSR)
;
;($105F)
SR105F:
                JSR SR1500
                NOP
                NOP
                NOP
                JSR SR14A0          ; sound effects
                JSR GAMEOVER        ; a routine in custom character area
```

```
                JMP $EABF       ; jump to hardware IRQ
;
;($106E)
SR106E:
                LDA #$09
                STA CHOMP
                LDA JIFFYL
@loop1:     CMP JIFFYL
                BEQ @loop1      ; wait up to a jiffy
                TYA
                LSR
                TAX
                LDA CAGEDATA,X  ; load waiting room time
                LSR
                LSR
                STA PENALTY,X   ; monster is waiting
                RTS
;
; game reset
RESET:
                LDA #$FF        ; -1 will become 0 at start
of "next" level
                STA FRUITLEVEL
                LDA #$03        ; start with 3-lives
                STA LIVES
                LDX #$00        ; reset score
@loop1:     LDA #$B0            ; each digit to "0"
                STA SCORE,X     ; into savebuffer
                INX             ; do next digit
                CPX #$06        ;all 6 of them
                BNE @loop1
;
;($1097)
RESETSND:
                LDA #$00
                TAX
@loop1:     STA VIC+$0A,X       ; reset sound channels
                INX
                CPX #$04
                BNE @loop1
                STA SPRITE      ; turn off all sprites
                JSR SPRITES     ; redraw sprites
                LDA #$0F
                STA VIC+$0E     ; volume mute
                JSR REDRAW      ; refresh screen
;
;($10B0)
RESETCHR:
                LDX #$05
@loop1:     LDA QUIKMANCLR-1,X  ; reset monsters starting
colors
                STA SPRITECLR-1,X ; into their sprite color reg-
isters
                DEX
                BNE @loop1
@loop2:     LDA #$78            ; reset monsters as chasing
                STA SPRITEIMG1+1,X ; into their sprite register
                LDA #$1D        ; reset monsters character
                STA $02C3,X     ; into their sprite register
                INX
                CPX #$04
                BNE @loop2
                LDX LIVES       ; paint lives remaining
@loop3:     LDA #$2D            ; quikman character
                STA $1FE2,X     ; bottom-left of screen
                LDA #$07        ; use yellow
                STA $97E2,X     ; and paint it
                DEX
                NOP
                NOP
                BNE @loop3
                LDY FRUITLEVEL
@loop4:     CPY #$0C            ; are we at the last level
(key)?
                BCC @skip1
                LDY #$0C        ; only keys remain
@skip1:     LDA FRUIT,Y         ; fruit character
                STA $1FF1,X     ; bottom right of screen
                LDA FRUITCLR,Y  ; get its color
                STA $97F1,X     ; and paint it
                CPY #$00        ; did we paint the cherry
yet?
                BEQ SR1100      ; if so, we're done
                INX
                STX $FF
                LDA FRUITLEVEL
                SEC
                SBC $FF
                TAY
                CPX #$07        ; no more than 7 fruits to
display
                BNE @loop4
;
;($1100)
SR1100:
                LDX #$00
                STX SPRITE      ; turn off all sprites
                STX QMANDIR     ; start off going RIGHT
                STX JOYVAL      ; preload last joystick value
as going RIGHT
                JSR SPRITES     ; redraw sprites
                LDX #$00
@loop1:     LDA STARTPOS,X      ; reset each sprite starting
position
                STA SPRITE+1,X
                INX
                CPX #$0A        ; 5 sprites per X,Y coordi-
nate pair
                BNE @loop1
                LDA #$1C        ; always point quikman
sprite image
                STA SPRITEIMG2  ; to $1CE8 ... stupid?
                LDA #$E8
                STA SPRITEIMG1
                LDA #$1F        ; turn on sprites 0-4
                STA SPRITE
                JSR SPRITES     ; redraw sprites
                LDX #$00
@loop2:     LDA $C378,X         ; print READY from ROM
                AND #$BF
                ORA #$80
                STA $1F19,X
                LDA #$07        ; make it yellow
                STA $9719,X
                INX
                CPX #$05
                BNE @loop2      ; how geeky is that?
                LDA JIFFYL
                STA $FF         ; save this jiffy
@loop3:     LDA JIFFYL
                SEC
                SBC $FF
                CMP #$80        ; wait 2+ seconds
                BNE @loop3
                LDX #$00
                LDA #$20        ; erase READY
@loop4:     STA $1F19,X
                INX
                CPX #$05
                BNE @loop4
;
; we're ready to play now!
                JSR ZEROVARS
                NOP
                NOP
                NOP
;
;($1160)
PLAYLOOP:
                LDA #$00
                STA $00
                STA $01
                LDA QMANDIR
                STA OLDDIR      ; save last direction quik-
man was going in
                LDA JOYVAL
                STA NEWDIR      ; do the same for the joy-
stick
                JSR MAZEMOVE
                BCS @skip1      ; is the direction valid?
                LDA JOYVAL      ; nope
                STA QMANDIR     ; request quikman to move
in direction of joystick
                CLC
                BCC @skip2
@skip1:     LDA QMANDIR
                STA NEWDIR
                JSR MAZEMOVE    ; keep the current direction
going?
@skip2:     LDA SPRITEX
                BNE @skip3      ; is quikman at end of tun-
nel left?
                LDA #$9E
                STA SPRITEX     ; put quikman at beginning
of tunnel right
```

```
@skip3:     CMP #$A0                    ; is quikman at end of tun-
nel right?
                    BNE @skip4
                    LDA #$00
                    STA SPRITEX         ; put quikman at beggining
of tunnel left
@skip4:     LDX #$00
                    LDA SPRITEX
                    AND #$07
                    CMP #$04            ; is quikman in the middle
of a left/right cell?
                    BNE @skip5
                    LDA QMANDIR
                    EOR #$02
                    TAY
                    INX
                    BNE @skip6
@skip5:     LDA SPRITEY
                    AND #$07
                    CMP #$04            ; is quikman in the middle
of an up/down cell?
                    BNE @skip6
                    LDY QMANDIR
                    CPY #$01
                    BEQ @next
                    LDY #$00
@next:      INX
@skip6:     CPX #$00
                    BEQ YUMMY
                    LDA SAVEBACK,Y      ; retrieve the character
from quikman's saveback buffer
                    TAX
                    LDA #$20
                    STA SAVEBACK,Y      ; replace the cell quikman
is on with an empty space
                    CPX #$1E            ; is it a dot?
                    BNE POWERUP
                    LDA #$01
                    STA POINTS          ; score 1
                    STA CHEWING         ; quikman has to chew this
dot, monsters keep movin'
                    LDA #$0A            ; score it @ 10-point digit
                    STA DIGIT
;
;($11D4)
EATING:     INC DOTS                    ; ate a dot, account for it
                    LDA DOTS
                    CMP #$AE            ; are all dots eaten?
                    BNE YUMMY
;===         end of level ===
                    JSR SPRITES         ; redraw sprites
                    LDA #$00
                    STA $FF
@Loop:      LDA #$03                    ; turn maze cyan
                    STA MAZE
                    LDY #$00
                    LDX #$00
@loop5:     INX                         ; bad wait loop
                    BNE @loop5
                    INY
                    BNE @loop5
                    LDA #$06
                    STA MAZE            ; turn maze blue
@loop6:     INX                          bad wait loop
                    BNE @loop6
                    INY
                    BNE @loop6
                    INC $FF
                    LDA $FF
                    CMP #$04
                    BNE @Loop
                    JMP NEXTLEVEL       ; complete
;
;($1206)
POWERUP:
                    CPX #$22
                    BCC @skip1          ; is X < 34 ?
                    CPX #$2A            ; no, is X >= 42 ?
                    BCS YUMMY           ; ate a piece of fruit?
                    TXA                 ; YUMMY!
                    SEC
                    SBC #$22            ; strip off char code for
score index
                    TAX
                    LDA FRUITSCORE,X
                    STA POINTS          ; award points
                    LDA #$09
                    STA DIGIT           ; in hundreds
                    STA CHOMP
                    CLC
                    BCC YUMMY
@skip1:     CPX #$1F                    ; ate a powerpill?
                    BNE YUMMY
                    LDA #$05
                    STA POINTS          ; award 5-points
                    LDA #$0A
                    STA DIGIT           ; score @ 10-digit
                    STA PPILLFLAG
                    BNE EATING          ; powerpills are dots on
steroids, account for it
;
;($1231)
YUMMY:
                    LDA FRUITFLAG
                    BNE @skip3          ; is fruit already on dis-
play?
                    LDA DOTS
                    CMP #$4B            ; has the 75th dot been eat-
en?
                    BEQ @skip1
                    CMP #$7D            ; has the 125th dot been
eaten?
                    BNE @skip3
@skip1:     LDX FRUITLEVEL              ; prepare thy bonus
                    CPX #$0C            ; reach the last level?
                    BCC @skip2
                    LDX #$0C            ; only the key is left, and it
leaves a bad metallic after-taste
@skip2:     LDA FRUIT,X
                    STA FRUITCELL       ; display fruit
                    LDA FRUITCLR,X
                    STA FRUITCELLCLR
                    LDA #$F2            ; 242-moves and counting
                    STA FRUITTIMER      ; reset fruit timer
                    STA FRUITFLAG
@skip3:     LDA FRUITTIMER              ; fruit is on display
                    BEQ @skip4          ; nothing to do
                    DEC FRUITTIMER      ; remove a tick
                    BNE @skip4          ; there is still time left
                    LDA #$20            ; time's up!
                    STA FRUITCELL       ; no more fruit
@skip4:     LDA DOTS
                    CMP #$4C            ; has the 76th dot been eat-
en?
                    BEQ @skip5
                    CMP #$7E            ; has the 126th dot been
eaten?
                    BNE @skip6
@skip5:     LDA #$00
                    STA FRUITFLAG       ; more fruit on this level
;
@skip6:     LDA PPILLFLAG
                    BEQ @skip8          ; just swallowed a power-
pill?
                    LDA FRUITLEVEL
                    ASL                 ; multiply 8
                    ASL
                    ASL
                    STA $FF             ; shrink powerpill effec-
tiveness per level
                    LDA #$F8
                    SEC
                    SBC $FF
                    STA PPILLTIMER      ; set powerpill timer
                    LDY #$00
@loop1:     LDX PENALTY,Y
                    BNE @skip7          ; is monster waiting in cage
already?
                    LDA #$06            ; no, make monster blue
                    STA SPRITECLR+1,Y
                    LDA #$80            ; make monster fleeing (0)
                    STA SPRITEIMG1+1,Y
@skip7:     INY
                    CPY #$04
                    BNE @loop1
@skip8:     LDA PPILLTIMER
                    BEQ CHASING         ; are monsters fleeing?
                    CMP #$41            ; yes ... but are they
                    BCS @skipA          ; getting confidence back?
                    AND #$08            ; yes, let's warn quikman
                    BEQ @skipA
                    LDY #$00
@loop2:     LDA SPRITEIMG1+1,Y
                    CMP #$80            ; is monster fleeing?
                    BNE @skip9
                    LDA SPRITECLR+1,Y
```

```
                EOR #$07        ; flash white / blue
                STA SPRITECLR+1,Y
@skip9:  INY
                CPY #$04
                BNE @loop2
@skipA:  DEC PPILLTIMER         ; drain powerpill
                BNE SR12D3      ; is there still power left?
;
;($12C1)
; restore all monsters to their default colors and chase mode
CHASING:
                LDY #$00
@loop:   LDA MONSTERCLR,Y
                STA SPRITECLR+1,Y
                LDA #$78        ; restore monster chasing
image (/)
                STA SPRITEIMG1+1,Y
                INY
                CPY #$04        ; four monsters
                BNE @loop
;
;($12D3)
SR12D3:
                JSR PAUSING     ; is the action pausing?
                NOP
                NOP
                LDX #$00
                LDA PPILLFLAG
                BEQ @skip1      ; just swallowed a power-
pill?
                LDA #$00        ; yes ...
                STA $3B         ; account for that action
                LDA #$02        ; start scoring @ 200-
points
                STA FLEEINGSCORE
@skip1:  LDY #$00
SR12E8:  LDA SPRITEX
                CMP SPRITEX+2,Y
                BNE @skip3
                LDA SPRITEY
                SEC
                SBC SPRITEY+2,Y
                BCS @skip2
                EOR #$FF
@skip2:  CMP #$05
                BCS @skip3
                LDX #$FF
                BNE ENGAGED     ; is quikman engaged
with a monster?
@skip3:  LDA SPRITEY
                CMP SPRITEY+2,Y
                BNE NEXTKISS
                LDA SPRITEX
                SEC
                SBC SPRITEX+2,Y
                BCS @skip4
                EOR #$FF
@skip4:  CMP #$05
                BCS NEXTKISS
                LDX #$FF
                BNE ENGAGED     ; is quikman engaged
with a monster?
;
;($131E)
NEXTKISS:
                INY
        ; increment for next
                INY
        ; X,Y coord pair check
                CPY #$08
                BCC SR12E8
                BCS MONSTERS    ; quikman is still freely
running!
;
;($1326)
ENGAGED:
                TYA
                LSR
                TAX
                LDA SPRITEIMG1+1,X
                CMP #$80        ; is monster fleeing?
                BEQ CAUGHTONE   ; ahah!
                LDY #$00        ; ooops... quikman got it
                LDX #$00
@loop2:  INX                    ; bad wait loop
                BNE @loop2
                INY
                BNE @loop2
                DEC LIVES
                BNE DEAD        ; any lives remaining?
                JMP RESTART     ; game over
;
;($1341)
DEAD:
                LDX LIVES
                LDA #$20
                STA $1FE3,X     ; erase avatar
                JMP DEATH
;
;($134B)
CAUGHTONE:
                LDA #$09
                STA DIGIT       ; in hundreds
                LDA FLEEINGSCORE
                STA POINTS      ; fleeing monster score
                ASL
        ; next is worth x2 bonus
                STA FLEEINGSCORE
                LDA #$78
                STA SPRITEIMG1+1,X ; reset monster as chasing
                LDA MONSTERCLR,X
                STA SPRITECLR+1,X
                TXA
                ASL
                TAX
                LDA #$50        ; reset "X" coord in cage
                STA $02A4,X
                LDA #$58        ; reset "Y" coord in cage
                STA $02A5,X
                NOP
                JSR SR106E
                CLC
                BCC NEXTKISS    ; is there another monster
here?
;
;($1375)
MONSTERS:
                SEI
        ; lock-out any background changes
QUIKMAN:
                LDA PPILLTIMER
                AND #$01        ; during fleeing mode, all
monsters move
                BNE CONT        ; every-other frame, re-
gardless
                LDA #$01
                STA $00         ; start with monster #1
;
;($1380)
DOMONSTER:
                ASL
                STA $01
                LDY $00
                LDX $4A,Y
                BEQ ITMOVES     ; is this monster free to
roam?
                DEX
        ; no, countdown to freedom
                STX $4A,Y
;
;($138C)
NEXTMONSTER:
                INC $00         ; process next monster
                LDA $00
                CMP #$05        ; all 5-sprites processed?
                BCC DOMONSTER
CONT:    JSR $1CCF              ; issues
                JMP PLAYLOOP
;
ITMOVES:
                LDX $01         ; get pairing index
                LDA SPRITEX,X
                CMP #$50
                BNE @skip1
                LDA SPRITEY,X
                CMP #$58
                BNE @skip1      ; is monster in cage
($50,$58 coord) doorway ?
                LDA #$57        ; could have just used
DEC SPRITEY,X instead
                STA SPRITEY,X   ; move it a pixel UP to
force it through the closed door
                LDX #$03
```

```
                    STX $4E,Y       ; make direction UP to get
out of cage
                    BNE @skip3
@skip1:   LDA SPRITEX,X
                    BNE @skip2      ; is monster against the
left-side of the tunnel?
                    LDX $00
                    STA $4E,X       ; force a change of direc-
tion to the right
@skip2:   CMP #$9F                  ; is monster against the
right-side of the tunnel?
                    BNE @skip3
                    LDX $00
                    LDA #$02
                    STA $4E,X       ; force a change of direc-
tion to the left
@skip3:   LDY #$00
                    LDX #$00
@loop1:   STX $61,Y                 ; preset move priority as
0=right,1=down,2=left,3=up
                    INY
                    INX
                    CPX #$04
                    BCC @loop1
                    LDY $01         ; start of monster's calcu-
lated move
                    LDA SPRITEX,Y
                    AND #$07
                    BEQ @skip4      ; is monster horizontally
aligned with a screen cell?
                    LDA SPRITEY,Y
                    AND #$07
                    BNE @skip5      ; is monster vertically
aligned with a screen cell?
@skip4:   JSR SR1457                ; yes, check to see if a di-
rection change is in its future
                    CLC
                    BCC @skip6
@skip5:   LDX $4E,Y                 ; not in a position to make
a direction change,
                    STX $61         ; so just keep monster go-
ing in its current direction
@skip6:   LDY #$00
                    STY $04
@loop2:   LDX $61,Y
                    TXA
                    LDX $00
                    EOR $4E,X
                    CMP #$02
                    BEQ @skip7      ; don't allow monsters to
reverse direction on their own
                    LDX $61,Y
                    STX NEWDIR
                    LDY $00
                    LDX $4E,Y
                    STX OLDDIR
                    JSR MAZEMOVE    ; validate
                    BCC MAKEMOVE    ; is this a good move?
@skip7:   INC $04
                    LDY $04
                    CPY #$04
                    BNE @loop2
                    JMP NEXTMONSTER
                    NOP                                 ;
legacy reasons
;
; preload $61-$64 with "best" moves this monster can make
; to give quikman the kiss of death
;($1418)
AI:
                    LDX $01
                    LDA $51,X       ; retrieve this monster's
"X" knowledge where quikman was
                    SEC
                    SBC SPRITEX,X
                    BCS @skip1
                    LDY #$02
                    STY $61         ; LEFT is best
                    LDY #$00
                    STY $64         ; RIGHT is worst
                    BEQ @skip2
@skip1:   LDY #$00
                    STY $61         ; RIGHT is best
                    LDY #$02
                    STY $64         ; LEFT is worst
@skip2:   LDA $52,X                 ; retrieve this monster's
"Y" knowledge where quikman was
                    SEC
                    SBC SPRITEY,X
                    BCS @skip3
                    LDY #$03
                    STY $62                     ; UP is 2nd
best
                    LDY #$01
                    STY $63         ; DOWN is 3rd best
                    RTS
@skip3:   LDY #$01                  ; DOWN is 2nd best
                    STY $62
                    LDY #$03        ; UP is 3rd best
                    STY $63
                    RTS
;
;($144E)
MAKEMOVE:
                    LDY $00         ; commit to this move
                    LDX NEWDIR
                    STX $4E,Y       ; save as monster's current
direction
                    JMP NEXTMONSTER
;
; prioritize monster move, based upon its current location in respect to
; its knowledge where quikman was considered last.
;$(1457)
SR1457:
                    JSR AI          ; make monsters with vary-
ing intelligence
                    LDX $01
                    LDA $51,X
                    SEC
                    SBC SPRITEX,X
                    BCS @skip1
                    EOR #$FF
@skip1:   STA $69
                    LDA $52,X
                    SEC
                    SBC SPRITEY,X
                    BCS @skip2
                    EOR #$FF
@skip2:   CMP $69
                    BCC @skip3      ; can monster improve up-
on order of choices?
                    NOP
                    LDX $61         ; swap 1st & 2nd choices
                    LDY $62
                    STX $62
                    STY $61
                    LDY $63         ; swap 3rd & 4th choices
                    LDX $64
                    STY $64
                    STX $63
@skip3:   LDA $10
                    BEQ @fini       ; monsters are in chase
mode
                    LDX #$00
@loop1:   LDA $61,X
                    PHA
                    INX
                    CPX #$04
                    BNE @loop1
                    LDX #$00
@loop2:   PLA
                    STA $61,X       ; reverse logic when in flee
mode
                    INX
                    CPX #$04
                    BNE @loop2
@fini:    RTS
;
; some sound effects and extras
;($14A0)
SR14A0:
                    LDA CHEWING
                    BEQ @skip1
                    LDA #$91        ; start with an odd frequen-
cy
                    STA VIC+$0C     ; ignite a voice
@skip1:   LDA #$00                  ; dot is swallowed
                    STA CHEWING
                    LDA VIC+$0C
                    BEQ @next1      ; is this voice mute?
                    LDA VIC+$0C
                    AND #$01
                    BEQ @skip3                  ; is it even?
                    LDA VIC+$0C
                    CLC
                    ADC #$10        ; increase tone
```

```
                    CMP #$F1
                    BCC @skip2          ; is voice too high?
                    SEC
                    SBC #$01            ; make it even
@skip2:   STA VIC+$0C
                    CLC
                    BCC @next1          ; goto next effect
@skip3:   LDA VIC+$0C
                    SEC
                    SBC #$10            ; drain tone
                    STA VIC+$0C
@next1:   LDX CHOMP
                    BEQ @skip4
                    LDA SNDBIT,X        ; load tone data
                    STA VIC+$0B
                    DEC CHOMP
@skip4:   LDA VIC+$0E
                    BNE @next2
                    STA EXTRAQMAN       ; reset bonus when volume
is mute
@fini:    RTS
@next2:   LDA EXTRAQMAN
                    BNE @fini
                    LDA SCORE+1
                    CMP #$B1            ; did quikman just score
10,000-points?
                    BNE @fini
                    STA EXTRAQMAN
                    INC LIVES           ; reward
                    LDA #$09
                    STA CHOMP           ; make some noise
                    RTS
;
;($1500)
SR1500:
                    LDY #$00
SR1502:   STY $03F0
                    LDA $03F0
                    ASL
                    STA $03F1
                    ASL
                    STA $03F2
                    CLC
                    BCC SR1567          ; go there to check this,
then come back (JSR)
SR1513:   LDA CAGEDATA,Y
                    BEQ AWARENESS       ; is monster "smart"?  Red
one is ...
                    CMP JIFFYL          ; no, so check as often as it
waits
                    BEQ AWARENESS       ; is its wait time equal to
the jiffy clock?
SR151C:   LDY $03F0
                    INY
                    CPY #$04
                    BNE SR1502
                    RTS
;
; update this monster's awareness to where quikman is
;($1525)
AWARENESS:
                    LDY $03F1
                    LDX SPRITEX
                    STX $53,Y
                    LDX SPRITEY
                    STX $54,Y
                    CLC
                    BCC SR151C
;
;($1535)
NEXTLEVEL:
                    JSR INITVARS
                    LDX #$04
@loop1:   LDA $4A,X
                    LDY FRUITLEVEL
                    INY
@loop2:   LSR
                    DEY
                    BNE @loop2
                    STA $4A,X           ; after each level, the mon-
sters dispatch quicker
                    DEX
                    BNE @loop1
                    JMP RESETSND
;
;($154B)
RESURRECT:
                    JSR INITVARS
                    JMP RESETCHR
;
;($1551)
PAUSING:
                    LDA CTRLCSHIFT      ; is the player holding
down any
                    BNE PAUSING         ; control, commodore, shift
key(s)?
                    RTS
;
;($1557)
                    .byte       $00, $00, $00, $00, $00, $00
SNDBIT:   .byte     $00, $00, $C0, $B8, $B0, $A8, $B0, $B8, $C0, $C8
;
;($1567)
SR1567:
                    LDY $03F2
                    LDA DOTS
                    CMP #$AA            ; make them all "smart"
when nearing the end
                    BCS AWARENESS
                    BCC SR1513
;
;($1572)
                    .byte       $00, $00, $00, $00
;
;($1576)
INITVARS:
                    LDY #$00
@loop:    LDX CAGEDATA,Y
                    STX PENALTY,Y
                    INY
                    CPY #$10
                    BNE @loop
                    RTS
;
;($1583)   zero $3A - $43
ZEROVARS:
                    LDY #$3A
                    LDX #$00
@loop:    STX $00,Y
                    INY
                    CPY #$44
                    BNE @loop
                    RTS
;
;($158F)
ADDSCORE:
                    LDY POINTS
                    BNE @skip1
                    RTS
@skip1:   LDX DIGIT
@loop1:   LDA ScreenData,X
                    CMP #$B9            ; reach "9" ?
                    BEQ @skip2
                    INC ScreenData,X    ; ding!
                    DEY
                    BNE @skip1
                    RTS
@skip2:   LDA #$B0
                    STA ScreenData,X    ; wrap to "0"
                    DEX                 ; and increment next order
                    CLC
                    BCC @loop1
;
;$(15AD)
JOYSTICK:
                    LDX JOYVAL          ; recall last joystick value
                    LDA #$00
                    STA $9113
                    LDA #$7F
                    STA $9122
                    LDA $9120
                    AND #$80            ; JOY 3
                    BNE @skip1
                    LDX #$00
@skip1:   LDA #$FF
                    STA $9122
                    LDY $9111
                    TYA
                    AND #$08
                    BNE @skip2
                    LDX #$01
@skip2:   TYA
                    AND #$10
                    BNE @skip3
```

```
                LDX #$02
@skip3:  TYA
                AND #$04
                BNE @skip4
                LDX #$03
@skip4:  STX JOYVAL                     ; save
                RTS
;
;($15E2) Refresh screen
REDRAW:
                INC FRUITLEVEL
                LDA #$93                ; Shift-HOME is clear-
screen
                JSR     CHROUT          ; print it
                LDX #$14                ; skip 1st row -- scoring
information
                LDY #$00
@loop:   LDA ScreenData,X
                STA $1E00,X
                LDA ScreenData+$0100,Y
                STA $1F00,Y
                INY
                INX
                BNE @loop
                STX DOTS                ; and no dots are eaten
(yet)
                RTS
;
;($1600)
BANNERMSG:
                                        ;©1984 ROBERT HURST
                .byte   $3F, $B1, $B9, $B8, $B4, $A0, $92,
$8F, $82, $85, $92, $94, $A0, $88, $95, $92, $93, $94
                                        ;PRESS 'F7' TO PLAY
                .byte   $90, $92, $85, $93, $93, $A0, $A7,
$86, $B7, $A7, $A0, $94, $8F, $A0, $90, $8C, $81, $99
;
;($1624)         cherry, strawberry, 2-peach, 2-apple, 2-pineapple,
2-tbird, 2-bell, key
FRUIT:   .byte   $22, $23, $24, $24, $25, $25, $26, $26, $27, $27,
$28, $28, $29
;
;($1631)         red, red, 2-yellow, 2-red, 2-green, 2-magenta, 2-yel-
low, cyan
FRUITCLR:
                .byte   $02, $02, $07, $07, $02, $02, $05, $05,
$04, $04, $07, $07, $03
;($163E)
STARTPOS:
                .byte   $50, $88, $50, $48, $50, $58, $60, $58,
$40, $58
;
;($1648)
;animate the quikman
ANIMQMAN:
                LDX #$50                ; closed mouth
                LDA JIFFYL
                AND #$04                ; flip every 4-jiffies
                BEQ @skip1
                LDA QMANDIR             ;  take
0=right,1=down,2=left,3=up value
                ASL                     ; multiply by 8 to get ad-
dress
                ASL
                ASL
                CLC
                ADC #$58                ; add base offset
                TAX
@skip1:  LDY #$00
@loop1:  LDA $1D00,X                    ; copy a quikman image
                STA $1CE8,Y             ; into its sprite
                INX
                INY
                CPY #$08
                BNE @loop1
                RTS
;
; if move is valid, carry flag will be clear on return
;($1668)
MAZEMOVE:
                LDY $01                 ; get X,Y coord index
                LDA OLDDIR              ; get the last direction mov-
ing
                AND #$01                ; mask UP/DOWN
                BEQ @skip1              ;  is  direction
LEFT/RIGHT?
                INY                     ; no, then fetch the "Y" co-
ordinate
@skip1:  LDA SPRITEX,Y          ; get one of sprite's coord
                AND #$07
                BEQ MAZEANY             ; at a crossroad?  check
move in any 4-directions
                LDA NEWDIR
                CMP OLDDIR
                BEQ MYMOVE              ; still want to move in the
same direction?
                EOR OLDDIR
                CMP #$02
                BEQ MYMOVE              ; is this a reverse direction
request?
                SEC                     ; no new move made
                RTS
;
;($1686)
MAZEANY:
                JSR SPRITEPREP
                LDA $FD
                SEC                     ;  this  should  point  to
>ScreenData
                SBC #$07                ; reset screen hi-byte back
into saved maze data
                STA $FD
                LDX NEWDIR
                CPX #$02
                BCS @skip2              ; is X (2=left) or (3=up)?
                LDA $FC                 ; no
                CLC
                ADC PEEKAHEAD,X         ;  look  (0=right)  or
(1=down)
                BCC @skip1
                INC $FD
@skip1:  STA $FC
                CLC
                BCC @skip4              ; go validate
@skip2:  LDA $FC
                SEC
                SBC PEEKAHEAD-2,X
                BCS @skip3              ; look (2=left) or (3=up)
                DEC $FD
@skip3:  STA $FC
@skip4:  LDY #$00               ; validate
                LDA ($FC),Y
                CMP #$31                ; is this direction into a
maze wall?
                BCC MYMOVE              ; good move?
                RTS
;
; continue this sprite's move in whatever is loaded in NEWDIR
;($16BA)
MYMOVE:
                LDA NEWDIR
                ASL                     ; 0=0, 1=2, 2=4, 3=6
                TAX
                LDY $01
                LDA INERTIA,X
                CLC
                ADC SPRITEX,Y
                STA SPRITEX,Y
                LDA INERTIA+1,X
                CLC
                ADC SPRITEY,Y
                STA SPRITEY,Y
                CLC
                RTS
;
;($16D6)
PEEKAHEAD:      .byte   $01, $15
INERTIA:  .byte   $01, $00, $00, $01, $FF, $00, $00, $FF
FRUITSCORE:     .byte   $01, $03, $05, $07, $0A, $14, $1E, $32
                .byte   $00, $00, $00, $00, $00,
$00, $00, $00
;($16F0)
CAGEDATA:       .byte   $00, $30, $70, $F0
                .byte   $02, $03, $02, $00, $A0,
$18, $A0, $18, $A0, $18, $A0, $18
;
; Maze data ($1700 - $18FF)
; Screen size: 24-rows by 21-columns
ScreenData:
SCORE    = $1706
                .byte   $93, $83, $8F, $92, $85, $BA, $B0,
$B0, $B0, $B0, $B0, $B0, $B0, $A0, $A0, $A0, $B0, $B2, $B0, $B0, $B0, $B0
```

```
                        .byte   $37, $3A, $3A, $3A, $3A, $3A, $3A,
$3A, $3A, $3A, $3D, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $38
                        .byte   $39, $1E, $1E, $1E, $1E, $1E, $1E,
$1E, $1E, $1E, $39, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $39
                        .byte   $39, $1E, $37, $38, $1E, $37, $3A,
$3A, $38, $1E, $39, $1E, $37, $3A, $3A, $38, $1E, $37, $38, $1E, $39
                        .byte   $39, $1F, $35, $36, $1E, $35, $3A,
$3A, $36, $1E, $32, $1E, $35, $3A, $3A, $36, $1E, $35, $36, $1F, $39
                        .byte   $39, $1E, $1E, $1E, $1E, $1E, $1E,
$1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $39
                        .byte   $39, $1E, $37, $38, $1E, $31, $1E, $33,
$3A, $3A, $3D, $3A, $3A, $34, $1E, $31, $1E, $37, $38, $1E, $39
                        .byte   $39, $1E, $39, $39, $1E, $39, $1E,
$1E, $1E, $1E, $39, $1E, $1E, $1E, $1E, $39, $1E, $39, $39, $1E, $39
                        .byte   $39, $1E, $35, $36, $1E, $3B, $3A,
$3A, $34, $20, $32, $20, $33, $3A, $3A, $3C, $1E, $35, $36, $1E, $39
                        .byte   $39, $1E, $1E, $1E, $1E, $39, $20,
$20, $20, $20, $20, $20, $20, $20, $39, $1E, $1E, $1E, $1E, $39
                        .byte   $35, $3A, $3A, $38, $1E, $39, $20,
$37, $3A, $3A, $C5, $3A, $3A, $38, $20, $39, $1E, $37, $3A, $3A, $36
                        .byte   $3A, $3A, $3A, $36, $1E, $32, $20,
$39, $20, $20, $20, $20, $20, $39, $20, $32, $1E, $35, $3A, $3A, $3A
                        .byte   $20, $20, $20, $20, $1E, $20, $20, $35,
$3A, $3A, $3A, $3A, $3A, $36, $20, $20, $1E, $20, $20, $20, $20
                        .byte   $3A, $3A, $3A, $38, $1E, $31, $20,
$20, $20, $20, $20, $20, $20, $20, $31, $1E, $37, $3A, $3A, $3A
                        .byte   $37, $3A, $3A, $36, $1E, $32, $20,
$33, $3A, $3A, $3D, $3A, $3A, $34, $20, $32, $1E, $35, $3A, $3A, $38
                        .byte   $39, $1E, $1E, $1E, $1E, $1E, $1E,
$1E, $1E, $1E, $39, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $39
                        .byte   $39, $1E, $33, $38, $1E, $33, $3A,
$3A, $34, $1E, $32, $1E, $33, $3A, $3A, $34, $1E, $37, $34, $1E, $39
                        .byte   $39, $1F, $1E, $39, $1E, $1E, $1E,
$1E, $1E, $1E, $20, $1E, $1E, $1E, $1E, $1E, $1E, $39, $1E, $1F, $39
                        .byte   $3B, $34, $1E, $32, $1E, $31, $1E,
$33, $3A, $3A, $3D, $3A, $3A, $34, $1E, $31, $1E, $32, $1E, $33, $3C
                        .byte   $39, $1E, $1E, $1E, $1E, $39, $1E,
$1E, $1E, $1E, $39, $1E, $1E, $1E, $1E, $39, $1E, $1E, $1E, $1E, $39
                        .byte   $39, $1E, $33, $3A, $3A, $3E, $3A,
$3A, $34, $1E, $32, $1E, $33, $3A, $3A, $3E, $3A, $3A, $34, $1E, $39
                        .byte   $39, $1E, $1E, $1E, $1E, $1E, $1E,
$1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $1E, $39
                        .byte   $35, $3A, $3A, $3A, $3A, $3A, $3A,
$3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $3A, $36
                        .byte   $A0, $A0, $A0, $A0, $A0, $A0, $A0,
$A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0, $A0
;
; sprite colors (0-7)
QUIKMANCLR:
                        .byte   $07                     ; yellow
MONSTERCLR:
                        .byte   $02, $05, $03, $07      ; red, green,
cyan, yellow
                        .byte   $00, $00, $00                   ;
not used
;
;($1900) recolor maze paint every 8-jiffies ...
MAZEPAINT:
                        LDX #$00
@loop:          LDA ScreenData,X
                        CMP #$31
                        BCC @skip1
                        CMP #$3F
                        BCS @skip1
                        LDA MAZE
                        STA $9600,X
@skip1:         LDA ScreenData+$0100,X
                        CMP #$31
                        BCC @skip2
                        CMP #$3F
                        BCS @skip2
                        LDA MAZE
                        STA $9700,X
@skip2:         INX
                        BNE @loop
;
; powerpill flash every 30-jiffies
EFFECTS:
                        LDA FLASHPILL
                        CMP #$1E                ; 30-jiffies?
                        BNE @skip1
                        LDX #$00                ; reset counter
                        STX FLASHPILL
@loop1:         LDA $1CF8,X                     ; custom graphic char
                        EOR $8288,X             ; rom graphic char
                        STA $1CF8,X             ; redraw 8x8 char cell
                        INX
                        CPX #$08
                        BNE @loop1
                        LDA #$FE                ; render monster feet
                        EOR $1D7F               ; custom graphic char
                        STA $1D7F               ; redraw agressive monster
                        STA $1D87               ; redraw fleeing monster
@skip1:         INC FLASHPILL
                        INC BANNERTIMER
                        LDA VIC+$0E
                        BNE @skip3              ; playing?
                        LDX BANNERTIMER         ; no, could replace this
with JIFFYL
                        BNE @skip3
                        LDX #$00
                        INC BANNERFLAG
                        LDA BANNERFLAG
                        AND #$01
                        BEQ @skip2
                        LDX #$12
@skip2:         LDY #$00
@loop2:         LDA BANNERMSG,X
                        STA $1FE5,Y
                        INX
                        INY
                        CPY #$12
                        BNE @loop2
@skip3:         LDA VIC+$0E
                        BEQ @top                ; playing?
                        LDX #$00                ; yes
@loop3:         LDA SCORE,X                     ;  check  current  score
against high score
                        CMP ScreenData+$0F,X
                        BCC @top                ; is quikman beating the
high score?
                        BNE @skip4              ; yes!
                        INX
                        CPX #$06
                        BNE @loop3
@skip4:         LDX #$00
@loop4:         LDA SCORE,X                     ; woot!
                        STA ScreenData+$0F,X
                        INX
                        CPX #$06
                        BNE @loop4
@top:           LDX #$00                        ; refresh top line
@loop5:         LDA ScreenData,X
                        STA $1E00,X
                        INX
                        CPX #$15
                        BNE @loop5
                        JSR ANIMQMAN            ; update area pointed to by
SPRITEIMG1&2
                        JSR JOYSTICK
                        JSR ADDSCORE
                        STY POINTS
                        JMP SR105F
;
;($19AD)
DEATH:
                        SEI                     ; don't bother with any-
thing else
                        LDA #$01   ; only feature quikman dying
                        STA SPRITE
                        JSR SPRITES             ; redraw sprites
                        LDA #$50   ; low-order byte of 1st quikman image
                        STA SPRITEIMG1
                        LDA #$20
                        STA $03A5 ; rotate quikman 8 times
@loop1:         LDA #$1D  ; 2nd page where quikman is on
                        STA SPRITEIMG2
                        LDA SPRITEIMG1
                        CMP #$70   ; are we at the 4th quikman image?
                        BCC @skip
                        LDA #$50   ; reset to 1st quikman image
@skip:          CLC
                        ADC #$08   ; advance to next image
                        STA SPRITEIMG1
                        JSR SPRITES             ; redraw sprites
                        LDY #$D0
@loop2:         INX
                        BNE @loop2              ; bad wait loop
                        INY
                        BNE @loop2
                        DEC $03A5
                        BNE @loop1              ; repeat next sequence
                        CLI                     ; resume dazzling effects
                        JMP RESURRECT
```

```
;
;($19E8)
                .byte   $00, $00, $00, $00
;
;($19EC)  my very own sprite routines
; custom hack for this maze game implementation
SPRITES:
                LDA #$00        ; start with sprite #0
                STA $02CB
                STA $00         ; current sprite # to render
sprloop:
                ASL
                STA $01         ; current sprite (x2) pairing
index
                ASL
                ASL
                STA $02         ; current sprite (x8) image
index
                LDX $00
                LDA SPRITE
                AND SPRITEMASK,X
                BEQ @skip2      ; nothing to do?
                LDA SPRITELAST  ; what state was this sprite
before?
                AND SPRITEMASK,X
                BEQ @skip1      ; it was "off"
                JMP ANIMSPRITE  ; was "on" before, and we
still want it "on"
@skip1:  JMP ANIMSPRNOW  ; skip the erase, go turn it "on"
@skip2:  LDA SPRITELAST
                AND SPRITEMASK,X
                BEQ NEXTSPRITE  ; still nothing to do?  Then
do nothing ...
                JSR ERASESPRITE ; make this sprite disappear
NEXTSPRITE:
                INC $00
                LDA $00
                CMP #$05        ; only 5-sprites needed in
this game
                BNE sprloop
                LDX #$00
@loop2:  LDA SPRITE,X    ; save copy of current sprite registers
                STA SPRITELAST,X
                INX
                CPX #$11        ; all 17 values, not includ-
ing colors
                BNE @loop2
                RTS                                     ;
fini
;
;($1A33)  erasure part 1
LASTSPRITEPREP:
                LDA $01         ; 0, 2, 4, 6, 8
                CLC
                ADC #$CD        ; set register = "last"
                BNE SPRITEPREP2
;
; prepares the following registers:
; $FC/$FD  screen cell pointer for sprite's "home" position
; $FE/$FF  color cell pointer for same
;($1A3A)  erasure part 1
SPRITEPREP:
                LDA $01         ; 0, 2, 4, 6, 8 index
                CLC
                ADC #$A2        ; set register = "current"
SPRITEPREP2:
                TAX                                     ;
save this coordinate register index
                LDA $0200,X     ; get "X" coordinate
                CMP #$A0        ; is "X" at or beyond last
column?
                BCC @skip1
                SBC #$A0        ; subtract 160-pixels wide
@skip1:  LSR                    ; and divide by 8-pixel
width
                LSR
                LSR
                STA $FC         ; save column offset from
left
                STA $FE         ; save column offset from
left
                LDA SCRNPAGE    ; get high order byte of
screen memory page
                STA $FD
                LDA $F4         ; get high order byte of
screen color page
                AND #$FE        ; make it and "even" number
                STA $FF         ; save high order
                LDA $0201,X     ; get "Y" coordinate
                CMP #$B8        ; is "Y" at or beyond last
row?
                BCC @skip2
                SBC #$B8        ; subtract 184-pixels high
@skip2:  LSR                    ; and divide by 8-pixel
height
                LSR
                LSR
                TAY
                BEQ @fini       ; if on top row, no math
required
@loop1:  LDA $FC                ; get column offset
                CLC
                ADC #$15        ; add 21 for next row
                BCC @skip3      ; overflow to next page?
                INC $FD         ; yes, increment high order
bytes
                INC $FF
@skip3:  STA $FC                ; save column offset
                STA $FE
                DEY
                BNE @loop1      ; do for each "row"
@fini:   RTS
;
;prepares saveback buffers for restoring, should a larger-numbered sprite be
;overlapping any part of a smaller-numbered sprite
;($1A7D)  erasure part 2
PREPMATRIX:
                LDY #$00        ; top-left sprite cell
                STY $FB
                LDA $01         ; 0, 2, 4, 6, 8 index
                ASL                                     ;
x2
                TAX
                STA $03         ; save my custom char #
@loop1:  LDA #$01               ; "white" color
                STA CASSBUFF,X  ; 2x2 cell saveback buffer
                LDA ($FC),Y     ; retrieve screen cell
@retry:  CMP $03
                BCC @skip1      ; is A < $03 ?
                CMP #$1E
                BCS @skip1      ; is A >= $1E ?
;there is a sprite # greater than us on top ...
                STX $04         ; save X index
                PHA                                     ;
push A to stack
                TAX
                LDA CASSBUFF,X  ;
                LDX $04
                STA CASSBUFF,X
                PLA                                     ;
pop A from stack
                TAX
                LDA SAVEBACK,X
                LDX $04
                CLC
                BCC @retry
@skip1:  STA SAVEBACK,X
                INX
                TYA
                LDY $FB
                CLC
                ADC CHARMATRIX,Y
                TAY
                INC $FB
                LDA $FB
                CMP #$03        ; truncated from 4 to 3-
cells
                BNE @loop1
                RTS
;
;puts the sprite character matrix on the screen
;($1AC1)
PLACEMATRIX:
                LDY #$00
                STY $FB
                LDX $00
                LDA $01
                ASL
                PHA
@loop1:  PLA
                STA ($FC),Y
                CLC
                ADC #$01
                PHA
                LDA SPRITECLR,X
                STA ($FE),Y
```

```
                TYA
                LDY $FB
                CLC
                ADC CHARMATRIX,Y
                TAY
                INC $FB
                LDA $FB
                CMP #$03        ; truncated from 4 to 3-
cells
                BNE @loop1
                PLA
                RTS
;
;restores the sprite's saveback buffer to the screen squares it occupies
;($1AE9)  erasure part 3
RESTOREMATRIX:
                LDY #$00
                STY $FB
                LDA $01
                ASL                             ;
x2
                TAX
@loop1:  LDA SAVEBACK,X  ; recover character
                STA ($FC),Y     ; restore to screen
                LDA #$01
                NOP
                STA ($FE),Y     ; and leave "white" behind
                INX
                TYA
                LDY $FB
                CLC
                ADC CHARMATRIX,Y
                TAY
                INC $FB
                LDA $FB
                CMP #$03        ; truncated from 4 to 3-
cells
                BNE @loop1
                RTS
;
;render sprite within its character matrix by merging its image over its save-
back
;($1B0D)
RENDER:
                LDX $00
                LDA SPRITEIMG1,X
                STA $05
                LDA SPRITEIMG2,X
                STA $06
                LDX $01
                LDA SPRITEY,X
                AND #$07
                STA $03
                TAX                             ;
X will hold the sprite's Y coord
                LDY #$00        ; erase temp image matrix
area
                TYA
@loop1:  STA CASSBUFF+$20,Y
                INY
                CPY #$18        ; customized from 4 to 3
character cells
                BNE @loop1
                TAY                             ;
copy 8x8 character image into temp matrix
@loop2:  LDA ($05),Y             ; $05/$06 points to charac-
ter matrix
                STA CASSBUFF+$20,X
                INX
                INY
                CPY #$08
                BNE @loop2
                LDX $01
                LDA SPRITEX,X
                AND #$07        ; get modulos on X coordi-
nate
                TAY
                BEQ @skip1      ; if its zero, no shifting required
@loop3:  LDA #$00
                LDX $03
@loop4:  CLC
                ROR CASSBUFF+$20,X
                ROR CASSBUFF+$30,X
                INX
                CLC
                ADC #$01
                CMP #$08
                BNE @loop4
                DEY
                BNE @loop3
@skip1:  STY $FB                 ; Y is always zero here
@loop5:  LDA $01                 ; index x2
                ASL                             ;
and x2 = x4
                CLC
                ADC $FB
                TAX
                LDA #$1C        ; 1st page is where sprites
are stored
                STA $06
                LDA SAVEBACK,X
                CMP #$80        ; is character reversed?
                BCC @skip2
                LDY #$80        ; yes, use start of ROM
character set
                STY $06
@skip2:  AND #$1F                ;get modulos of first 32-characters
                ASL                             ;
and multiply by 8-pixel height
                ASL
                ASL
                STA $05         ; save as low-order byte
index
                LDA SAVEBACK,X
                AND #$60        ; mask 01100000
                LSR             ; divide by 16
                LSR
                LSR
                LSR
                LSR
                CLC
                ADC $06         ; add result to high-order
page index
                STA $06
                LDY #$00
                LDA $FB
                ASL
                ASL
                ASL
                TAX
@loop6:  LDA ($05),Y             ; copy 8x8 character image into behind matrix
                STA CASSBUFF+$40,X
                INX
                INY
                CPY #$08
                BNE @loop6
                INC $FB
                LDA $FB
                CMP #$03
                BNE @loop5
                LDY #$00
@loop7:  STY $FB
                LDA CASSBUFF+$40,Y
                EOR CASSBUFF+$20,Y
                AND CASSBUFF+$20,Y
                CMP CASSBUFF+$20,Y
                BEQ @skip3
                LDX $00
                LDA SPRITEMASK,X
                ORA $02CB
                STA $02CB
@skip3:  LDA $02
                ASL
                ASL
                CLC
                ADC $FB
                TAX
                LDY $FB
                LDA CASSBUFF+$20,Y
                ORA CASSBUFF+$40,Y
                STA $1C00,X
                INY
                CPY #$18        ; customized from 4 to 3
character cells
                BNE @loop7
                RTS
;
;($1BD9)
ANIMSPRITE:
                JSR ERASESPRITE
ANIMSPRNOW:
                JSR SPRITEPREP
                JSR PREPMATRIX
                JSR RENDER
                JSR PLACEMATRIX
                JMP NEXTSPRITE
```

```
;
;$(1BEB)
ERASESPRITE:
                    JSR LASTSPRITEPREP
                    JSR PREPMATRIX
                    JSR RESTOREMATRIX
                    RTS
;
;($1BF5 - $1BFF)
CHARMATRIX:
                    .byte       $15, $EC, $15
SPRITEMASK:
                    .byte       $01, $02, $04, $08, $10, $20, $40, $80
;
; Custom character data ($1C00 - $1DFF)
GraphicData:
                    .byte       $3E, $7C, $F8, $F0, $F0, $F8, $7C,
$3E
                    .byte       $00, $FF, $00, $00, $00, $00, $81, $42
                    .byte       $00, $00, $00, $18, $18, $00, $00, $00
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
                    .byte       $38, $7C, $FE, $92, $FE, $FE, $FE,
$AA
                    .byte       $00, $FF, $00, $00, $00, $00, $00, $00
                    .byte       $00, $00, $00, $00, $00, $00, $00, $00
                    .byte       $00, $FC, $02, $01, $01, $02, $FC, $00
                    .byte       $38, $7C, $FE, $92, $FE, $FE, $FE,
$AA
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
                    .byte       $00, $00, $00, $00, $00, $00, $00, $00
                    .byte       $18, $24, $42, $42, $42, $42, $42, $42
                    .byte       $38, $7C, $FE, $92, $FE, $FE, $FE,
$AA
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
                    .byte       $42, $42, $42, $42, $42, $42, $42, $42
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
                    .byte       $38, $7C, $FE, $92, $FE, $FE, $FE,
$AA
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
                    .byte       $00, $00, $00, $00, $00, $00, $00, $00
                    .byte       $42, $42, $42, $42, $42, $42, $24, $18
;
;($1CA0)
GAMEOVER:
                    LDA VIC+$0E         ; no sound?
                    BEQ @skip1
                    RTS                 ; yes, must be playing
game
@skip1:     LDY #$00                    ; no, paint for banners
                    LDA #$03            ; use cyan
@loop1:     STA $97E3,Y                 ; color RAM
                    INY
                    CPY #$15            ; do the whole row
                    BNE @loop1
                    LDA #$20            ; use a space
                    STA $1FE3
                    STA $1FF7
                    LDX FRUITLEVEL
                    CPX #$0C
                    BCC @skip2
                    LDX #$0C
@skip2:     LDA FRUIT,X
                    STA FRUITCELL       ; display final level achieved
                    LDA FRUITCLR,X
                    STA FRUITCELLCLR
                    RTS
;
;($1CCF)
SR1CCF:
                    LDA CHEWING         ; is quikman eating a dot?
                    CLI
                    BNE @skip1
                    JSR SPRITES
                    RTS
@skip1:     JSR SPRITES
                    PLA
                    PLA
                    JMP QUIKMAN
;
; resume graphic character data
;($1CE0)
                    .byte       $00, $00, $00, $00, $00, $00, $00, $00
            ;
                    .byte       $3C, $7E, $FF, $FF, $FF, $FF, $7E,
$3C          ; ] pacman animated
                    .byte       $00, $00, $00, $18, $18, $00, $00, $00
            ; ^ dot
                    .byte       $00, $3C, $7E, $7E, $7E, $7E, $3C,
$00          ; <- powerpill animated
                    .byte       $00, $00, $00, $00, $00, $00, $00, $00
            ;  empty space
                    .byte       $92, $54, $28, $C6, $28, $54, $92, $00
            ; ! explosion
                    .byte       $04, $08, $18, $24, $62, $F7, $F2, $60
            ; " cherry
                    .byte       $10, $7C, $FE, $AA, $D6, $AA, $54,
$28          ; # strawberry
                    .byte       $20, $10, $7C, $FE, $FE, $FE, $7C,
$38          ; $ peach
                    .byte       $08, $10, $7C, $FE, $FE, $FE, $7C,
$28          ; % apple
                    .byte       $08, $10, $38, $38, $7C, $FE, $FE,
$6C          ; & pear
                    .byte       $10, $30, $92, $FE, $7C, $38, $10, $28
            ; ' tbird
                    .byte       $10, $38, $7C, $7C, $7C, $7C, $FE,
$10          ; ( bell
                    .byte       $18, $24, $18, $08, $08, $18, $08, $18
            ; ) key
                    .byte       $3C, $7E, $FF, $FF, $FF, $FF, $7E,
$3C          ; * pacman closed
                    .byte       $3E, $7C, $F8, $F0, $F0, $F8, $7C,
$3E          ; + pacman right
                    .byte       $3C, $7E, $FF, $FF, $E7, $C3, $81,
$00 ; , pacman down
                    .byte       $7C, $3E, $1F, $0F, $0F, $1F, $3E,
$7C          ; - pacman left
                    .byte       $00, $81, $C3, $E7, $FF, $FF, $7E,
$3C          ; . pacman up
                    .byte       $38, $7C, $FE, $92, $FE, $FE, $FE,
$AA          ; / ghost chasing
                    .byte       $38, $7C, $FE, $92, $FE, $82, $FE,
$AA          ; 0 ghost fleeing
                    .byte       $18, $24, $42, $42, $42, $42, $42, $42
            ; 1 maze wall north
                    .byte       $42, $42, $42, $42, $42, $42, $24, $18
            ; 2 maze wall south
                    .byte       $00, $3F, $40, $80, $80, $40, $3F, $00
            ; 3 maze wall west
                    .byte       $00, $FC, $02, $01, $01, $02, $FC, $00
            ; 4 maze wall east
                    .byte       $42, $41, $40, $40, $40, $20, $1F, $00
            ; 5 maze wall s-w elbow
                    .byte       $42, $82, $02, $02, $02, $04, $F8, $00
            ; 6 maze wall s-e elbow
                    .byte       $00, $1F, $20, $40, $40, $40, $41, $42
            ; 7 maze wall n-w elbow
                    .byte       $00, $F8, $04, $02, $02, $02, $82, $42
            ; 8 maze wall n-e elbow
                    .byte       $42, $42, $42, $42, $42, $42, $42, $42
            ; 9 maze wall vertical
                    .byte       $00, $FF, $00, $00, $00, $00, $FF, $00
            ; : maze wall horizontal
                    .byte       $42, $41, $40, $40, $40, $40, $41, $42
            ; ; maze wall west tee
                    .byte       $42, $82, $02, $02, $02, $02, $82, $42
            ; < maze wall east tee
                    .byte       $00, $FF, $00, $00, $00, $00, $81, $42
            ; = maze wall north tee
                    .byte       $42, $81, $00, $00, $00, $00, $FF, $00
            ; > maze wall south tee
                    .byte       $3C, $42, $99, $A1, $A1, $99, $42,
$3C          ; ? copyright symbol
```