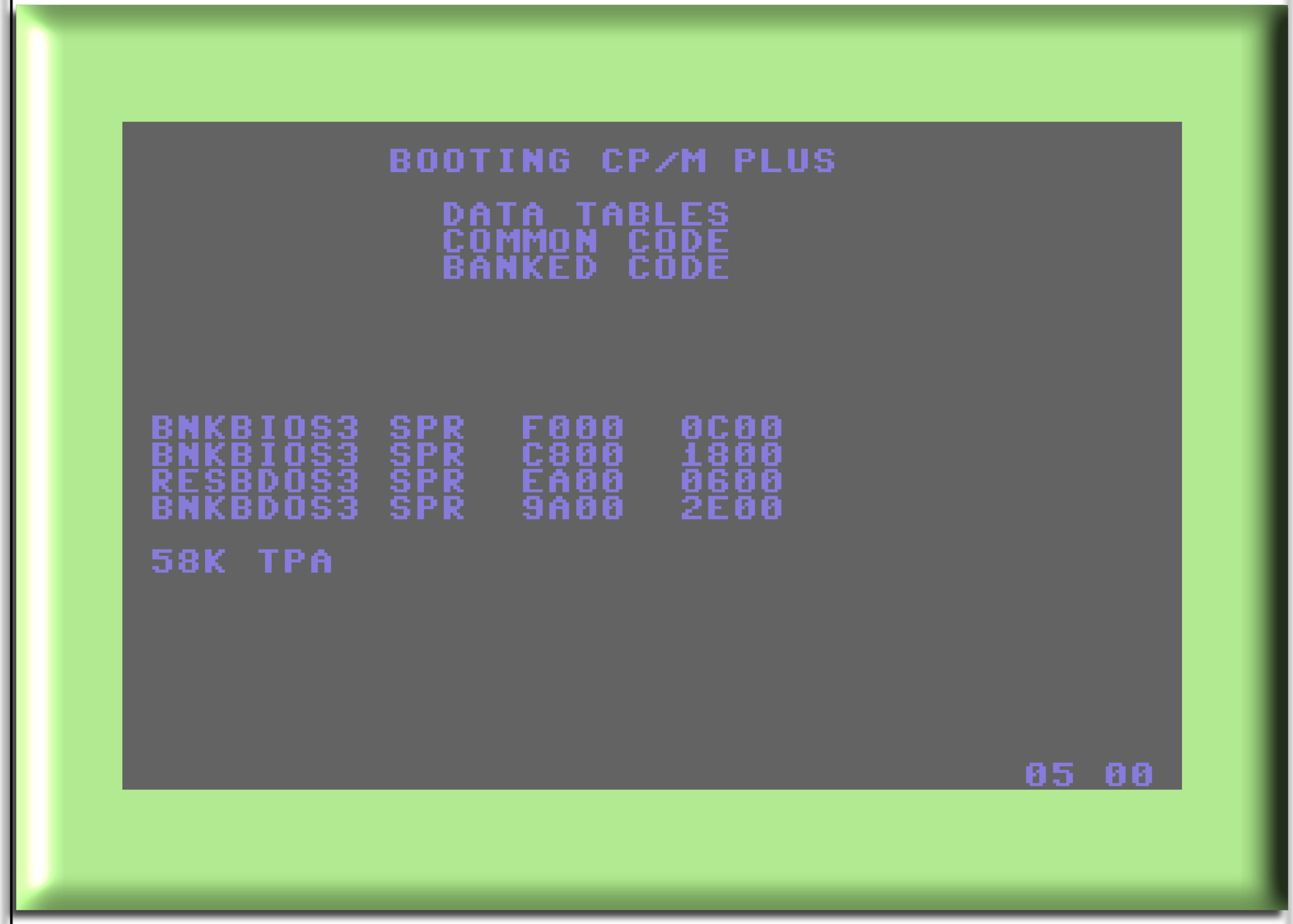
Commodore Free

Issue 30 May 2009

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



```
          BOOTING CP/M PLUS

            DATA TABLES
            COMMON CODE
            BANKED CODE



BNKBIOS3 SPR  F000  0C00
BNKBIOS3 SPR  C800  1800
RESBDOS3 SPR  EA00  0600
BNKBDOS3 SPR  9A00  2E00

58K TPA







                                    05 00
```

Editor

Hello again!
Well it's an issue fairly dedicated to the CP/m Operating system, many Commodore users will not have heard or even used CP/m and I always think that Gary Kildall never receives the credit he deserves for its creation. Gary has almost been removed from computing history especially if you look at some of the larger company's version of history. To
address the problem I have been given the ok to reprint an article originally created by Sol Libes. I also found some information about CP/M in a document from HEREN systems, I would like to than them for the permission to reprint the article.

Disappointment
The main problem I had was creating an issue where it seemed no one uses the CP/M Operating system on a Commodore machine, I know this is not true as many forum questions are posted about Commodore and CP/M so it seemed strange no one wanted to talk about Commodore and CP/M I did however manage to contact Gaby Chaudry Who agreed to supply some information but it was more general and not directly Commodore related. I know very little about CP/M using Commodore machines so here is my best effort, I asked for information and received little back.

I did manage an interview with the creator of " Dos65 commodore" and you can read his comments about the system later in the magazine. Also starts another Alternative Programming Languages: this time in C

Looking intelligent
Someone on the radio (an academic researching pausing and breaks in language and communication) said that "starting a sentence with the word SO makes you seem more important and also seems to be a trend at the moment especially with University students"

SO..
Moving quickly on, we have some interesting news from Retrobright who seem to have solved the problem of yellowing cases. As Chemistry is not a good subject for me and the chemicals involved are rather potent to say the least I think I wont bother doing my own trials on this particular project. However the results on the website do look rather good, and is something many users though impossible for a number of years.

Andrew Fisher sent in a review of the Retro Gamer CD CARTRIDGE RANGE although these have been looked at briefly in other issues, any new games are always a welcome to anyone's collection, especially when the are as well produced as these titles

TV
Last month I moaned about Red Dwarf, I will now leave this to die out. However I did manage to catch the remake of Reggie Perrin that was originally called "The Fall And Rise Of Reginald Perrin". Although some of the jokes
were amusing and the show did capture the spirit of the original, you can't help but wonder why companies remake older shows, the originals are always better version, however I suppose in the current climate it's cheaper for
companies to remake than inventing something new. I sat and watched the show rely because there was nothing else on the TV. With hundreds of cable channels to choose from nothing was worth the money I spend on the cable subscription, the usual story these days. Weren't we better in the UK at least when we had just 5 channels of "choice" or even 3 channels. I remember my grandparents thinking things were better with 2 channels. The more channels the less there is to watch! Oh no I am turning into Reggie Perrin. The BBC posting about the remake can be found here

http://www.bbc.co.uk/pressoffice/proginfo/tv/2009/wk16/unplaced.shtml#unplaced_reggieperrin

Still however critical you want to be about remakes, the show was amusing but I would have preferred the BBC to just re-run the original series.

CONTENTS

HOW TO HELP COMMODORE FREE

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

NEWS

commodore computer club

3rd CCC (UK) meet in Durham; 19th-21st June 09

The third Commodore Computer Club (UK) club meeting will be held in Durham from Friday 19th to Sunday 21st June 2009. Beer and socialising will happen on the Friday with the main event on the Saturday. Pete (Badders) will be updating us on his little project to restore a Commodore PET 2001, and I've got my SX-64 (serial number GA0000001) to working order which will also be there to look at and use. (The original chip set is preserved).Our last meet was vaguely themed around the Plus/4 and C16, with all other Commodore 8-bits getting a look in. If you`re interested in attending, please email me or visit our website - http://www.CommodoreComputerClub.co.uk. Non-members are welcome as long as they don't mind paying the L2.50 signing-in fee. Members get free entrance and other extras.

Binary Zone Spring Sale and new Psytronik releases

There is a special spring sale with reduced prices at Binary Zone Interactive. http://www.binaryzone.org/retrostore/
Loads of items in the Binary Zone store have been specially reduced for the sale including a selection of the
Psytronik Software Deluxe Disk releases. The cost of postage has also been reduced on ALL the Psytronik releases.
http://www.binaryzone.org/psytronik/

The latest releases by Psytronik, a sub label of Binary Zone for C64 games, are Special Editions of Creatures 1 and
2. Further more, a Special Edition of Armalyte is being prepared.

A new game is being worked on as well: Lots of screenies and a picture of the superb Knight and Grail box art can now be seen in the Psytronik blog. http://psytronik.blogspot.com/

HardSID 4U Winamp plugin

We're proud to release the HardSID 4U Winamp plugin
Check out the HD Video ScreenCast (Watch in HD and full-screen):
http://www.youtube.com/watch?v=FTnSEUxYkm0

* Seek in .sid files on cycle-accurate SID hardware!
* Remote-control your SID files using an IR remote controller! (free plugin req.)
* Enjoy the benefits of Winamp playlists! (+song-length support)
* Put sub-songs on your playlists! (not just that annoying default-song)
* Build interesting community SID listening stats published at

http://www.hardsid.com/stats.php

Download the plugin from:
http://www.hardsid.com/files/other/HardSID%204U%20Winamp%20Plugin.rar

Lotek64 #29

Lotek64 issue 29 has been shipped to its subscribers.

Content (extract):
* a small historiography in two parts
* musical video games
* mood of crisis in Soviet-Unterzögersdorf, sector II
* Bonjour tristesse!
* economic crisis also with C64
* The games by Ralf Glau
* eBay, Master System and the whole rest
* Master System / NES tips

The PDF version is available from the Lotek64 homepage, www.c64.at

Software Hut Selling GVP Products

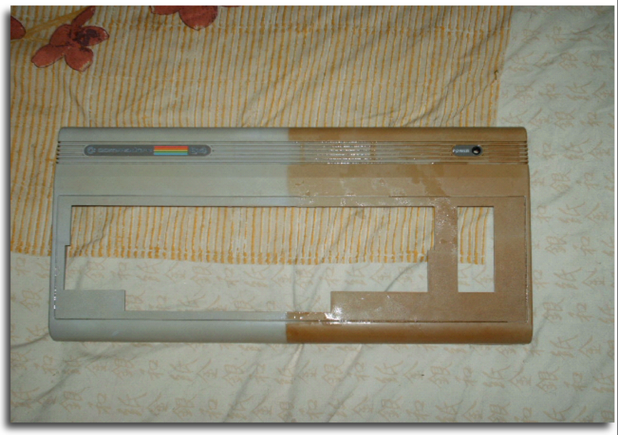
Software Hut and GVP have teamed to bring back many popular GVP products for the Amiga at new, lower prices.
These are newly made production pieces with a full 1 year warranty. Many of the items are limited production so hurry as they will sell out fast.
http://www.softhut.com/cgi-bin/test/Web_store/web_store.cgi?page=catalog/hardware/accelerators/gvp-m_index.html

GVP
GREAT VALLEY PRODUCTS INC.

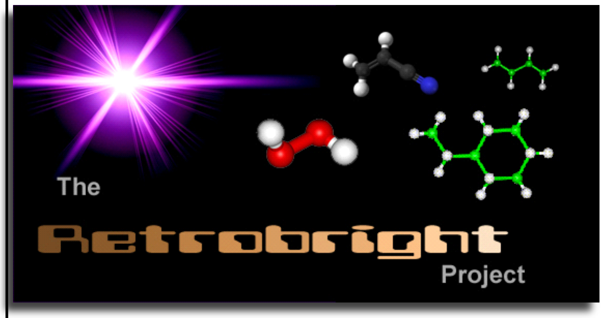
SoftwareHut

RETROBRIGHT
Restore old Yellowing computer cases

Many people have wanted a cure to the problem of aged plastic especially Commodore users as these machines were mainly cream in colour, with the exception of a few models. So How do you reverse the aging process and make the cream coloured case that has yellowed in the sun turn back to Cream. Many have stated this is impossible because it?s a chemical change, various chemical have been used with no success until now..



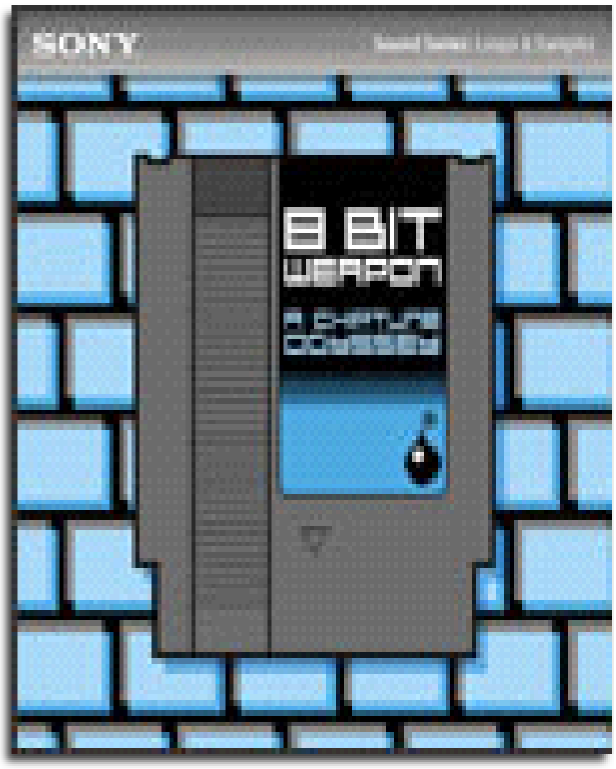
Taken from the website
" a chance discovery was made in March 2008, by The CBM Museum at Wuppertal in Germany (http://www.forum64.de), that immersing parts in a solution of Hydrogen Peroxide for a few days could partially reverse the process. This was initially taken up by the Amiga community in Germany (http://www.a1k.org) and the idea eventually found its way to the English Amiga Board (http://eab.abime.net), where a madcap collection of chemists, plastics engineers and retro hackers managed to perfect this concept and put it on steroids, with help from other forums."



The results are truly amazing all yellowing has been removed and the cases that have been half dipped in the solution look unbelievable (half still yellow but the dipped sides are bright and look better than new) also keys on machines have been brought back to life removing the yellowing process with the solution

http://retr0bright.wikispaces.com/

8 Bit Weapon loop and sample library



Sony Creative Software has released our new loop and sample library called, "8 Bit Weapon: A Chiptune Odyssey". The library covers Apple II, Commodore 64, NES, Gameboy, and the Atari 2600. Each system library has everything from drums, bass and synth to special effects. The C64 section of the library was made with Prophet 64/Sidstation/ Sight & Sound Music and we used both the 6581r4 & the 8580 SID chips. Both Computeher and I have featured
song demos built into the collection that can also be remixed!

The sound library works with ACID, Ableton Live, Cubase, Garage Band, Logic, Soundtrack, and more!

Users who purchase the "Green" Downloadable version of the Sony loop library will get the full MP3 release of our new "Electric High EP" MP3 album plus an exclusive track "Break Beat Bonanza" as an added bonus!

Here is Sony's Page on it:
http://www.sonycreativesoftware.com/8bitweapon

If you just want our Limited Edition "Electric High EP" CD go here
http://www.8bitweapon.com/store.htm

- Seth & Michelle :) :]
www.8bitweapon.com
www.computeher.net

Retro Gamer CD RGCD 5 Released

I forgot to include this in the other editions, of Commodore Free Magazine (jeez there is just to much stuff going on for me to follow)
Ok then Retro Gamer Issue 5 was released in March listed below are a list of the items included

Contents
Featured Game
ThrustBurst (PC)

Retro Reviews
3D Starstrike (PC)
8Bit Killer (PC)
Animal Party (Atari XE/XL)
Bob Fossil's Sudok universe (Atari ST(E)/Falcon)
Crocodingus In Cube Island (NDS)
Destructivator (PC)
Diagonal Ball (C64)
Droid Assault (PC)
Farmer Jack 2 & 3 (Spectrum 48/128K)
Follia NBA (Amiga)
Harpooned (PC)
Jihad (C64)
Lead (Atari 2600)
Loops Of Zen (Atari XE/XL)
Manical Drop (Atari ST(E)/Falcon)
Noitu Love 2: Devolution (PC)
Plutos (Atari 7800)
Qwak (PC)
RallyBug (Spectrum 48/128K)
Rana Remake (PC)
Rock Boshers (PC)
Rom Check Fail! (PC)
Ropor (PC)
Self Destruct (PC)
Sirius (Atari 7800)
Skull Pogo (PC)
Soup Du Jour (PC)
Splattr (Spectrum 128K)
Staroid (PC)
Sub Hunter (C64)
Swapz (Atari XE/XL)
The Hordes (PC)
The Pairs Are Gone / Spy4K (Atari ST(E)/Falcon)
War Twat (PC) War
Twat 2: SY!NSO! (PC)
Wizard Of Wor (Spectrum 48/128K)
World Reborn (GBA)
Zompocalypse (PC)

Extras
D-Bug Falcon/MSTE Fixes/ULS Update (Atari Falcon/Mega STE)
Enforcer 2 Preview (C64) Homebrew Retrospective
(Atari ST(E)/Falcon) Kikstart C16 & Invasive Action Cartridges
(C64) Night of the Cephalopods Preview (PC) The
Alien Team (C64)

http://www.rgcd.co.uk/current-issue/
Emulators
iDeaS 1.0.2.9 (PC) Klive 1.1 (PC) MESS 0.129 STEem 3.2 (PC)
Stella 2.7.3a (PC) Visual Boy Advance 1.7.2 (PC)
WinUAE 1.5.3 (PC) WinVICE 2.1 (PC)

High Voltage SID Collection Update 50

http://hvsc.c64.org/

Date: May 01, 2009

Resulting Version: 50
Previous Version: 49

Hello fellow lover of SID music!
Nice you found some time to read through this script, to see what has been changed in the HVSC and for what reason.

After this update, the collection should contain 36,000 SID files!

This update features (all approximates):
678 new SIDs
59 fixed/better rips
573 PlaySID/Sidplay1 specific SIDs eliminated
189 repeats/bad rips eliminated
482 SID credit fixes
135 SID model/clock infos
15 tunes from /DEMOS/UNKNOWN/ identified
10 tunes from /GAMES/ identified
36 tunes moved out of /DEMOS/ to their composers? directories
11 tunes moved out of /GAMES/ to their composers? directories

As you can see, a great amount of files were eliminated in this update. We decided to remove all the old _PSID files that have a RSID equivalent. Those _PSID files were only hacks meant for digitized sounds to be played inSidplay1, but not on real C64. The _PSID files will be eliminated in every future releases as soon as new RSID versions of the same .sid files are added. For those still needing the _PSID files we have put them in a separate archive:

http://hvsc.c64.org/Downloads/C64MUSIC_PlaySID.rar

We are always looking for others to help make HVSC a better collection. If you find any errors or have new rips to add, please email:

Jan Harries, iAN CooG : hvsc(sid)c64.org

Thanks to all the people who have helped to make HVSC the collection that it is today, without your help it would be a much tougher task.

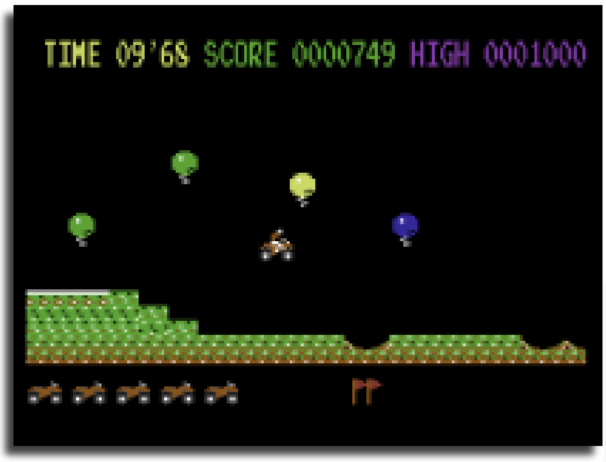
iAN CooG/HF^HVSC

RGCD CARTRIDGE RANGE
Review by Andrew Fisher

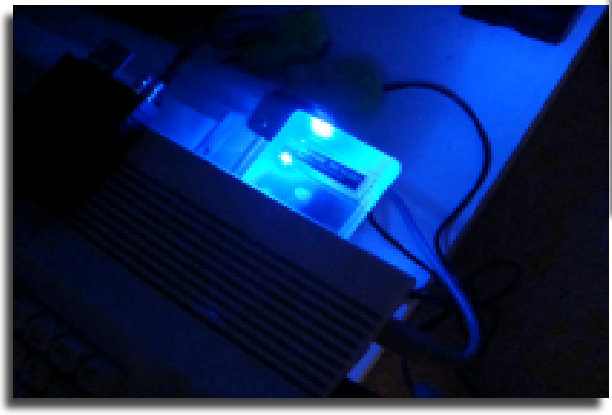
James Monkman (Heavy Stylus) is a fan of retro gaming, and decided to set up his own retro gaming fanzine, dedicated to homebrew games, remakes and new titles for old machines. He called it RGCD and has so far published five issues, which can be downloaded from the website or purchased on CD. Each issue has reviews and features on new homebrew, and includes download links or the actual games and emulators themselves. Among the C64 games getting coverage in the magazine have been Sub Hunter, Greenrunner and Joe Gunn. Issue 5 even has the RGCD C64 game charts.

But as well as the CD magazine, James has been selling games through the website. It started with a small run of GBA games, and has now expanded to include a range of cartridge games for the Commodore 64. There have been four released so far, with more planned for the near future. Here are some short reviews of each game, and an added bonus is that all four titles are compatible with the C64GS (Games System).

KIK START C16 by Cosine
The first release came in two forms – the limited edition, in a large red plastic case, and the more standard cardboard box as used on the other games. I was one of the first to order, so I got hold of the limited edition. In case you were unaware, this is actually a radically different game to the split-screen racing action of the C64 Kik Start games. Instead, it is based on an old arcade game where the player has to jump obstacles and collect balloons. It`s a great game on the C16, and the conversion does it justice with added music and retaining the original`s playability.



INVASIVE ACTION by Backward Engineering



This is also by TMR, a Centipede clone that soon gets tough. It is actually based on a game called 'Invasive Action' from Cascade's Cassette 50. But probably the most interesting thing about this one is that the cartridge has an LED in it – and so when it is plugged in and switched on, it glows!

BLOCK FRENZY by Backward Engineering
Yet another game from TMR, this one based on a Flash game that was doing the rounds a couple of years ago. The player controls a red square trapped between electric walls. Moving around the screen are blue blocks that must be avoided. And that`s it – the longer you play, the more you score.

GEIR STRAUME'S MINIGAME COMPETITION COLLECTION
Packed onto the 16K cartridge are four different mini-games, created by Geir for competitions between 2002 and 2005. Each game comes with a little laminated instruction card, which is a nice touch. Sub Destroyer has your ship at the top of the screen, dropping mines to destroy the passing submarines. With only a limited number of mines per level/ship, accuracy is important. Diamond Maze sees your happy face avoiding the enemies and collecting the three diamonds to open the exit to the next level. (Amusingly, I played this once and opened the exit – and one of the enemies ran out!) Bellringer has four horizontally scrolling levels filled with enemy knights that drain your energy. At the end of the level you have to ring the right combination of bells to open the exit. Bellringer II turns the action vertically, adding bags of gold to collect. Once again there are knights to avoid, and the bell-ringing puzzle at the end. All four games on this cartridge are well put together but offer limited amounts of fun. It?s one you can come back to every now and then.

CONCLUSION
So, it is great to see new cartridge games being released, and I am looking forward to seeing more in the future. Keep an eye on www.rgcd.co.uk for more news.

The Gary Kildall Legacy
by Sol Libes

Gary Kildall died in July 1994 at the age of 52. The computer media, with a few small exceptions, ignored his passing. The Circumstances of his death are pretty murky. One report attributed it to a fall from a ladder, another an incident at a bar, and another to a heart attack.

Every PC owner owes Gary a debt of gratitude. Bill Gates and Microsoft owe him more than anyone else. Gary was the first person to interface a disk system to a microcomputer and create an operating system for it. He changed what had previously been a circuit designed for process control applications into a fully functional computer. Microcomputers now did tasks that previously could only be done on minicomputers and mainframes. The world changed dramatically because of his work.

Gary received a Ph.D. in Computer Science from the University of Washington in 1972 and joined the Navy. It is interesting to note that both Gary and Bill Gates were born and raised in the Seattle area. Like Gates, Gary also had a passion for computers. However; unlike Gates he completed his college education. Their paths crossed early on when Gates, a high school student, and Gary, a college student, both worked on the same DEC PDP-10 computer system.

The Navy appointed Gary to be a Computer Science instructor at their Post-Graduate School in Monterey California. At the school; Gary purchased an Intel 4004 microprocessor chip set for himself and his students to experiment with. The 4004 was Intel's first microprocessor and the first in the world. It was programmable, handled 4-bit words and contained all of 2,250 transistors. Intel, at the time was primarily in the memory IC business, and created the 4004 as a custom project for one customer. When the customer wanted to renegotiate the price Intel asked that they give up their exclusive rights to the device. Intel introduced the chip in November 1971. Much to Intel's surprise the device was an instant success. Engineers began designing it into a wide variety of applications such as scales, traffic light controls, musical instruments, washing machines, printers, and more; Intel soon realized that 4004 system designers needed software development support. Gary was hired as a consultant to create a programming language for the device. Gary created PL/M (Programming Language /Microprocessor) to run on an IBM 360 computer and generate executable binary code that was then burned into the ROM memory of the 4004 system.

Marcian "Ted" Hoff designer of the 4004 (Who also worked later in his career for Atari Corp), quickly followed with the 8008, the first 8-bit microprocessor. It was introduced in March 1972. Gary was again hired to develop PL/M for the device. Intel also designed an 8008-based small computer called the Intellec-8, it was about the same size as the IBM PC. The device was used for hardware and software development. Gary set one up in a classroom at school for his students. To make it easier to use Gary developed a simulator/debugger program for the unit. Intel began to see that microprocessors helped sell more memory chips and developed a much more powerful 8-bit microprocessor the 8080. Gary was again hired to create the development software. He was given an Intellec-80 to use at school.

In 1973 Shugart gave Intel a sample 8" floppy disk. Gary was immediately intrigued by the device and with a friend, John Torode, built a controller interface to an Intellec-80. Gary, and his students, wrote a small control program, which he called CP/M (Control Program/Microcomputer). It enabled him to read and write files to and from the disk. Gary copied the commands and file-naming conventions from the DEC PDP-10 VMS operating system. Gordon Eubanks, one of Gary's students, created a BASIC interpreter for the system. Early versions of CP/M and the BASIC interpreter were in the public domain since it had been created at a publicly funded institution. Copies found their way to some other government contractors and agencies.

In 1976, after his discharge from the Navy, Gary became a full- time consultant, using the name Intergalactic Digital Research. Together with Torode he designed floppy disk systems for several microcomputer manufacturers. At the time, MITS and IMSAI, the two leading 8080 microcomputer system kit makers, announced floppy disk systems. MITS offered a version of BASIC (written by Bill Gates and Paul Allen) that could load and save BASIC programs on disk. MITS contracted with another software developer for a Disk Operating System. When shipped in early 1977, it proved unreliable and had poor performance. MITS also refused to license the DOS to other system makers.

IMSAI, needing to compete with MITS, approached Gary for a non-exclusive CP/M license for a fixed $25,000 fee. Since several other manufacturers also wanted CP/M, Gary rewrote it completely to make it easier to install on different hardware systems. He made it hardware-independent by creating a separate module, which he called the BIOS (Basic Input/Output System). He also added an editor, assembler, debugger, and several utilities. CP/M became a full-blown computer development system. Gary and his former wife Dorothy McEwen formed Galactic Digital Research Inc. (Later it was changed to just Digital Research, Inc) to market CP/M-80. They placed a small classified ad in Dr. Dobb's Journal and were surprised by the large number of orders from hobbyists for the $90 software package. By early 1977, several manufacturers were including CP/M with their systems. They provided a ROM-BIOS so that CP/M loaded immediately on power-up.

By 1978 Microsoft BASIC and FORTRAN, UCSD Pascal, MicroPro's WordStar, Ashton-Tate's dBase, and other programs were running on CP/M-based on machines from Apple, Radio Shack, Commodore, Zenith, Sharp, and almost a hundred other manufacturers. In 1980, IBM approached DRI, to license CP/M-86, an 8086 version of CP/M then being developed. Gary had been working on this but delayed finishing it while working on several language projects. Intel had introduced the 8086 16-bit microprocessor in June 1978 and followed it a year later with the 8088, a lower-cost and slower version. IBM decided to use the 8088 for its new PC.

Seattle Computer Products in early 1979 introduced the first 8086 computer kit. Sales languished while SCP waited for DRI to introduce CP/M-86. In desperation SCP hired Tim Paterson to develop a DOS for them. Tim quickly created a simplified 8086 version of CP/M, which he called QDOS (Quick and Dirty Operating System, since it did not implement all of CP/M's features). Microsoft who were located nearby, modified BASIC for the system. IBM met with a cool reception when they approached DRI for a CP/M license. Dorothy McEwen and DRI's attorney refused to sign the IBM non-disclosure agreement (Gary did not attend the meeting), refused to make any modifications to CP/M-86 and insisted on a higher royalty than what IBM proposed. Bill Gates, who had been negotiating a BASIC license with IBM, seized the opportunity and offered to provide a DOS/BASIC package to IBM on favourable terms. Gates licensed SCP-DOS (for $50,000) and hired Tim Paterson to modify it to run on the IBM-PC. Microsoft submitted a copy to IBM for testing, who found over 300 bugs. IBM cleaned up many of the bugs, made a number of improvements and wrote the user manual. DRI released CP/M-86 soon after IBM released DOS Version 1.0. DOS had fewer features and poorer performance. IBM offered both CP/M-86 and DOS. CP/M-86 was offered at $240 versus $60 for DOS. Few PC owners were willing to pay the extra money DRI considered suing Microsoft for copying all the CP/M system calls, program structure, and user interface. However, DRI knew it would also have to sue IBM. It knew it did not have the resources for this and that its chances of success were remote. In 1982, IBM asked Microsoft to develop a hard disk version of DOS. Microsoft used the opportunity to completely rewrite DOS so that version 2.0 was very different from version 1.0 and DRI's opportunity to sue was gone. DRI continued to improve CP/M-86 introducing multi-tasking and multi-user versions. However, they were not completely compatible with DOS and largely ignored by the marketplace.

In1989 DRI introduced a DOS compatible version (DR-DOS) this was recognized as superior to DOS. However, Microsoft marketing tactics (disclosed in the Justice Department investigation) shut DRI out of the market. Microsoft responded with versions 5.0 and 6.0 incorporating many of DR-DOS's features.

Kildall also pioneered in the development of a GUI (Graphical User Interface) for the PC. Called GEM (Graphical Environment Manager), it was demonstrated at the November 1983 COMDEX and shipped in the spring of 1984. Atari also licensed it for use with their new 520ST computers. GEM presented the user with a screen virtually identical to that of the Macintosh, Apple threatened to sue DRI. DRI responded by making some cosmetic changes to GEM. DRI did not recognize the potential of a GUI interface and did not put any marketing effort behind it. DRI eventually withdrew GEM from the retail market. It continued to market GEM to software developers as a front end for their graphics products. The most well-known product to use the GEM GUI was "Ventura Publisher" from XEROX.

Microsoft finally demonstrated their Windows GUI at the spring 1985 Comdex, shipping version 1.0 in the fall. Microsoft learned from DRI's experience with Apple and made Windows appear slightly different from the Mac GUI. Version 1.0 proved an embarrassment to Microsoft; it was incredibly slow, unreliable, and lacked the smooth performance of GEM and the Mac, version 2.0 of Windows did likewise. Windows was completely rewritten for version 3.0 and released in the spring of 1990, with the most expensive software promotional campaign the industry had ever seen coupled with aggressive marketing (initial price was $39 and thousands of copies were given away free). Gates something that neither IBM, DRI, Apple, Xerox, or the other GUI developers were willing to do, namely, to make a total commitment, risking the entire company on the success of a GUI.

Microsoft sought to gain the largest market share by distributing Windows primarily through OEM channels. System manufacturers were persuaded to offer DOS and Windows preloaded onto hard disks by offering a low OEM price of $35 on average while offering Windows to retailers at $75. Microsoft actually made more money on the OEM version because the manufacturer assumed the cost of printing manuals, providing disk backups, the packaging, and support. Version 3.0 also proved unreliable, Microsoft fixed the bugs, added a few minor features and introduced it as version 3.1. Gates turned a major problem into a marketing success. 3.0 owners paying a second time, in effect paid for the repair of design defects.

Gary was also the first person to work on the development of software for driving CD-ROM interfaces. We will probably never know all of the system software work that he has created.

There is no doubt that Gary Kildall led the way in microcomputer software development. I wonder what Microsoft will do now that they no longer have Gary Kildall to lead the way for them?

Many ACGNJers met and spent time with Gary at the 1979 Trenton Computer Festival. I met with him many times, as a magazine author and editor, and President of ACGNJ. I developed great admiration for his talents, his hard work, and willingness to help others. I will also miss him as a friend.

From: Sol Libes
Subject: [Fwd: Fw: Commodore Free Magazine (The Gary Kildall Legacy)]
To: Commodore FREE

Hi Commodore Free,
Gary Kildall was a very remarkable person. He and I were friends and I was very upset when he passed away.
My article on Gary is not copyrighted and has appeared in numerous places on the web. I do not know who put the copyright notice on the article. It should not be there. To know more about Gary go to the Wikipedia web page http://en.wikipedia.org/wiki/Gary_Kildall>about him.

Sol

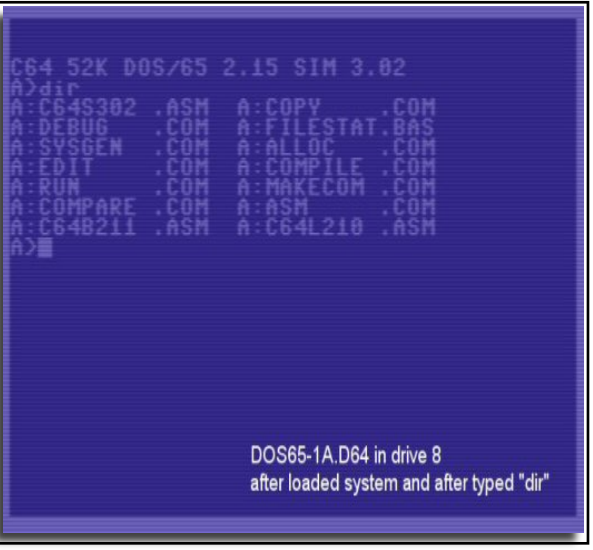
Dos65 commodore
richardaleary AT gmail.com
http://www.z80.eu/dos65.html

DOS/65 - an interesting operating system with file system compatibility to CP/M-80 (and other similarities)From the system description of DOS/65:"What I (Richard A. Leary) have done is attack the software side of the problem in order to make any 6502 system a truly workable disk based system. In addition a degree of compatibility is now possible not only between 6502 systems but with large parts of the world of CP/M systems. The result of my efforts is a system of software which I have named DOS/65."

Richard's implementation is initially made for S-100 based systems, but he began to port it to the C64 also (see below).

DOS/65 has a lot of CP/M look-alike commands for the command line and also some applications like BASIC-E included.
Of course some differences between CP/M-80 and DOS/65 exist - the lowest memory areas (e.g. the "zero page") are not usable for the TPA, so the usable system memory area starts at a higher address (e.g. $0400).

http://www.z80.eu/dos65.html



```
C64 52K DOS/65 2.15 SIM 3.02
A>dir
A:C64S302 .ASM  A:COPY     .COM
A:DEBUG   .COM  A:FILESTAT.BAS
A:SYSGEN  .COM  A:ALLOC    .COM
A:EDIT    .COM  A:COMPILE  .COM
A:RUN     .COM  A:MAKECOM  .COM
A:COMPARE .COM  A:ASM      .COM
A:C64B211 .ASM  A:C64L210  .ASM
A>█
```

DOS65-1A.D64 in drive 8
after loaded system and after typed "dir"

Interview with Dos65 Creator Richard Leary

COMMODORE FREE:
Please introduce yourself to our Readers

RICHARD LEARY:
My name is Richard (Rich) Leary and I have been involved in 6502-based systems since 1974. My first 6502 based computer was totally built by hand in 1974 using a plug board. The computer had two 1702A EPROMs, four 3539 256x8 static RAMs, and serial console I/O to a Teletype Model 19 Baudot teletype. If I remember correctly the machine used a 6520 for the serial I/O. The machine evolved through several stages including adding a cassette tape and other changes.

As I watched the 8080/Z80 computers evolve it was clear to me that I needed some form of disk I/O for my system and that meant a disk operating system (DOS) was also needed. By the late 1970s it looked like the 8080/Z80 DOS that was emerging as the cross-platform standard was CP/M operating system. In addition the Altair computer started the 8080/Z80 down a hardware path that used the S-100 bus. While there were other buses in use, the S-100 was embraced by a large number of manufactures so hardware selection had some options.

My effort then was devoted to getting a Z80, S-100, & CP/M based system working. I built two very large boxes with power supplies - one for the S-100 motherboard and all the plug-in cards and the other to house two eight-inch floppy disk drives. Once I got that working I built a 6502 CPU board for the S-100 motherboard and started to develop my 6502 DOS by disassembling CP/M and then creating 6502 equivalent software.

By 1981-1982 that software, that I named DOS/65, was operating well enough that I started to consider offering it as a commercial product to other 6502 enthusiasts. That led to a number of commercial sales culminating in a sizeable license to Rockwell in 1983-1984 that allowed Rockwell to use DOS/65 for their commercial products.

CF:. Would you say you were a commodore fan or is the Commodore link purely down to CP/M software

I started my Commodore experience many, many years ago with a VIC-20 when it first came out. The VIC was later replaced by a C64. I used those for some limited game playing as well as for the traditional word processing, spreadsheet, and similar applications.

I still have that first C64 and when I did the C64 port of DOS/65 it was that system with its two 1541s and JiffyDOS that was used for testing.

Later in the 1980s I leaped on the Amiga platform with first an A1000 and when the expansion limitations of that platform became an issue I traded it in for an A2000. That A2000 is also still operational and now has a 68030 processor, hard disk, CD-ROM, high resolution display, and other enhancements and runs Amiga OS 3.5

CF:. What is CP/M

CP/M (some say this is an acronym for Control Program/Microcomputer but there are other opinions) is a disk operating system originally developed by Gary Kildall (Digital Research). This site http://www.cadigital.com/kildall.htm has a brief history of what Gary did and what CP/M and especially CP/M-80 (the version Gary developed for the 8080 processor) was and what it's later relationship was to Bill Gates and Microsoft.

CP/M is both a development environment and an application environment. It provided the floppy or hard disk file system that made applications flexible and powerful for their day. More importantly by Version 2.2 it had matured into a system that achieved platform independence by concentrating their platform specific interface in an user alterable module called the Custom Basic Input Out System (CBIOS). This meant that different hardware could useCP/M as long as it had an 8080-compatible processor. Eventually most such systems used the Z80 but there were 8085-based systems as well as 8080-based systems.

CP/M then evolved to other platforms including the 16-bit 8088/8086 and to completely foreign processors, most notably the 68000. It eventually evolved into a multi-processing systems (MP/M) that was used in business environments and other applications.

CF:. Would you say DOS on the PC was a copy of the CP/M operating system then?

The reference link above discusses the relationship between CP/M and MS-DOS. I will say no more other than to note that this is a subject that stirs people's hearts to this day.

CF:. Were there different versions of CP/M and were they all compatible with each other?

There were differences. For example, CP/M-80 and CP/M-86 were targeted for different processors and hence a CP/M-86 application would not run on a CP/M-80 system or vice-versa. There was a degree of file system compatibility that would allow file exchange if the disks were the same.

Even within CP/M-80 the various disk formats meant that any specific system could only exchange files through media exchange if the hardware was compatible. In the early days when the "standard" was the "IBM" eight inch, single sided, single density diskette, media exchange was usually easy. As manufacturers moved to 5.25 inch drives
such compatibility became spotty. To this day there are some that are often exchangeable. For example the C128 version of CP/M will support both Kaypro 2 and 4 diskette formats using the 1571 drive.

CF:. I know DOS/65 was ported to the Commodore 64 was this the main idea to have the system running on the Commodore 64 or was it more a programming exercise did you do the coding ?

I had always had the Commodore platform as an intended target but just never had time for it. In late 2007 I brought my S-100 system back to life after being dormant for at least five years. However in early 2008 my S-100 system failed and rather than trying to immediately fix it I decided this was the incentive to build a new "primary" development environment using hardware and software I had and that was not "one-of-a-kind" nor as bulky and heavy as the S-100 system. After thinking about it I decided to use the C64.

I had to acquire a few tools (e. g., Star Commander, X1541 interface) to allow my PCs to interact easily with the C64 & 1541. It took only a short time to get familiar with those new tools as well as use of TASM as a cross development assembler. I also used the VICE X64 simulator for easier testing in addition to the actual hardware.

It took a little while to understand the C64 1541 CP/M disk format and how to use the C64 kernel (called kernal by Commodore) calls to accomplish both serial bus as well as console I/O. As noted separately, by early April I hada stable system and provided "beta" releases through Peter Dassow's site for people to try.

CF:. I believe you still own the rights to DOS/65, is there a charge for users wishing to use the software?

There is no charge as long as it is for personal or educational uses. If someone wants to use it commercially they need to talk to me. I doubt anyone will want commercial use but if they do it is not free.

CF:. I actually found the software on this website "http://www.z80.eu/dos65.html" were you approached for distribution rights or to produce a conversion to Commodore systems?

Peter Dassow's site is a CP/M oriented site. Peter expressed some interest in DOS/65 because it uses the CP/M file system and looks and feels a lot like CP/M. After some email exchanges I agreed to accept his very generous offer to host a DOS/65 page. This also motivated me to clean up the documentation, some of which had not been touched in ten years or more. I did that and produced .pdf versions of the documentation in the first half of 2008 and provided them to Peter. As the C64 version evolved it was also posted on Peter's site.

CF:. When was the Port of the software created for the Commodore 64?

I started working on the C64 implementation in January 2008 and by mid-April 2008 it was stable and complete. The C64 implementation focused on the disk I/O using the serial bus and 1541 drive as well as on the console I/O using the keyboard and normal C64 text video. DOS/65 was used as it existed although one small change was made to the main DOS/65 software to facilitate more flexible builds of new configurations, including the C64 version.

One of the features of DOS/65 that was inherited from CP/M is the use of a system interface module to handle all system-specific I/O needs. That means that the core portions of DOS/65 including the DOS itself as well as all applications need not know anything about the actual I/O.

CF:. Why was it called DOS/65?

That was my attempt to have a name similar to CP/M but with a name that tied itself to the 65xx processor family.

CF:. To your knowledge what other Computer systems has the software been ported to?

DOS/65 has been implemented on SYM-1, KIM-1, AIM-65, and some S-100 systems. It was also implemented on a couple of different STD-bus systems in the 1980s.

CF:. Can you explain why the software was created or ported to the Commodore 64, for example why was such software needed for Commodore systems in particular

That is an excellent question. Over the two decades plus since DOS/65 was developed and stabilized there were instances in which various people asked if I had a C64 version. While that had always been my intent I just never had the time. During those 20+ years I was working for a major aerospace company and just had little free time.

When I retired two years ago from that career I started to have more time available. Once we had our new housein the Colorado Rockies finished and we then had moved in during early November 2007, the possibility of applying some time became real. That was helped by the severe winter we had in December 2007 through May 2008 that meant "inside time" was plentiful.

While I thought there would be some people interested in trying DOS/65 on the C64, in the end I did it for the fun and to prove to myself that what I had intended 20+ years ago was possible and practical. As to value for the Commodore user - that is up to the user. Some may see it as pointless but in my view it has the same value as all other vintage computer hardware and software - it is fun. DOS/65 provides a development environment that is self contained. All the tools are there to build software in BASIC-E or assembly language. That gives the Commodore user the opportunity to do things that may not be easy to do in other ways but most importantly allows the user to develop skills and knowledge that are self-satisfying.

I am not working any other Commodore implementations now even though I have a C128D that would be a very nice DOS/65 platform. I also have an Apple IIGS that would be even better given it's somewhat faster and capable processor. I am not doing anything with the IIGS at this time.

What I am working on is the hardware to add a floppy disk interface and possibly an IDE interface to Daryl Rictor's SBC-2 v2.3 6502 based single board computer. I built one of Daryl's SBCs this year and it is working quite nicely. The floppy I/O board is in the design stage and probably won't be completed until this winter when the snows again make "inside time" dominant.

CF:. What software will run on DOS/65 is it still relevant today or is this more an historic environment

That is really up to the user. When all is said and done we all use vintage computers for the fun of the experience. Whether that means productive use, self-education, or pure fun is up to the user.

DOS/65 comes with an editor, assembler, debugger, and other miscellaneous tools. It also comes with BASIC-E/65, a DOS/65 implementation of Gordon Eubanks BASIC-E, the precursor to CBASIC. Small-C is available as is FORTH.

There are no word processor, spreadsheet, or database applications that I know of but there might be a challenge for a DOS/65 user.

CF:. Most user will be aware the Commodore 128 can run CP/M software but the 128 has a z80 processor dedicated
to the task, how can the Commodore 64 run the software

As amplified below, the only way the C64 can run CP/M is by use of the Z80-based CP/M cartridge. DOS/65 does not
give you the ability to run CP/M software without that cartridge but it does allow you to exchange files with a C64 or
C128 CP/M system.

CF:. The z80 based CP/m cartridge for the C64 allowed CP/M software to run on the Commodore 64 can you tell our readers about the device it seems to be an illusive device

I actually have one of the CP/M cartridges for the C64. It has a Z80 processor and some glue logic so it can communicate with the C64. I bought it within the past year so I could see how the 6502 in the C64 and the Z80 in the cartridge communicated and how the Commodore serial bus I/O was handled.

It turns out that the CP/M cartridge is not compatible with all C64s. It will only work reliably with early C64s because Commodore later made some C64 changes that reduced the display flicker and that caused compatibility issues with the cartridge. Commodore never tried to fix the compatibility issues with the later C64s. My C64 is late enough that it falls in the "will not work" category. It will actually boot CP/M but then randomly goes into never-never land.

CF:. What advantages are there in running Cp/m from a cartridge rather than from a Commodore disk

On the C64 you need the cartridge and the CP/M disk but as noted above it is unreliable. The C128 has the Z80 built in so only needs the CP/M software with the C128 specific CBIOS. The C64 CP/M cartridge will not work at all with the C128. In addition the C128 will not run the C64 version of CP/M.

CF:. Does the Commodore 1541 disk drive read CP/M formatted disks?

Yes but only the C64 specific format that uses the standard 1541 GCR disk format. To do more than that under DOS/65 the DOS/65 System Interface Module (SIM), the DOS/65 equivalent of the CP/M CBIOS, must include the drivers for the various 1571 CP/M formats be they GCR or MFM based. The same is true for use of the 1581 or any other formats.

CF:. IS DOS/65 to be updated? and do you intend to update or maintain the software yourself?

If time permits I may add drivers for the other disk formats and perhaps some other minor expansion. Right now I do not plan any other enhancement of the system itself. If bugs are found by users or by myself I will of course fix them and release updated versions. During the C64 effort I actually found a couple of 20+ year old bugs in a couple of utilities and fixed them during the C64 implementation effort.

CF:. If our reader has become excited about the port can he/she help out in anyway?

Absolutely! Anyone can build the system interface code for the 1571 or 1581 drives and a C128 implementation is also doable. In addition porting new tools or enhancing the existing tools are all efforts that some users may find both challenging as well as rewarding.

CF:. Is there anything further you would like to tell our readers

Just that DOS/65 should be enjoyed and viewed as a means of doing some productive tasks but more importantly as a framework to learn more about how a DOS or it's supporting tools work. The source code is there for all to view and examine. While I claim no special talent, it may still help people understand at least one way to do things. There may be better ways and as users explore the code they should not hesitate to suggest them.

CF: .Thanks for the interview

You are welcome and thank you for the opportunity to discuss something that has been part of my life for over 20 years.

Rich

An Introduction to C-128 CP/M

CPM1.TXT rev 1a 96-11-01

Herne Data Systems Ltd.,
PO Box 250, Tiverton, ON N0G 2T0 CANADA.
Voice/fax 519-366-2732,
e-mail herne@herne.com
internet: http://www.herne.com

* *

An Introduction to C-128 CP/M

What is CP/M?
CP/M stands for 'Control Program/Monitor'. It is not a programming language, the way BASIC or Fortran is, but a micro computer operating system, much like the KERNAL used in C-128 native mode and other 6502 based Commodore computers. CP/M was developed in the 1970's by Digital Research Inc. (DRI) for Intel 8080 micro processor based computers that allows a given program to run on a wide variety hardware configurations with little or no changes. It was later extended to run on Zilog Z80 based computers as well as others. (The Z80 used in the C-128 is 'upwardly compatible' with the 8080. That is, its machine language instruction set includes everything that the 8080 has and more.) With CP/M, a software developer has a uniform set of basic operating system functions to work with (such as screen output, keyboard input and disk access routines). To the end user, these system functions remained fixed, no matter what computer the program is actually running on, thus providing a very high degree of portability. (You need only develop one version of a program which operates under the CP/M system instead of having a different version for brand x, y, and z computers.) For those who may be interested, there is also a 16-bit version of CP/M designed to run on 8086/88 type computers called CP/M-86, a multi user version called MP/M and a 16/32 bit version for the MC68000 chip series called CP/M-68k.

CP/M is divided up into four main components: the BIOS (Basic Input Output System); the BDOS (Basic Disk Operating System), the CCP (Console Command Processor) and the TPA (Transient Program Area). Each of these parts will be discussed in later chapters. Of the four parts, only the BIOS is hardware dependent. It acts as the messenger between the other parts of the system and the actual hardware. That is, a separate custom BIOS must be created for each system that runs CP/M.

CP/M on the C-128
The version of CP/M used on the C-128 is termed 'CP/M Plus' or 'CP/M version 3'. This is the latest 8-bit CP/M version, and is also used on other recent CP/M based computers such as the Epson CF:X-10, and TRS-80 Model 4.

The minimum configuration for running CP/M on the C-128 is the C- 128, one disk drive, and a 40 or 80 column monitor. For speed and ease of use, a fast disk drive (i.e. the 1571 or 1581) is recommended along with the 80 column monitor. For disk intensive programs (many CP/M programs are), a 1700 or 1750 RAM expansion unit is highly recommended as it can be used as a super fast RAM disk. A printer (device 4 or 5) along with a modem will round out your system. (Use of the modem requires CP/M release dated DEC 85 or later.)

To start up CP/M on the C-128 or 128 D, put the CP/M BOOT disk in disk drive device 8 and turn on the drive and then the computer (in that order). If the computer and drive are already on, insert the boot disk, then press the reset button on the C-128 or type in BOOT followed by pressing the <return> key. After a few moments and some sign-on status message screens, you will see a prompt on the screen in the form of the letter A followed by a 'greater than' sign and a flashing cursor. (A>_)

Congratulations! You have just started up CP/M successfully and are ready to enter your first CP/M command. Just for fun, type in the letters D I R then press the key marked <return> at the right side of the main keyboard. A 'directory' or 'catalog' of the contents of the boot disk should be displayed on the screen.

The Status Line The bottom line of the screen display (i.e. the 25th line) on either the 40 column or 80 column screen is used by CP/M as a status line to display various system messages. These are outlined below.

Disk Status
C-128 CP/M has an optional disk status window displayed at the bottom right corner of the screen. The format of the status window is as follows:

O Dtt ss

```
where:O    =Operation, either R or W indicating a Read or Write
      D    =the logical drive (A,B,C,D, etc)
      tt   =the track number currently being read or written
      ss   =the sector number currently being read or written
```

Normally, the track and sector numbers are separated by a space. If the disk is formatted in MFM, and the track and sector are separated by a '-', then the second side of the disk is currently being accessed.You can toggle this status window on or off by typing the <CONTOL>-<RUN/STOP> key combination. Note that window is also erased temporarily, i.e. until the next disk access, when you (or your application program) perform a screen clear.

The disk status window is not used for RAM disk operations (i.e. 1700/1750 RAM expansion unit). You may also note that, due to a bug in the fast bus GCR sector translation routine, the sector numbers displayed during reading and writing of both single and double sided C-128 native GCR type CP/M disks are incorrect when using a 1571 drive. Correct numbers are, however, displayed when using a 1541 drive to access these disks.

MFM Disk Type Selection
C-128 CP/M supports a number of industry standard MFM disk types (such as KayPro, Osborne, IBM, Epson, etc.) when used with a 1571 disk drive. However, CP/M sometimes cannot automatically distinguish between two very similar disk types such as KayPro II and KayPro IV. For example, although the KayPro II is a single sided disk and the KayPro IV is a double sided one, CP/M can confuse the two because both have ten sectors of 512 bytes per track, numbered 0 to 9 on side 0. CP/M looks at the physical format of side 0 to determine the foreign disk type. When two or more matches are found in its internal parameter table, you are asked to manually chose which one you want.

When CP/M cannot uniquely identify the foreign disk format being accessed, the program will pause momentarily and the disk type window will open up at the center of the bottom line of the screen to display the name of the first match, such as KayPro II, in reverse video. At this point there are several things that you can do.

First, if you are sure that the disk is of the type displayed,simply press <return> or <enter>. This will select the currently displayed entry. If you change disks, you will have to go through the selection process over again.

Second, if you are going to be using this disk format repeatedly and do not wish to re-select each time you change disks, you can 'lock in' the selected type by pressing <control>-<return>. This action has the effect of performing a burst mode 'forced login' on the drive each time you change disks. If you insert a new type of disk while locked in, you will go back to the selection window.

Note:
The forced login commands are different for the 1571 and 1581 drives and therefore, <control>-<return> should not be used with a 1581 drive. If you do use it on a 1581 and change disks, the results may be unpredictable, even resulting in system crash

The third option is used to display and select the alternate disk choices. Use the cursor left and cursor right keys in the cursor control keypad (above the main keyboard) to scroll through the list of choices. In the example used above, the display should change between KayPro II and KayPro IV. When the correct entry is highlighted, you can use either of the two methods outlined above for making the selection. Note that because you are in a suspended program loop, nothing else can be done until you make a selection.

PAUSE
The key labelled NO SCROLL executes a PAUSE function. Although this key appears to have the same effect as typing <control>-s (stop) and <control>-CF: (start) for pausing to read screen dumps, it is not actually related in terms of the internal workings of the operating system. Using NO SCROLL sends the program into an endless loop while displaying the message 'PAUSE' on the status line. Because of this, NO SCROLL cannot be used to pause a display before pressing <control>-p to toggle the printer, or <control>-c to abort execution of a program.

To resume execution after a PAUSE function, press NO SCROLL again (for AUG or DEC release only) or RETURN (for MAY release only).

Disk Drives
C-128 CP/M supports a variety of different disk drives (virtually any serial bus drive is either supported directly or can be easily adapted). However, since CP/M tends to be a disk intensive operating system (i.e. it makes heavy use of temporary disk files, overlay files, working files, etc.), a fast bus drive (i.e. a 1571 or 1581) is highly recommended for maximum disk access speed.

CP/M in general can support up to 16 disk drives, usually labelled A: to P:. On the C-128 implementation, normally only a maximum of6 drives are actually supported. In the default system configuration, these are:

DRIVE	DEVICE
A:	serial port device 8, drive 0
B:	serial port device 9, drive 0
C:	serial port device 10, drive 0
D:	serial port device 11, drive 0
E:	serial port device 8, drive 0 (see 'Virtual Disk Drive' below)
M:	17xx RAM expansion unit (see 'RAM Disk' below)

Virtual Disk Drive
The physical disk drive A can be used as logical drive A or as the 'virtual' or 'phantom' logical disk drive E. Whenever you access drive E (for example by doing DIR E:, or PIP E:=A:FILE1), the system will display a prompt on the status line asking you to insert the diskette for drive E (respond by putting whatever diskette you want into physical drive A, then type <return>). The next time you access logical drive A, the system will prompt you to insert the A diskette in drive A. Again, insert the desired diskette and type <return>. Note that CP/M does not actually check to see if you in fact changed the disk. It will accept whatever is in the drive when you press <return> as being the correct disk.

RAM Disk
When the 1700 (128k byte) or 1750 (512k byte) RAM expansion unit (REU) is connected to the C-128 in CP/M mode, it becomes an ultra fast RAM disk, normally assigned to CP/M drive M:. That is, any CP/M command or transient utility can access the REU as drive M:, just like it can any other 'disk drive' connected to the system. Note that this is totally separate from the virtual disk drive discussed above.

Unlike a physical disk drive, the contents of the RAM disk are not preserved when the C-128 is turned off. However, the contents are preserved when the reset button is pressed momentarily to re-boot the system, even if you enter C-128 native mode or C-64 mode before returning to CP/M mode (providing, of course, you do not alter the contents either accidentally or intentionally by running some other program while in one of these other operating modes).

Note that in order to use the RAM disk, you must have the C-128 CP/M release dated 6 DEC 85 or later.

The Screen
C-128 CP/M supports both the 40 column and 80 column display screens. The screen which is initially active is determined by the position of the 40/80 switch on boot up. After booting, the screen can be selected using the DEVICE utility (see Chapter 3) or equivalent. The screen can be changed during the execution of a user program by setting the CONOUT redirection flag in the system control block (see Chapter 9).

The 40 column screen is configured as a window on a larger 80 column 'virtual' screen. This allows commands and
programs to think that they are dealing with an 80 column wide screen (the default width for most CP/M programs). Use <control>-<left arrow> and <control>-<right arrow> to scroll the window across the virtual screen (see Arrow Keys, below).

Printer Support
C-128 CP/M can support two printers, serial port devices 4 and 5. In its default configuration, device 4 (CP/M physical device PRT1) is assigned to the logical LIST device with a secondary address of 7 and ASCII to PET-ASCII translation enabled. To change to a device 5 printer, use the DEVICE utility (see Chapter 3). To disable ASCII to PET-ASCII translation, use the CONF utility.

The Keyboard
The operation of the C-128 keyboard is slightly different in CP/M mode than it is in native mode. For one thing, it uses the standard 7 bit ASCII character set rather than the PET-ASCII set used in native mode. In addition, each key is completely programmable as to its character codes or other special functions. Other differences in keyboard operation are outlined below.

Alpha Keys
The alpha keys on the main keyboard (e.g. A, B, C, D, etc.) generate the standard lower case, upper case, CAPS

LOCK and control values. The CAPS LOCK value defaults to the upper case value and is used as a software shift lock. It is not related to either the SHIFT LOCK key or the CAPS LOCK key. To turn the CAPS LOCK mode on, press the Commodore logo C= key at the lower left of the main keyboard. Pressing the same key again, will turn off CAPS LOCK mode.

Numeric Keys
The numeric keys (the row of keys above the alpha keys) produce the appropriate number as the unshifted value and the CAPS LOCK value, the symbol above the number on the key as the shifted value and the color on the side of the key (the top one of the two colors shown) as the control value. These colors are defined as 80 column foreground colors. The control value produced by the numbers on the numeric keypad are defined the same as those on the numeric keys, except that they generate 80 column background colors.

Arrow Keys
The arrow keys are located in the cursor keypad above the top right portion of the main keyboard. The functions and character codes assigned to these keys in CP/M mode are different to those assigned to the cursor control keys at the lower right of the main keyboard. The default values for the arrow keys are those use used by most application programs which feature full screen cursor movement, such as text editors, etc.

The up arrow key generates the value 05 (<control>-e) as the unshifted, shifted and CAPS LOCK values; it generates the hexadecimal 12 (<control>-r) as the control value.

The down arrow key generates the hexadecimal value 18 (<control>-x) as the unshifted, shifted and CAPS LOCK values; it generates the value 03 (<control>-c) as the control value.

The left arrow key generates the hexadecimal value 14 (<control>-s) as the unshifted, shifted and CAPS LOCK values; it executes the 40 column screen left function as the control value.

The right arrow key generates the value 04 (<control>-d) as the unshifted, shifted and CAPS LOCK values; it executes the 40 column screen right functions as the control value.

Additional ASCII Characters
The key labelled as a British pound sign generates the character '#' as the unshifted, shifted and CAPS LOCK values. The reverse apostrophe character ` is generated as the control value.

The up arrow key next to the <RESTORE> key generates the caret character '^' as the shifted and control values. The pipe character '|' is generated as the unshifted and CAPS LOCK values.

The '=' key generates the character '=' as the unshifted, shifted and CAPS LOCK values. The tilde character '~' is generated as the control value.

The '/' key generates the character '/' as the unshifted, shifted and CAPS LOCK values. The back slash character '\' is generated as the control value.

The key labelled 'INST DEL' generates the hexadecimal value 7f (ASCII rubout) as the unshifted, shifted and CAPS LOCK values; the hexadecimal value 16 (<control>-v) is generated as the control value.

Special Key Functions
The key labelled NO SCROLL executes a PAUSE function. To resume execution, press NO SCROLL again (for AUG or DEC version only) or RETURN (for MAY version only). The control value of the NO SCROLL key toggles between two modes (on/off) of automatically tracking the CP/M screen on a 40 column monitor.

The right and left arrows at the top of the keyboard initiate the special functions screen right and screen left respectively (40 column screen tracking) as their control values.

The key labelled <ENTER> generates a Carriage Return as the unshifted, shifted and CAPS LOCK values. The control value initiates the special function boot, which reboots the system as if the reset button has been pressed. If a CP/M boot disk is in drive A:, the CP/M system will be re-started, otherwise normal C- 128 native mode will be started.

The key labelled <RUN/STOP> is undefined as its unshifted, shifted and CAPS LOCK values, and toggles the display/nondisplay of the disk status window as its control value.

The key labelled CRSR with up and down arrows (bottom right on main keyboard) generates the hex value 17 (<control>-w) which causes the latest CP/M command line to be re-typed on the screen. The key next to this (CRSR with right and left arrows) allows you to move right (<control>-f) and left (<control>-a) (unshifted and shifted, respectively) through a command line to edit it.

The key labelled F1/F2 generates the two character string 'F1' in the unshifted and CAPS LOCK modes and the string 'F2' in the shifted and control modes.

The key labelled F3/F4 generates the four character string 'dir <return>' in the unshifted and CAPS LOCK modes and the string 'dir' in the shifted and control modes.

The key labelled F5/F6 generates the two character string 'F5' in the unshifted and CAPS LOCK modes and the string 'F6' in the shifted and control modes.

The key labelled F7/F8 generates the two character string 'F7' in the unshifted and CAPS LOCK modes and a string representing the date that the system files were assembled in the shifted and control modes (i.e. the date which is displayed in the sing-on message).

The key labelled HELP generates the string 'Help' in all modes.

The Command Line
CP/M commands and transient programs are accessed from the system prompt by typing in a sequence of characters called a command line. A CP/M command line is composed of a command, an optional command tail, and a carriage return. The command is the name or filename of a program to be executed. The optional command tail can consist of a drive specification, one or more file specifications, and some options or parameters.

A>COMMAND {command tail} <cr>

Command Conventions
The following special symbols define the syntax of a normal command line.

{}	surrounds an optional item.
|	separates alternative items in a command line.
<cr>	indicates a carriage return.
^	indicates the Control Key.
n	substitute a number for n.
s	substitute a string (group) of characters for s.
o	substitute an option or option list for o.

[]	type square brackets to enclose an option list.
()	type parentheses to enclose a range of options within an option list.
RW	Read-Write attribute - opposite of RO
RO	Read-Only attribute - opposite of RW
SYS	System attribute - opposite of DIR
DIR	Directory attribute - opposite of SYS
...	preceding element can be repeated as many times as desired.
*	wildcard: replaces all or part of a filename and/or filetype
?	wildcard: replaces any single character in the same position of a filename and/or filetype.

Command Line Editing
The resident CP/M command interpreter (CCP) allows a certain degree of editing of the CP/M command line before pressing the <return> key. The editing keys are outlined below along with their usage. It should be noted that these editing keys are totally independent from the control codes used by the terminal emulation for the video display.

Control Character	Function
CTRL-A	moves cursor one character to the left.
CTRL-B	moves cursor from beginning to end of command line and back without affecting command.
CTRL-C	stops executing program when entered at the system prompt or after CTRL-S.
CTRL-E	forces a physical carriage return without sending command.
CTRL-F	moves cursor one character to the right.
CTRL-G	deletes character at current cursor position if in the middle of a line.
CTRL-H	delete character to the left of cursor.
CTRL-I	same as the TAB key.
CTRL-J	moves cursor to the left of the command line and sends command to CP/M. Line feed, has same effect as carriage return.
CTRL-K	deletes character at cursor and all characters to the right.
CTRL-M	same as carriage return.
CTRL-P	echoes console output to the list device.
CTRL-CF:	restarts screen scrolling after a CTRL-S.
CTRL-R	retypes the characters to the left of the cursor on a new line; updates the command line buffer.
CTRL-S	stops screen scrolling.
CTRL-U	updates the command line buffer to contain the characters to the left of the cursor; deletes current line.
CTRL-W	recalls previous command line if current line is empty; otherwise moves cursor to end of line.

CTRL-J,-M,-R,-U and RETURN update the command line buffer for recall with CTRL- W.

CTRL-X deletes all characters to the left of the cursor.

Filespec
CP/M identifies every file by its unique file specification,= which can consist of four parts:

the drive specification,
the filename,
the filetype and
the password.

The term "filespec" indicates any valid combination of the four parts of a file specification, all separated by their appropriate delimiters. A colon must follow a drive letter. A period mustprecede a filetype. A semicolon must precede a password.

The general symbols and rules for the parts of a file specification follow:

d:	drivespec	optional	single alpha character (A-P)
filename	filename		1-8 letters and/or numbers
typ	filetype	optional	0-3 letters and/or numbers
password	password	optional	0-8 letters and/or numbers

If you do not include a drive specifier, CP/M automatically uses the default drive. Valid combinations of the elements of a CP/M file specification are:

```
filename
d:filename
filename.typ
d:filename.typ
filename;password
d:filename;password
filename.typ;password
d:filename.typ;password
```

Some CP/M commands accept wildcard (* and ?) characters in the filename and/or filetype parts of the command tail. A wildcard in the command line can in one command reference many matching files on the default or specified user number and drive. This is sometimes referred to as an 'afn' or ambiguous filename. If no wildcards are present, an 'ufn' or unambiguous filename results.

COMMODORE FREE
I would like to thank Herne and 'Herne Data Systems Ltd.' For granting the reprint permission of this text
From: Herne <herne@herne.com>
Subject: Commodore Free magazine
To: Commodore Free Magazine

You have our permission to run the text. Pls be sure to send us a copy for our records when it appears.
regards, Mike

Herne Data Systems Ltd.
31 Adelaide St E, PO Box 357
Toronto, ON M5C 2J4
CANADA

CP/M the Operating System for microcomputers of the 1980`s

Mrs. Gaby Chaudry
http://www.gaby.de

By the end of the 1970`s and early 1980`s CP/M was the standard operating system for microcomputers. On top of that, it was the first DISK Operating System (DOS), and therefore the basis for today's DOS, and in first place Microsoft?s DOS. MS-DOS itself derived from nothing more than a CP/M clone called QDOS (Quick and dirty operating system) written by Tim Patterson.

Microsoft bought QDOS from him to use with the first IBM PC. Many former CP/M programs have been ported to DOS, first of all DBase and WordStar. CP/M itself on the other hand is based on the Operating System for the DEC PDP-10, called "TOPS-10", at least as far as naming conventions for files and devices and some commands are concerned. Surely, there are other influences, but they remain untold.

Many of the computing things that we take for granted today, have there roots back to the work of a single man, that man was Gary Kildall, the developer of CP/M. Who has ever thought, why his my hard disk is called C: or where does the DOS command DIR comes from?

Gary Kildall was born in 1942, and received a Ph.D. in Computer Science in 1972, soon he fulfilled a draft obligation to the United States Navy by teaching at the Naval Postgraduate School in Monterey, California. When Gary attended the introduction of the new Intel processor 8080 in 1973, he was so enthusiastic about it, that he suggested to the Intel managers to write a compiler (i.e. a program that builds a ready-to-run program out of the program code) for the language PL/1 (Programming Language Number 1). At this time, PL/1 was often used on mainframes, so that the Intel people agreed immediately; the program was the called (= Programming Language for Microprocessors).

There was only a small problem: Gary didn?t own a computer that ran an 8080 CPU. He had access to a Digital Equipment PDP-10 only, and so Gary built his PL/M compiler in FORTRAN on the PDP machine. when the compiler was ready, he needed an 8080 computer for testing. He even managed to convince Shugart to donate him a floppy drive. But, since cable, power supply and controller were missing; it didn?t help him very much.Together with Gordon Eubanks (who later became CEO of Symantec) he built an 8080 based computer.

For this machine he wrote an operating system which he called CP/M (Control Program/Monitor). He offered a package containing CP/M, PL/M, and a development system to Intel for 20.000 US$. Intel wasn?t interested in CP/M , but they agreed to buy PL/M. So Gary at least had success with his compiler and decided to put CP/M on the market on his own. Together with his wife, he established a company called "Intergalactic Digital Research Inc.", which later was renamed to "Digital Research Inc." ('DR'). Gary started to sell his CP/M through mail order in technical magazines.

It should be mentioned that exactly at that time many enthusiasts began to build computers on their own (e.g. Altair and IMSAI offered their now famous assembly kits). What those computers were missing, was on operating system, so CP/M turned up in the right place at the right time.In 1979, DR published CP/M version 2.0, which soon was followed by CP/M 2.2. This version was the most popular one and was used worldwide, as it offered high performance with only small amounts of memory usage. The next version, 3.0, also called CP/M Plus, but wasn?t offered before 1982. Unfortunately, this was one year too late, since in 1981 IBM started its triumph.

What was so special about CP/M?You should be aware of the fact that microcomputers at that time were equipped quite badly and often had to be programmed using switches (e.g. the Altair and IMSAI 8080 models). If ever, there was a programming language like e.g. BASIC, which made it possible to write and save simple programs. An Operating System on the other hand offered possibilities, that today we take for granted, but at this time were pure luxury.

In first place, it had a standardized user interface, which made it possible to run the same commands and programs on a couple of different machines, so that the user didn`t need to care about the hardware any more. It was just the manufacturer who had to adopt the CP/M to his hardware. But, since this was quite easy to be done, CP/M became very popular. CP/M was based on three components:

1. The BIOS (Basic Input Output System): This was the part that controlled the basic communication between hardware and software, and therefore was different with different hardware. It consisted of functions as e.g. reading letters from the keyboard, show characters on the screen, or reading a sector from a floppy disk. A computer without BIOS is unthinkable nowadays.

2. The BDOS (Basic Disk Operating System), the "brain" of the CP/M. It was the place where communication between the command interpreter and the BIOS took place, i.e. the inputs and outputs were translated and forwarded to the CCP.

3. The CCP, the command interpreter. This was a simple program that had some built-in commands (e.g. DIR to show the directory listing) and could start other programs. In 1981, CP/M did already run on about 300 different computer types. It even ran on machines without 8080 or z80 CPU. Microsoft sold the so-called "Softcard", which made it possible to run CP/M on an Apple computer. Believe it or not, but at this time they had more income with CP/M than with their own software. As time passed, other CP/M versions followed: those for the 16 bit CPU 68000 (CP/M-68000) and Intel 8086 (CP/M-86).

In 1980, the decisive turn in the history of Digital Research took place. IBM developed the PC, and of course they needed an Operating System for it. CP/M seemed to be the idealproduct, as enough software for all scopes was available for it. By mistake they thought that Microsoft produced CP/M (as they sold the Softcard along with CP/M), and they made an appointment with Bill Gates, just to learn that Digital Research was the right contact. Gates sent the IBM managers to the nearby town of Monterey.

When they arrive there, Gary wasn?t at home though. Nobody knows exactly what happened at that time, but it seems that Dorothy McEwen, Gary`s wife, who managed the company, didn?t agree to IBM's terms and therefore cancelled the negotiations.IBM's people addressed themselves to Bill Gates again, who realized the great chance and bound himself to provide an Operating System. When Gary came to know that PC-DOS was a plagiarism of CP/M (what he could even proof), he filed a lawsuit against IBM. But shortly after that he abandoned it again, because IBM offered him a generous compensatory payment. In the same course, they agreed to offer CP/M alternatively to DOS together with their PC. What they didn?t mention: they charged three times the price of DOS for the CP/M.

Of course the available range of software for CP/M was much larger, but because of the price most customers choose DOS. Anyway, during this period many machines where offered that had both the 8086 and the z80 CPU, and even z80 cards for the IBM PC where available, so that many customers could still use CP/M together with DOS. Some companies even kept on offering CP/M and its successors with their x86 machines, e.g. Siemens. Digital Research didn?t become inactive, though, and published a couple of innovative products in the following years.

An example was Concurrent CP/M, a real Multitasking System, which was efficient and easy to operate, because it used the well known CP/M commands. Later on they issued DR-DOS, a MS-DOS compatible, but much more comprehensive Operating System. In 1991, Gary Kildall sold Digital Research to Novell. After that, he was involved in other developing projects, e.g. the file system for Multimedia CD ROMs. DR also brought out the graphics user interface GEM, which wasn?t that popular for CP/M or DOS users, but then was licensed by Atari and became well-known as the GUI of the ST series.

In any case, the choice of computers that ran CP/M was quite large at the beginning of the 1980ies. It can even be said that in the Personal Computer sector the Digital Research Operating System was ruling the market. Software was available for all kinds of applications. Freeware was offered in an extensive range so that CP/M computers became affordable for almost everybody. There was only one disadvantage in CP/M:

the mostly incompatible hardware. I.e. an Osborne isn?t able to read floppies formatted on a Kaypro and vice versa. When you wanted to exchange data between different computers, you had to use either a converter program or a serial connection between the machines. On the other hand, at that time modems and acoustic couplers already existed, so that users could connect to a mailbox and download programs and files.Of course, at that period, files had completely other dimensions than those to which we are used today. You must consider that a z80 based CP/M machine didn`t have more than 64 Kbyte of memory, and a floppy usually could save not more than 360 Kbyte – imagine how memory thrifty programs had to be under these circumstances. Comprehensive programs therefore used a modular structure, which made it possible to load only those parts into memory that were currently needed.

Hard disks were rare and nearly priceless. Nevertheless, CP/M was able to manage partitions up to a size of 8 Mbytes. If somebody had to maintain a bigger project with higher amounts of data therefore had a distinct advantage when using a hard disk instead of a higher quantity of floppy disks. An example of a CP/M machine with hard disk was the Kaypro 10, which had a built-in 10 Mbyte hard disk.

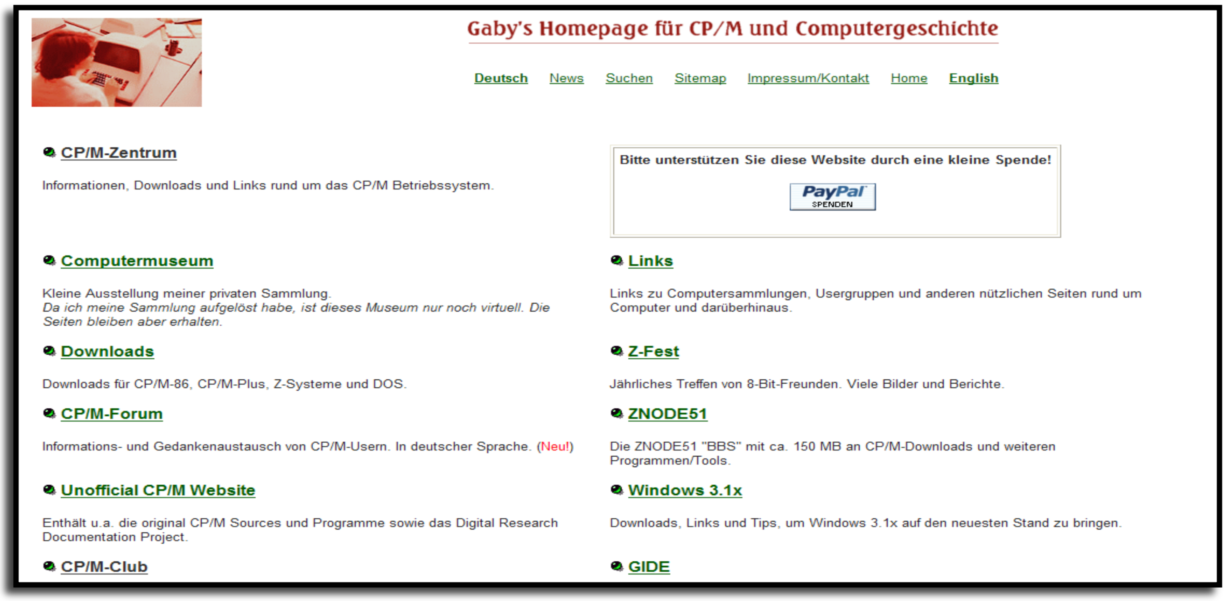
One reason for the popularity of the CP/M successor DOS surely was (apart from the price) the similarity of its structure and user interface. For somebody changing from CP/M to DOS, it was easy to become acquainted to the new system. The only big difference was that DOS introduced sub-directories to the microcomputer world, where CP/M used to work with User areas. Both solutions had their roots in mini and mainframe systems, though.

The most important CP/M commands were:
DIR to show the directory listing
STAT to show drive and media information and to change attributes
PIP to copy
ED line editor for text files
SYSGEN for creating a system disk

Even other conventions were ported from CP/M to DOS, e.g. the 8.3 convention for file names (FILENAME.TXT), the usage of the asterisk and question mark as wildcards and the extension .COM for executable files. A typical CP/M command e.g. could be: DIR B:*.* or PIP B:=A:*.TXT, where in the latter example the copying direction is inverted in comparison to DOS. This may result in the Assembler convention, a programming language many user were acquainted to at this time.

It is justifiable to say that if history had developed slightly different, its possible that we might work with CP/M and GEM (or their successors) instead of DOS and Windows today.

EDITOR
I would like to thank Gaby for contacting me regarding CP/m and offering this text, also I would like to thank her for converting the text from German to English.


Gaby's Homepage für CP/M und Computergeschichte

Deutsch News Suchen Sitemap Impressum/Kontakt Home English

CP/M-Zentrum
Informationen, Downloads und Links rund um das CP/M Betriebssystem.

Bitte unterstützen Sie diese Website durch eine kleine Spende!
PayPal SPENDEN

Computermuseum
Kleine Ausstellung meiner privaten Sammlung.
Da ich meine Sammlung aufgelöst habe, ist dieses Museum nur noch virtuell. Die Seiten bleiben aber erhalten.

Links
Links zu Computersammlungen, Usergruppen und anderen nützlichen Seiten rund um Computer und darüberhinaus.

Downloads
Downloads für CP/M-86, CP/M-Plus, Z-Systeme und DOS.

Z-Fest
Jährliches Treffen von 8-Bit-Freunden. Viele Bilder und Berichte.

CP/M-Forum
Informations- und Gedankenaustausch von CP/M-Usern. In deutscher Sprache. (Neu!)

ZNODE51
Die ZNODE51 "BBS" mit ca. 150 MB an CP/M-Downloads und weiteren Programmen/Tools.

Unofficial CP/M Website
Enthält u.a. die original CP/M Sources und Programme sowie das Digital Research Documentation Project.

Windows 3.1x
Downloads, Links und Tips, um Windows 3.1x auf den neuesten Stand zu bringen.

CP/M-Club

GIDE

Commodore Computer club Forum post

I've had the following email from Commodore128.org about a new 128 application:

"Clock-Calendar 128 (CC128) is now available for downloading. See the link at the end of this post and be sure to read the documentation before trying it out. Also, if you wish to integrate CC128 with the C-128 80 Column Display Manager (80CDM) you will need to download the latest 80CDM version, as the previous version is not compatible with CC128.

Clock-Calendar 128 is a utility for the C-128 that provides date and time services that may be used by other programs on an ad hoc basis. When started, CC128 is linked into the C-128's interrupt request (IRQ) processing subsystem, causing the utility to run in the background until its services are needed. There are any number of uses to which CC128 can be put, such as providing date and time services to BBS software, date and time stamping of records in a database, date and time stamping of entries into a log, etc.

CC128's basic features are:
- 12 hour, AM/PM continuous time-of-day function. All CC128 timing functions are derived from the hardware TOD clock in CIA #2. Unlike the TI$ software jiffy "clock" available in BASIC, the CIA's TOD clock is unaffected by operating system activities and maintains excellent accuracy over long periods.

- Calendar with day of week, year, month and date. CC128 can maintain dates from January 1, 1753 to December 31, 9999, inclusive, with full leap year compensation.

- Audible alarm. At the alarm time, your C-128 will alert you with a pleasant gong tone. The alarm has the software equivalent of a "snooze bar."

- Continuously updated date and time values. CC128 can export date and time data into any desired location in RAM-0 for use by your software.

- Automatic detection of NTSC and PAL systems. Upon startup, CC128 will determine which video system is in use and will program the TOD clock for the correct power line (mains) frequency.

- Presence signature. CC128 includes a "presence signature" that can be tested by other programs to determine if CC128 has been loaded into memory and is intact..

-CC128 can interface with the C-128 80 Column Display Manager (80CDM) and offer the following extra services: Timed "screen saver" function. CC128 can be configured to monitor the C-128's keyboard and turn off the 80 column display after a programmable period of inactivity. Pressing almost any key will turn on the display. The inactivity period can range from one to 32,767 seconds (more than nine hours).

-Continuous date and time display. If 80CDM's status line has been enabled, CC128 can be told to generate a constantly-updating date and time display on the right-hand end of the status line (leaving plenty of room for user-generated status line text).

-CC128 runs in RAM-0 underneath the kernel ROM and consumes less that 2 KB. Data is stored in normally unused memory at the very top of RAM-0. When loaded and started, CC128 will adjust the top-of-BASIC pointers to protect itself from outsized programs.

-CC128 is a low level operating system enhancement, and thus natively operates at the machine language level. Access is through a jump table similar to the kernel ROM jump table, and date and time inputs and outputs are given in compressed binary-coded decimal (BCD) format for easy manipulation and conversion in the M/L environment. However, BASIC programmers have not been left out in the cold! I have also developed CC128CTL (CC128 control, included in the CC128 distribution) to allow CC128 functions to be called from BASIC, using a simple SYS calling syntax that even a trained monkey can understand. :

If you have any questions about the use of CC128 please peruse the documentation. If that doesn't help then feel free to post here. If you are reporting what you think may be a bug, please be as specific as possible as to the problem. Posts that lack adequate information or aren't germane to the topic will be ignored.

To unsubscribe from these announcements, login to the forum and uncheck "Receive forum announcements and important notifications by email." in your profile.
You can view the full announcement by following this link:

http://landover.no-ip.com/forums/index.php?topic=2266.0
Regards,
The Commodore 128 Alive! Team."

Regards,
Shaun.

WORLD'S GREATEST MENU
By Alan Reed

There are probably a million "menu" programs out there. If the menu is the first program on your disk, users can LOAD"*",8:RUN and be presented with a list of programs on the disk. I've always been fascinated by these programs -- I've humbly called my attempt "World's Greatest Menu." I think it surpasses most of the old generic menus you've seen before in a few ways.

WGM is written entirely in DotBASIC plus and has the following features:

* Works with all drives/device numbers.

* Looks nice. Uses a custom character set, and is generic enough to be suitable for any collection of programs or text.

* Uses a mouse in port 1, a joystick in port two, or CRSR keys and RETURN. Press 'Q' to QUIT if you are using the keyboard only.

* If the selected file is of the Mr.Edstar persuasion (prefixed with a 'T.' the program will let you read and/or print the text file.

* If you place a SID music file on your disk (prefixed with a 'M.') WGM will play the music. The SID must be less than 15 disk blocks long.

WGM is the natural evolution of the simple menu program I presented as a tutorial in the DotBASIC Plus manual.
The main differences are the addition of an FTS screen, a second clean character set for the text reader, and a few lines of code to play the first SID file found on the disk.

There are lots of REM statements in the program listing detailing what WGM is doing. I invite you to take a look to see how simple programming in DotBASIC Plus can be.

Enjoy! http://8bitcentral.com/

Alternative Programming Languages: C Part 1 By Paul Davis

In the first article of this series we explored some of the capabilities of the Forth language. Forth has many unusual characteristics that make it well suited to programming on the Commodore, but it's a language that polarises opinion. Some people love it, some hate it, and some just can't seem to wrap their head around it. So this time we're going to look at a more mainstream language, C.

C is a compiled language. It's not like BASIC or Forth where there's an interpreter you can type commands into and experiment with. Program code is entered into text files which are then run through a compiler to translate the source code into an executable program. This makes programming in C a bit more involved than you may already be used to. Nevertheless, C is a very powerful and flexible language and is well worth adding to your programming repertoire.

Although there are several implementations of C that run on the Commodore, the lack of memory and poor disk drive performance makes writing programs a laborious process. With that in mind, it provides a good opportunity to introduce the concept of cross-development. That is, using a modern computer to edit and compile our programs into a form that can be run either in an emulator or on a real Commodore machine.

As was the case with the Forth article, this is not a complete tutorial. It's merely an introduction to some of the features and programming style of the language to give you an idea of how it can be used. I'm going to follow a similar format as before and present some example programs with a brief commentary on how they work.

Getting started
We're going to be using a freeware C compiler called cc65 which is available in executable form for Windows, and as source code that can be compiled for OS X and Linux. The cc65 web site can be found at http://www.cc65.org/. We will also be using the VICE emulator to test our programs. If you don't already have this, it can be downloaded from http://www.viceteam.org/.

The first step is to download and install cc65. Point your browser or favourite FTP client at the download area: ftp://ftp.musoftware.de/pub/uz/cc65/ then follow the relevant instructions for your OS below.

Windows XP and Vista
Download the files called cc65-win32-2.12.0-1.zip and cc65-c64-2.12.0-1.zip. Next, create a directory where you want to install the compiler. This can be anywhere you like, although I would recommend avoiding the 'Program Files' directory (especially on Vista) or directories with spaces in their name. This article will assume the directory is C:\cc65. To install the program, extract the zip files you downloaded into the cc65 directory you have just created.

Before we can use the compiler we need to set up our programming environment at the command prompt. Open a command prompt window (hold down the Windows key, press R then type 'cmd' and press Enter) and type the following commands:

```
path %path%;c:\cc65\bin
set CC65_INC=c:\cc65\include
set CC65_LIB=c:\cc65\lib
```

These instructions tell the command shell where to find the cc65 programs and tell cc65 where to find various files that are needed to compile a program.

You will need to enter these commands each time you open a new command prompt window. If you want the settings to be permanent, here's how to do it. Press Windows + Break (or right-click on 'My Computer' and choose 'Properties') to bring up the System Properties dialog. On Vista select 'Advanced system settings' from the task panel on the left and confirm the UAC dialog. In the System Properties dialog click on the 'Advanced' tab, then on the 'Environment Variables' button. In the bottom panel for system variables click on 'New' then enter 'CC65_INC' for the name and 'c:\cc65\include' for the value and click 'OK'. Do the same for the 'CC65_LIB' variable with the value 'c:\cc65\lib'. Next, scroll down the list to find the entry for 'Path' and double click it. Click on the value field and move the cursor to the end of the value then add ';c:\cc65\bin' to the end and click 'OK'. Click on the 'OK' button of the Environment Variable and System Properties dialogs to complete the settings.

Finally, we need to create a directory within the cc65 directory to store the files we will create during this tutorial.

```
cd c:\cc65
mkdir tut
cd tut
```

OS X and Linux
To keep these instructions as short as possible I'm going to describe the procedure for OS X and assume Linux users will know how to deal with any differences.

First, to compile cc65 we need to install the Xcode developer tools. Insert the OS X installation DVD that was shipped with your Mac and follow the links to install Xcode.

Next, download the file called cc65-sources-2.12.0.tar.bz2 from the cc65 web site. You will not need to download any of the other files. Assuming the file has been saved to your 'Downloads' directory, open a new Finder window and navigate to that directory. Double-click on the cc65-sources file and the Archive Utility will extract it, creating a directory called 'cc65-2.12.0' in the Downloads directory.

Now we need to work at a command line, so navigate to Applications then Utilities and double-click the Terminal program. Enter this command to change the current directory to the cc65 directory in Downloads:

```
cd ~/Downloads/cc65-2.12.0
```

Now type this command to build the cc65 program suite:

```
make -f make/gcc.mak
```

This will take a minute or two to run. When the command prompt returns enter these lines:

```
sudo mkdir /usr/local
sudo make -f make/gcc.mak install
```

You will probably need to enter the administrator password to run those commands. They will install cc65 into /usr/local/lib/cc65 and the executables into /usr/local/bin.

At the end of the installation, there will be a message listing two variables that need to be set up. We will do this in a minute, but first we need to create a work directory to store the files that will be created during this tutorial. Enter these lines:

```
cd ~/Documents
mkdir cc65
cd cc65
```

To make setting up the environment variables a bit easier we will create a script to do it for us. Enter this command:

```
xed -xc init.sh
```

The Xcode editor will start up and show a blank document. Enter the following lines into the file:

```
export CC65_INC=/usr/local/lib/cc65/include
export CC65_LIB=/usr/local/lib/cc65/lib
alias edit='xed -xc'
```

Press Cmd+W and confirm the dialog to save the file. Finally, enter this command in the terminal window to run our script:

```
. init.sh
```

That's a dot and a space before the init.sh. This command will need to be run in any new terminal window when you want to use cc65.

Creating your first program
Because C is a compiled language, the program code is entered into plain text files. These files should have a '.c' extension. The cc65 compiler is then used to translate the source code file into an executable program.

To keep things as simple as possible we will use the standard tools provided by the operating system. On Windows we will use Notepad for entering the source code. On OS X we will be using the Xcode editor which you have already seen. If you are using Linux replace 'notepad' or 'edit' in the instructions below with your favourite text file editor. Windows users enter this command:

```
notepad first.c
```

Notepad will ask if you want to create a new file, answer yes. OS X users enter this command:

```
edit first.c
```

Now type in (or copy and paste) the program below. To get the # on a Mac with a UK keyboard press alt/option and 3.

```
// My first C program

#include <stdio.h>

void main(void)
{
   puts("It works!");
}
```

Press Ctrl+S on Windows or Cmd+S on OS X to save the file. Now flip back to the command prompt/terminal window. Windows users enter the command:

```
dir
```

OS X or Linux users enter the command:

```
ls -l
```

The directory should show our 'first.c' file. Now we need to compile this into a C64 program. To do this, enter the command:

```
cl65 first.c
```

If all goes well, the command prompt should return with no other output (cc65 follows the Unix philosophy of 'no news is good news'). If you get an error message, swap back to the editor window, make any necessary corrections, save the file, then enter the cl65 command again at the command prompt.

Once your program has compiled without errors, enter the 'dir' (or 'ls') command again and you should see a couple of new files, 'first.o' and 'first'. The 'first.o' file is an intermediary file created during the compilation. The file called 'first' is our C64 program.

Next, load up x64, the VICE C64 emulator. Select the option called 'Autostart' or 'Smart attach' in the 'File' menu. On Windows navigate to the c:\cc65\tut directory and select 'All files (*.*)' as the file type filter. OS X users navigate to the Documents/cc65 directory. Now double click on the file called 'first' and our program should load and run, displaying 'It works!' on the screen.

That is, in a nutshell, the process for creating programs in C. This 'edit, compile, run' sequence of actions will soon become second nature.

Okay, let's take a look at the program code in more detail. The first thing to notice is that the code is written predominantly in lower case. C is case sensitive and all of its built-in keywords must be entered in lower case. The first line in the program is a comment line. The double slash characters mark the start of a comment and all text after the // up to the end of the line is ignored. C also supports comments that span multiple lines. All text between /* and */ markers will be ignored.

Skip past the #include line for now, we will come back to this in a moment.

C programs are broken down into separate named blocks of code called 'functions'. Each function can take any number of parameters and can optionally return a single value. The next line in the program creates a function called 'main'. The word 'void' before the function name declares the type of the return value, in this case, void because we are not going to return a value from the function. The parameters of a function are listed after its name inside brackets. Again, 'void' here means the function takes no parameters.

Typically, many programs follow the convention of using lower case for function names, often with multiple words separated by underscores. For example, 'move', 'fire' or 'show_high_scores'. Sometimes you may see programs or libraries that use mixed case function names such as 'NewList' and 'AddListitem'. These are not enforced by the compiler, they are just conventions used to make names consistent and readable.

The function called 'main' serves a special purpose in C. It's automatically called when the program is run and therefore marks the starting point of the program. When all the instructions in the main function have completed, the program will exit and return back to BASIC. Every C program must have a 'main' function.

The body of a function is contained within curly braces. This is the way C groups together lines of code into discrete blocks. Our 'main' function only contains one instruction that calls another function named 'puts'. The brackets contain the arguments passed to that function, in this example the string "It works!" delimited by double quotes. The 'puts' function is one of many standard functions defined by the C language and is used to print out a string. Notice that the instruction is followed by a semi-colon. This must be used to mark the end of every separate instruction in C.

Before a function can be called, C needs to have been previously told about its name, what parameters it takes and what type of value it returns. The C compiler uses this information to check that the arguments you pass to the function match up with the parameters it expects. C is quite strict in this regard. For example, it would not allow you to pass a number to the 'puts' function which expects a string. So, how does the compiler already know about 'puts' in our example program?

The answer lies with the #include line at the start of the program. This instruction reads in another file containing declarations of functions that are in the standard library, one of which is 'puts'. These files are called 'header' files and have the extension '.h'. The one used here is 'stdio.h', short for 'standard input and output'. It contains functions used to read and write data to files and other devices including the screen.

The core C language itself is very simple, consisting only of a few keywords for declaring functions and variables, testing conditions, looping and arithmetic operations. The rest of the features are provided by libraries of functions. The standard C language provides a good number of library functions for many purposes including string, file and memory handling. The cc65 compiler provides some additional libraries and you can also create your own libraries of often-used functions.

Variables
Now let's try using variables in C. Close the 'first.c' file and create a new file called 'var.c' at the command prompt:

```
  notepad var.c
```

Enter the following program into this file:

```
#include <stdio.h>

void main(void)
{
  int n = 1000;
  char c = 'X';
  char s[] = "Commodore";
  int a[5] = { 1, 10, 100, 1000, 10000 };
  int i;

  printf("n = %d\n", n);
  printf("c = %c\n", c);
  printf("s = %s\n", s);
  printf("a =\n");

  for (i = 0; i < 5; ++i)
  {
     printf("%5d\n", a[i]);
  }
}
```

Save the file and compile it at the command prompt using the cl65 command as before:

```
  cl65 var.c
```

When the program has compiled correctly, switch back to VICE. You could run the program using the same auto-load method as before but, because VICE has now set up device 8 to point to our tutorial directory during the last auto-load, you can also enter the familiar load command directly into the C64 emulation:

```
  LOAD "VAR",8
  RUN
```

The program should display the values of the variables. 'n' is a number variable, 'c' is a character, 's' is a string and 'a' is an array of numbers. These short variable names have been chosen so the code fits in the narrow columns of the magazine. Normally, variables would have longer, more descriptive names!

Variables declared inside a function must be positioned at the start of the code block, that is, after the opening curly brace before any other instructions. These variables will be 'local' to the function. In other words, they are only accessible by that function and only exist temporarily while the function is running. You may also create global variables in C by declaring them outside of any function. We will see an example of global variables later.

Variable declarations follow this general format:

```
  type name = value;
```

All variables in C must be declared to be a specific type. The type dictates what kind of data can be stored, the limit on its values and how much memory the variable takes up. The most commonly used types in cc65 are:

```
  char - characters or whole numbers from -128 to 127
  int - whole numbers from -32768 to 32767
  long - whole numbers up to +/- 2,147,483,647
```

These are 'signed' types because they allow both positive and negative numbers. C also supports 'unsigned' types that allow larger positive-only values:

```
  unsigned char - numbers from 0 to 255
  unsigned int - numbers from 0 to 65535
  unsigned long - numbers up to 4,294,967,295
```

The variable name can be any combination of letters, numbers and underscores (although it can't start with a number). Conventionally, variables have lower case names, often with words separated by underscores. For example, 'level', 'score', 'high_score' etc.

A variable doesn't have to be assigned an initial value. Any variable without an explicit value will contain random garbage. To avoid errors in your programs it's usually a good idea to give variables an explicit initial value.

Arrays
C has no built-in type for strings, although it does allow a literal string to be created by putting it in double quotes. In C, a string is simply considered to be an array of characters.

Arrays are declared by putting the size of the array in square brackets after the variable name. In the example program, the variable 's' is declared with empty brackets so the array will be as long as it needs to be to fit the string. The variable 'a' is declared as length 5 and has 5 values assigned to it. Notice the use of curly braces again for grouping these values together.

To access an element in an array, the index number of that element is put in square brackets after the variable name. Index numbers always start at zero and go up to the array length - 1. In the example program, we could access the first element of the array with a[0] and the last element with a[4]. Often, we would use some kind of loop to iterate through all the elements in an array and use a variable as the index.

Formatted outputWe have also introduced a new library function here, printf. The 'f' stands for 'formatted' and this function gives us greater control over the layout of the printed text. The first parameter is a 'format string'. This is the text we want to print with placeholders (marked by a % sign) for the values we want to display. These values are passed as extra parameters to the printf function. The first printf instruction displays the value of the 'n' variable. We use a %d placeholder to print the value of this variable as a decimal number. The second printf displays the value of the 'c' variable and uses the %c placeholder to display it as a character. The third printf uses the %s placeholder to display the contents of the 's' variable as a string.

There is no special format placeholder for an entire array so we use a loop to print the values one at a time. Notice this time we use %5d in the format. This causes the numbers to be printed in a 5 character wide field. Any numbers shorter than 5 digits will be padded with spaces on the left hand side to make them line up.

There are many other formatting options provided by the printf function. I will leave it as an exercise for the reader to explore them in more detail. One important thing to remember though is that 'printf' is a large, complex function and will add a considerable amount to the size of your program. If you need to keep the size of your program down you may prefer to use other library routines for generating output.

Finally, notice the '\n' at the end of each format string. This stands for 'new line' and moves the cursor down to the start of the next line.

The for loop
The previous example also introduces C's for loop. The syntax looks strange at first but it's not too complicated. The basic form is:

```
  for (initialisation; condition; increment)
```

There are three sections in the brackets, separated by semi-colons. The first section initialises the values of any variables needed during the loop, usually the loop counter. In our example, we set the value of 'i', the loop index, to 0. The next section is the condition. The loop will run as long as this condition is true. Note that the condition is tested at the start of the loop so it's possible for the body of the loop to never run at all if the condition is false the first time round. In our example, the condition is 'i < 5'. The loop will keep running while 'i' is less than 5 and will terminate once 'i' is 5 or more. The final section of a for loop is used to increment the loop variable. Here we have used '++i' to increment (add 1 to) the value of 'i'. The for loop here is therefore roughly equivalent to the following BASIC loop:

```
  FOR I=0 TO 4:PRINT A(I):NEXT I
```

The console library
While the functions provided by the 'stdio' library are very useful, they are general purpose routines designed for writing and reading streams of data to and from files. The screen just happens to be one such type of data stream. The cc65 compiler provides another library called 'conio' that is designed specifically for printing to the screen and reading from the keyboard (collectively known as the 'console'). These routines allow extra features such as moving the cursor to a specified location on the screen, changing colours, waiting for a key press etc. They are also usually smaller and faster than the equivalent 'stdio' routines. Let's write a program to demonstrate some of this library's features.

Close the previous editor window and create a new file called 'console.c'. Enter the following program into this file:

```
#include <conio.h>

void colors(int paper, int ink)
{
  bgcolor(paper);
  bordercolor(ink);
  textcolor(ink);
  clrscr();
}

void show_title_screen(void)
{
  colors(COLOR_GRAY3, COLOR_GRAY1);

  cputsxy(14, 12, "Welcome to C");

  cgetc();
  colors(COLOR_BLUE, COLOR_LIGHTBLUE);
}

void main(void)
{
  show_title_screen();
}
```

Compile the program with 'cl65 console.c' and run it in VICE. The program will change the screen colours to grey and display a message at the centre of the screen. It then waits for a key to be pressed before restoring the default colour scheme.

This program introduces a few new concepts. Let's go through it. The #include line reads the conio header file so we can call the functions in that library.

Next, we have created a function called 'colors' to change the colours on the screen (I used the American spelling so the name is consistent with the names of the library functions). The function takes two colour parameters and uses these to set the background, border and text colours. It also clears the screen. Notice how the parameters, just like variables, must be declared as a specific type.

The second function, 'show_title_screen', calls our 'colors' function to change the screen to grey. We can use named constants for the colours rather than having to use literal numbers which makes the program more readable. The convention for naming constants is to use all upper case letters with words separated by underscores. This makes it easy to differentiate them from variables.

The function 'cputsxy' is used to print the message at the centre of the screen. There are many such functions in the conio library. Have a look through the 'conio.h' file (in c:\cc65\include or /usr/local/lib/cc65/include) to see all the functions this library provides.

The 'cgetc' function waits for a key to be pressed. It actually returns the ASCII code of that key but we are ignoring the return value in this example. Once a key has been pressed, the colours are restored to the familiar blue on blue.

Notice how the program has been organised. Each function performs a small well-defined task, and the 'main' function does little more than call our other functions. It's good practice to build programs up like this, rather than just dumping everything into the 'main' function.

Also notice that the functions are defined before the point where they are first used.

Remember that C needs to know what parameters a function expects before it can call it. One way to ensure this in your own program is to place the function definition before any code that calls it. C also provides another, more flexible, way to do this which we will look at next time.

It would be nice if our program could calculate the co-ordinates and centre the text for us. Let's write a function to do this. First, we need to know the dimensions of the screen. Okay, we know that the C64 screen is 25 lines of 40 columns, but if this program were run on a C128 in 80 column mode that assumption wouldn't work. Let's do it the proper way by asking the conio library for the dimensions of the screen. First, add this line at the top of the file under

#include <conio.h>:

```
#include <string.h>
```

We need to call a function that is in the string library so this line reads the header file for that library. Now add the following lines above the 'colors' function:

```
typedef unsigned char byte;

byte scr_width, scr_height;

void init(void)
{
  screensize(&scr_width, &scr_height);
}

void center(char text[], byte y)
{
  cputsxy((scr_width - strlen(text)) / 2,
          y, text);
}
```

Here we have introduced a number of new things. First is the 'typedef' line. We are going to be using a lot of variables of type 'unsigned char' which is quite cumbersome to type (and read). We can use 'typedef' to make a new type name that is equivalent to some other type. Here, we are telling the compiler to allow us to use the word 'byte' to mean 'unsigned char' which makes our program much neater.

Now, using our new 'byte' type, we create two global variables to hold the width and height of the screen. The variables need to be global so that the other functions can use them.

Next, we create a function that initialises these variables. using the 'screensize' function in the conio library. The significance of the '&' will be explained in more detail later when we look at pointers. For now, suffice it to say that the ampersands allow the 'screensize' function to modify the variables passed as parameters. Remember that a C function may only return a single value. This 'modifying the parameters' approach is one way to get around that limitation.

The last new function is 'center' which takes a text string and a line number as parameters. The function then uses 'cputsxy' to print the text as before, but this time calculates the x co-ordinate using the length of the string (obtained using the 'strlen' library function) and the screen width.

Now we need to change the line that prints the message to use our new center function:

```
  center("Welcome to C", scr_height / 2);
```

And finally, change 'main' to call the 'init' function at the start of the program so the screen size is set up ready for the other functions to use:

```
void main(void)
{
  init();
  show_title_screen();
}
```

Save these changes, re-compile and run the new program. The result should be the same as before, but now we have a general purpose function that can centre text on any line of the screen.

The conio library also supports a limited form of line drawing using the Commodore's graphical characters. Let's use these to put a box around the edge of the screen. Add the following function above the 'colors' function:

```
void box(byte x, byte y, byte w, byte h)
{
  cputcxy(x, y, CH_ULCORNER);
  chline(w-2);
  cputc(CH_URCORNER);
  cvlinexy(x, y+1, h-2);
  cputc(CH_LLCORNER);
  chline(w-2);
  cputc(CH_LRCORNER);
  cvlinexy(x+w-1, y+1, h-2);
}
```

This function takes as parameters the co-ordinates of the top left corner of the box, its width and its height. It uses a combination of 'cputc' to place individual corner characters on the screen and 'chline' and 'cvline' for drawing horizontal and vertical lines respectively.

Now change the start of the 'show_title_screen' function to use our new box drawing capability:

```
  colors(COLOR_GRAY3, COLOR_GRAY1);
  box(0, 0, scr_width, scr_height);
  box(9, 10, scr_width-18, scr_height-20);
```

Re-compile and run the console program again. Now the screen and message have a box drawn around them. Try experimenting with these functions to create your own title screen. How about changing the colours or having multiple lines of centred text?

Strings

In the previous program we touched briefly upon the string handling library. Let's now look at strings in more detail. As mentioned earlier, a string is an array of characters. We can create a string in C by enclosing it in double quotes and access the characters in it through the array. For example, say we created a string like this:

```
  char s[] = "String";
```

We can access the individual characters in the array through the 's' variable. The expression s[0] would reference the first element in the array, the letter 'S', the
expression s[1] would reference the 't' and so on.

How do we know when we have reached the end of the string? C doesn't store the length of a string, it uses a terminating 'null' character to mark the end of it instead.

Every function that manipulates strings needs to check for this 'null' character (ASCII code 0).

C provides a library of string functions which we can access by including the 'string.h' file in our programs. You have already seen one such function, called 'strlen' which calculates the length of a string by counting the characters until it reaches the terminating 'null' character.

One slightly awkward feature of the C language is that the normal assignment and comparison operators don't work with strings. The expressions 'str1 = str2' or 'str1 < str2', for example, don't work the way you might expect. Instead, library functions must be used to copy and compare strings.

Let's create another program to demonstrate some of the standard string library functions that you will use most often. Create a new file called 'string.c' and enter the following code into the file:

```
#include <stdio.h>
#include <string.h>

void main(void)
{
  char str[] = "Commodore";
  char copy[20];

  printf("str: %s\n", str);
  printf("len: %d\n", strlen(str));

  strcpy(copy, str);
  printf("\ncopy: %s\n", copy);
  printf("len: %d\n", strlen(copy));
  printf("cmp: %d\n", strcmp(str, copy));
  if (strcmp(str, copy) == 0)
  {
    puts("equal");
  }
  else
  {
    puts("not equal");
  }

  strcat(copy, " 64");
  printf("\ncopy: %s\n", copy);
  printf("len: %d\n", strlen(copy));
  printf("cmp: %d\n", strcmp(str, copy));
  if (strcmp(str, copy) == 0)
  {
    puts("equal");
  }
  else
  {
    puts("not equal");
  }
}
```

As usual, save the file, compile it using 'cl65 string.c' and run it in VICE. The program creates a string and a second array that is used to store a copy of that string. The 'strlen' function is used to return the length the string. The 'strcpy' function copies the original string into the 'copy' array.

The 'strcmp' function is used to compare two strings. It returns 0 if the strings are equal, a number less than 0 if the first string is less than the second, or a number greater than 0 if the first string is greater than the second. The 'if' statement compares the two strings and displays 'equal' or 'not equal' depending on the result. Notice the use of the '==' operator to check for equality. In C, the '=' operator is only used for assigning values to variables.

Finally, the program uses the 'strcat' function to concatenate extra text to the end of the copy string. See the difference this makes to the output of the 'strlen' and 'strcmp' values.

POKE and PEEK

The cc65 compiler provides a neat way of performing POKE and PEEK operations. Use #include <peekpoke.h> to get these features:

```
  POKE(address, value);
  PEEK(address);
```

Now, you may be thinking that we can use these for controlling the graphics and sound capabilities of the C64. Indeed we could, but cc65 provides a better way to help us program the custom chips by providing a named set of variables for their registers. These features are found in the header file called 'c64.h'.

As an example, while we could change the border colour using a POKE such as this:

```
  POKE(53280U, COLOR_BLACK);
```

A better way would be to use the following instruction:

```
  VIC.bordercolor = COLOR_BLACK;
```

This is more readable and, perhaps surprisingly, is just as efficient as the equivalent POKE.

VIC registers
To demonstrate some of these features, here's a short program that displays a sprite on the screen. Create a new file called 'sprite.c' and enter the following program into it:

```
#include <string.h>
#include <peekpoke.h>
#include <c64.h>

#define SPRITE0_DATA 0x0340
#define SPRITE0_PTR 0x07f8

void main(void)
{
  memset((void*)SPRITE0_DATA, 0xff, 64);
  POKE(SPRITE0_PTR, SPRITE0_DATA / 64);
  VIC.spr0_color = COLOR_WHITE;
  VIC.spr_hi_x = 0;
  VIC.spr0_x = 100;
  VIC.spr0_y = 100;
  VIC.spr_ena = 1;
}
```

After including all the required header files, this program uses #define to create two constants that are used to tell the VIC chip where the sprite is stored in memory. The '0x' prefix on the numbers tells C they are in hexadecimal.

The main function uses 'memset' to fill the sprite data memory with the value 0xff (a byte with all bits turned on). This results in a solid block image. Next, the pointer for sprite 0 is set to point to the sprite data using a POKE. Then we use the pre-defined variables to set the VIC registers for sprite colour and position and finally, make it visible.

Try experimenting a little with this program. How about using a 'for' loop to move the sprite around the screen? If you're feeling adventurous, how about writing a function that takes the x and y co-ordinates as parameters and moves the sprite to that position. The function would have this type signature:

```
  void move_sprite(int x, int y)
```

The tricky bit is dealing with x co-ordinates of 256 and above. If x is less than 256, 'VIC.spr_hi_x' should be 0 and 'VIC.spr0_x' should be x. But when x is 256 or more, 'VIC.spr_hi_x' should be 1 and 'VIC.spr0_x' should be 'x - 256'.

Next time
In the next article we will continue with the sprite theme and see how to build your own library of sprite handling functions. We will also look at how to integrate assembly language into your C programs for extra speed. See you then.

Article text and resources are now available at
http://sites.google.com/site/develocity/commodore/articles