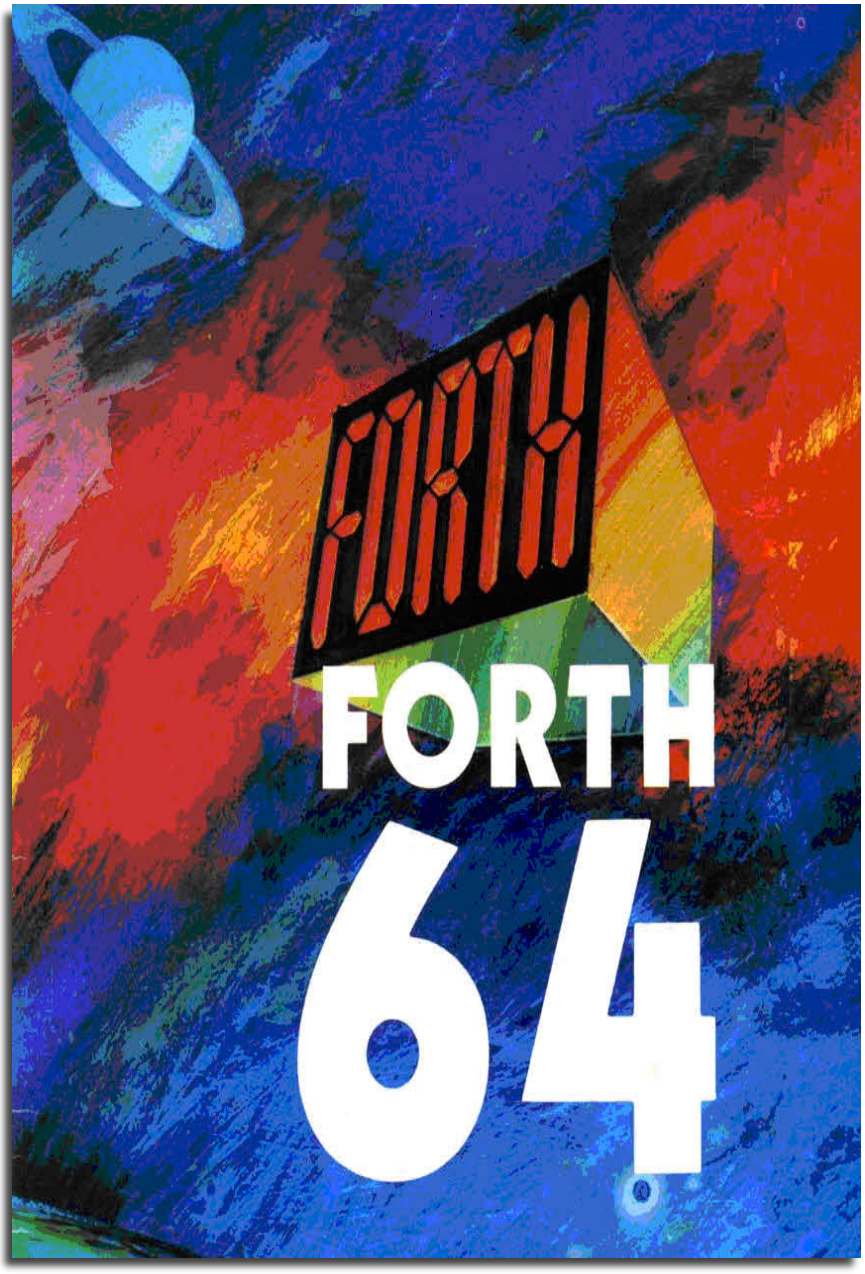
Commodore Free

Issue 28 March 2009

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



CONTENTS

How you can help

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

EDITORIAL

First a big thanks to all the people who took time to email me giving me positive comments and encouragement, I must say I didn`t expect such a number of emails of praise to arrive in my inbox! I did receive a few negatives but they criticized certain layouts of the magazine and interviews however these users did make suggestions on ways to improve so I cant call them complaints really as they were helpful in producing a better more polished magazine.

This issue is rather Forth orientated, I was contacted by a reader with an offer of a tutorial about the programming language Forth, its quite a long tutorial and the reader suggested it not be split, but would prefer the tutorial as is so I hope you enjoy reading about the Forth language, I also contacted a Forth user group and asked some questions you may like to read the answers. I have never come across the language before; it makes some fascinating reading, if you want to learn more then feel free to contact me.

BASIC to MACHINE CODE
Ok I am trying to write a tutorial a game that starts using Commodore graphics and BASIC then extends into using sprites then adding music and finally finishes with a game in machine code, this will run over a number of issues, problem is, I am definitely not a programmer and so am requesting help to write this tutorial, although there are a number of machine code type tutorials there is little taking the BASIC programmer with a minimal amount of knowledge into the real's of assembler or even just BASIC game design and its pitfalls. If you can help out in anyway I would appreciate your input, as a reward I can print your name in the magazine, heck that must be worth something.

BYTE BACK
As I write this Byte back is imminent and it is sadly impossible for me to attend due again to the back problems, you could say the I am "BACK BYTE" Argh that`s a very poor joke, thanks to everyone who expressed concerns and although I don`t feel it I am assured the back is getting better and will "fully recover", or is that just because I pay private and they want my custom to continue, I suppose if they said "you wont recover" then I would stop paying for treatment.

CP/M
I am making a final push for you guys to supply some information about Cp/m, I have a little something to put together for a future issue; but I am a little disappointed that no one has come forward saying they used, or indeed still use CP/M with there Commodore machines. Although many people are still posting messages about CP/M and Commodore on various news groups and forums. I guess these users are a little shy so if you feel that way I can publish your items anonymously if you so wish.

Commodore Haters
Its sad that I found out some people are attacking Commodore users via email and forums criticizing everything they do or post, Commodore Free magazine tries to cater for all users as I say every issue from Beginners who have just acquired a machine through to advanced users. I know the main focus is on the Commodore 64, but this is a special machine for me as it holds so many memories. If you have any Commodore related news feel free to contact me. Back to the haters then, we are building up a list of user names and I have successfully traced some users back to ISPs (at cost to myself) should I publish the information? Well I am undecided at yet

Ok I am trying to write a tutorial a game that starts using Commodore graphics and BASIC then extends into using sprites then adding music and finally finishes with a game in machine code

Regards
Nigel
www.commodorefree.com
www.commodorecomputerclub.co.uk

Editor	Nigel Parker
Spellchecking	Peter Badders
Disk image Build	Al Jackson
Submissions	Lord Ronin (in the Beginning)
	Paul Davis (beginning forth tutorial)
	Members Silicon Valley Forth Interest Group (interview) (http://www.forth.org/)
	Forth Inc (http://www.forth.com)

COMMODORE FREE DISK IMAGE REVAMP

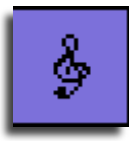
NOTE: Issue 26 marked the beginning of a new format for the C64 version of Commodore Free:
Use a Mouse in Port #1, a Joystick in Port #2, or Keyboard Cursor keys to navigate the new format.

On the Main Menu screen, you will find a list of Topics on the left, 3 Icons on the right and a Title bar on the bottom.
As you move the 'Arrow' cursor over the Topics/Subtopics or Icons, the Title bar provides a short description of what you can expect when you select that item.

The top left Icon is a question mark.

If selected, you can read about Commodore Free.

The top right Icon is a music staff If selected, you can play or change background music.



The bottom Icon is a curled arrow. Select this to quit Commodore Free & return to BASIC.

When you select a Topic, you will either get the article or a list of Subtopics to pick from. The article screen displays an article title at the top, a scrollbar (with single & double arrows) on the right and the words EXIT & PRINT on the bottom. If you click on a single arrow, the text will scroll one line at a time in the direction of the arrow. Use the double arrow to scroll a page at a time. On the keyboard, the UP/DOWN cursor key scrolls a single line & the RIGHT/LEFT cursor key scrolls a page at a time.

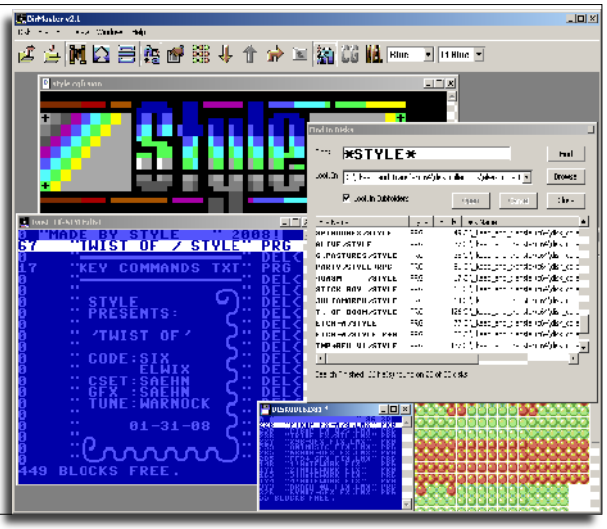
Select EXIT to return to the Main Menu or PRINT to get a Hardcopy of the article in 80 columns. The keyboard 'E' or 'P' keys also provide these functions. On the main screen pressing M will toggle the music on or off, but has no effect in the reader application.

THANKS go to
Al Jackson - Testing
Nigel Parker - Ideas testing
Alan Read - ALL programming

The Whole application was written using LOADSTAR`s BASIC DOTNET +

NEWS

DirMaster v2.1/Style



DirMaster is a Windows-based GUI tool useful for the management of common (and uncommon) emulator formats (such as .d64, .d81, .t64, .dfi, .g64, etc) as well as native archival formats (such as .arc, .sda, .lnx, etc).

New in this version:
- t64 *write/modify* support
- t64 "Run In" support
- disk/archive content preview on file open dialog
- remembers window location/size between launches
- cross linked sector check

"Run In" can now operate from a transient disk or tape image; this creates a temporary image which is passed to the chosen emulator; temp images are deleted when DirMaster exits

Configurable character replacements for exported file names and various other user interface tweaks and bug fixes

Jim Brain uIEC/SD Device

From: Jim Brain
I held off on announcing uIEC/SD availability until I had some stock (impatient folks, you know who you are but I do now have some stock (90 units).

Since uIEC shares the same firmware (kudos to Ingo Korb, who does not get enough recognition for this fine piece of code) as the recently announced SD2IEC, I'll spare everyone rehashing the similarities and just note the differences:

- uIEC/SD is currently the smallest known CBM drive (1.5" x 1.5" by 0.3"). Perfect for embedding in your favourite machine, drive, or calculator (shout out to Tone007, who stuffed one in a CBM pocket calculator)

- uIEC/SD shares the same 128kB Atmel AVR 8-bit microcontroller as the rest of the uIEC line. With 51kB used for firmware, there's plenty of room left for the future.

- uIEC/SD comes complete for use with IEC connector and power supply cassette port connector. VIC/64/C128 users can simply plug the wires in and use. (SX64/+4/C116/C16 users need to source 5V elsewhere, let me know before purchasing if you'd like an alternate connector)

- Although not yet defined for use, uIEC/SD offers an additional switch line and programmatic LED for future use.

- uIEC/SD not only supports SD and SDHC cards, but either SD or SDHC cards can also be used for updating the firmware (new feature, older firmware update software only supports SD cards)

Pictures available at:
http://www.jbrain.com/vicug/gallery/uIEC/IMG_6514
As with uIEC/CF, units are $50.00/unit, with shipping (for up to 3 units) as follows:

$5.00/US
$10.00/Canada
$15.00/Intl

Email brain@jbrain.com for ordering information, or simply PayPal the address for immediate purchase.
I'd like to thank all those who have purchased uIEC units. I hope everyone finds them useful.
Jim

SCACOM Aktuell issue 10

SCACOM Aktuell issue 10 (the German Commodore magazine) is available for download in a number of formats ZIP, PDF and JPG. Grab a copy from here http://www.scacom.de.vu/

Vote for me

I notice I have an entry in csdb but I only have one vote though, I , head over to the sight and make your vote heard http://noname.c64.org/csdb/scener/?id=10226

3 New Vic 20 games arrive

Kweepa has released "WhackE". Go to http://www.kweepa.com/step/vic20/whack-e/whack-e.prg

Jeff Daniels has released "Ten Ten Duality". As mentioned earlier in the news section Click the link at http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=3362&...

Boray has released "How Many". Download at http://user.tninet.se/~pug510w/datormuseum/howmany.zip

The SEUCK Vault

Six more games enter the Vault.
- Piequest and Lost Treasure (both Amiga) by Ricky Derocher
- Legion of the Damned by Anthony Burns
- Renovator and R-Type Reverse by Simon Peterson
- Imaginator v3 by Richard Bayliss

Watch out soon for a re-designed archive and more games to download. http://www.seuckvault.co.uk

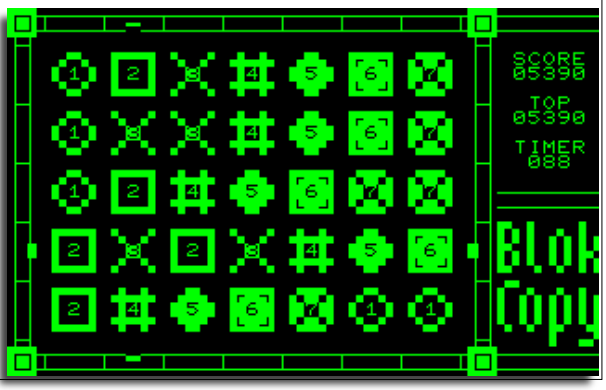
Blok copy for the PET

Jason Kelk (TMR) has released Blok Copy, for the Commodore PET, you will need a minimum of a 40-column screen and 8K of RAM to run the game. The game also supports sound from the parallel port of the PET. To read about it and download a copy head over to

<http://www.cosine.org.uk/products.php?prod=blok_copy&4mat=other>

The mission objective for Blok Copy is simple; the playfield contains seven distinct designs of tile arranged into columns of five and, at the start of each level, those tiles are shuffled around; the operator must then reorganise those tiles to resemble their starting order to unlock the stage and progress to the next until all ten levels have been re-synchronised. The controls are W for up, X for down, A for left and D for right, using these controls alone move the cursors at the edge of the playfield to select a row and column and holding down the right hand shift key at the same time will instead cause the selected row or column to move.

Blok Copy is the first release for Cosine on the Commodore PET. In order to run it, you'll need a machine with a 40 column display and a minimum of 8K RAM, any PET that meets these specifications should be able to run it and those fitted with a parallel port sound hack will also get titles music, in-game effects and jingles (the sound is enabled by default in VICE but is very loud, turn down the volume before running the game!) If you'd like a version of this game on tape or disk it should be available soon from Cronosoft for £1.99 (excluding postage and packing).
Trivia: the original and, at the time of writing, incomplete version of this game was written in order to learn the Atari 2600 hardware where it utilised vertical splits to achieve eight independently coloured playfield objects; despite all of that colour, the PET version is, of course, rendered entirely as green (or orange) on black. The music is a cover of "End Theme" done on a one channel sound hack...!



New revolutionary C64 music routine unveiled

Commodore Free :- Although this demo was original released on the 29th of October in CSDb http://noname.c64.org/csdb/release/?id=72678 it's one I missed! Al Jackson noticed the article on this blog website http://c64music.blogspot.com/ and I have copied the text below. Now for the people who still doubt; how about that C64 entering the charts with minimal external processing effects is getting closer, wouldn't it be great to see errr hear on a cd :====

A team of super talented coders have managed to break the limits of C64 sound once again.

The Human Coding Machine from Germany and SounDemoN from Finland, have managed to create a music routine that allows you to have:

- 4 channels of 8-bit sample rate, digi playback
- 2 channels of SID synth sound
- You can filter both SID channels AND SAMPLES!
- And you have enough raster time so you are not forced to turn off the screen, and can actually do something with it ;)

This technique has been presented in the X'2008 demo party, the bi-annual classic of the C64 scene where usually all barriers are broken and this one has been, fortunately, no exception!

The routine in question was demo'ed in the production VICIOUS SID (picture), which also included other crazy audio routines you can check out by following the attached link.

Beta testing of this software will start soon, a few artists have started to experiment with pre-beta versions of the tools, like Fanta, who made an amazing X'2008 SID music entry with his tune Fanta in Space, which you can check out by clicking here.
http://oxyron-party.untergrund.net/fanta_in_space.mp3
(this is the output straight from a Commodore 64, nothing added!)

http://c64music.blogspot.com/

Ten Ten Duality

http://sleepingelephant.com/ipw-web/bulletin/bb/viewtopic.php?t=3362

A game for the Unexpanded Vic 20, Joystick required.
All BASIC.

Sleep deprivation does strange things to you. I made this game earlier this week between 1am and 5am. I couldn't sleep. It made sense to me at the time and seemed very fun. The next day, I played it, and I couldn't understand what the heck I was thinking the previous night! I decided to just abandon it.

Tonight I am up late again, and the game makes sense to me again. So, at the moment, my judgment may be clouded. I present to you, Ten Ten. Whatever that means.

The Story:
In a world of bi-location, you must achieve synchronization. "Ten-ten" comes from the Japanese "dot dot" or Dakuten. Your objective to align a universe of dots.

Game Play:
Use the joystick to control both "Tens" on separate playfields. Take advantage of asymmetry to bring each simultaneously to its respective goal. Fail, and your doppelganger will escape to autonomy.

Ok, I know none of that makes much sense, but play it anyway.

http://sleepingelephant.com/denial/games/TenTen.zip

THE MEGABIT 128 INTERNAL ROM ADAPTER
For the C128/C128D Internal ROM Socket

Developed by D. C. Newbury Email:=newbury(at)planetkc.com
Review by Mark R. Brown

THE BASICS The Megabit 128 internal ROM Adapter is a small circuit board that plugs into the internal expansion ROM socket on the Commodore 128 or C128D. Its purpose is to allow you to use high-capacity EPROMS: 27010 (1 meg.), 27020 (2 meg.), 27040 (4 meg.), or 27080 (8 meg.). It's very easy to install. Since the board is longer than a standard EPROM, you have to bend down a couple of small disk caps on the 128 motherboard before you snap it into the internal ROM socket. There's also a pigtail with a clip that has to be attached to a pin on the U3 chip. Piece of cake. No soldering. Easily removed if you need to.

Newbury currently sells two different versions of the adapter board. One addresses up to 16K at a time; the other addresses up to 32K at once. Each is the same price: $15 PPD in the US. Each uses the same EPROMs; which you choose comes down to how big a memory space you need to have active at any one time. The 32K version lets you include bigger apps, like KeyDos, the Servant, and BASIC 8.

THE BARGAIN An even better deal than just buying the bare board is to add five bucks and order either board with an EPROM pre-programmed with some apps and a nice menu program. This not only gives you a set of useful apps right out the gate, it means non-technical types can actually do something with this board.

Because if you want to do something of your own, you're on your own. While the developer is a very helpful guy, there's no easy way right now to get your own apps up and running on this board. Here's what he wrote back when I asked how to program my own EPROM apps for this board:

"I will have to put something together, like a separate manual that will have the commented ML listings for the auto start routine, menu and program loaders. It may take awhile."

Then he mentioned something about fishing.
Just remember that full support for developing your own menus and apps is planned. It's coming. Sometime. Also remember you're only paying $20 ([domestic] shipping included) for the pre-programmed edition of either version of this little marvel. If it were $50 or more, I'd linger here awhile, complaining. But for twenty bucks I think that's more than fair.

Okay. Fine. So we're stuck for the time being with the programs and menu system he's burnt for us. So let's talk about that. Because what's already included is a load and a half.

THE BYTES When I turn on my C128 in 40-column mode and press F1, nothing happens. At least, nothing happens on my C128. It might be interference with JiffyDOS, since I get the JiffyDOS '@$' command with F1. I don't know, but I'm not going to pull my JiffyDOS chip to find out. Maybe it's just an 80-column app. The manual doesn't make that clear. So let's try 80-column mode. Press F1 and... okay, THERE we go! A nice menu of apps.

What's available? Some of the best, most-used C128 and C64 utilities. Here's a list:

16K version w/4meg EPROM:

Merlin 128 v1.0	REU Test
Promos 2.0	My Disk Editor
Function keys	Viza Write 128
Viza Star 128	Seq Reader 128
Begin & End Adrs	Fastrac File copy
Directory Editor	Color 80 column
Basic Data Maker	Monitor 64
Basic Merge	Basic Merge +
Maverick File Copy	Maverick Track Editor
Single 41 Data copy	Dual 41 Data Copy
Single Nybbler	Dual Nybbler
Single 81 Data copy	Maverick File Tracer
Maverick Track & Sector Editor	64K VDC RAM Test

32k version w/1meg EPROM:

Key Dos	Servant
BASIC 8	My Disk Editor
Seq Reader 128	Color 80 Col
BASIC Merge	Function Keys
Begin & End Adrs	BASIC Data Maker
BASIC Merge +	

The 16k version is a melange of C128 and C64 apps, some commercial and some written by Newbury. All are very useful. I can't imagine there's much you'd want to do that isn't covered in here somewhere. But, just to be contrary, I do miss not having a disk cataloguer and a simple text editor like ZED. When you pick a program it comes up in a flash, of course. C64 programs kick into C64 mode and run flawlessly.

The programs have not been modified in any way that I can tell, other than to make them work from the internal ROM. That means when you're done using one of these programs, you have to power cycle your machine to quit. (Newbury's own programs do politely drop you back into BASIC when you exit them.) No problem, as long as you didn't expect this thing to work like Partner 128, letting you jump back and forth from applications to utilities like Bond from babes to bomb blasts. You can't have everything, dude. You DID remember to save your program before you hit F1, didn't you?

THE BITS What other goodies do you get? For twenty bucks? Are you kidding?

Newbury sent me a review package that was complete with a printed manual and a CD-ROM containing the documentation for all of the programs on the EPROM. I doubt you'll get all that for a couple of sawbucks. But I assume he'll be making it all available on the web for free. Installation instructions are complete and more than adequate to the (very simple) task. The manual includes documentation for all of the utility programs Newbury developed himself. The CD-ROM has PDF manuals and d64 files for all of the commercial apps included. As stated above, there is, as yet, no documentation on how to create your own EPROMs. But did I mention that it's coming?

THE BOTTOM LINE Best twenty bucks you ever spent. Ever. Pawn your grandpa's watch and buy this right now. Really. I mean it. Though I desperately want to create a menu and load it up with my own apps, for twenty bucks what Newbury provides is an excellent selection.

The board couldn't be easier to install. If you can install an internal ROM, you can install this board. This is a fine product and it solves a basic problem. At least it did for me.

I've already decided to keep the 32K version installed permanently. I had burnt EPROMs for KeyDos, the Servant, and BASIC 8, and have been struggling for a year over which of them to keep in that socket. Now I don't have to choose. Best of all, all three apps are 128-mode programs; I hate having to switch to C64 mode to do anything. Now I don't have to. [Dan Newbury, developer of the Megabit 128 Internal ROM Adapter, has more to say about it.]

I did not mention... that the printed manual is not included in the package.... It costs too much (ink, paper, cover and postage). But, if you want to pay extra for a printed manual, we could work something out. The adapter will come with a CD that has the manual in pdf format and all the other goodies.

I did find a bug in the adapters after I sent them... It`s a hardware problem, not in the software. Sometimes when you power up, the internal ROM will not be on page 0. The F1 function key will not get reprogrammed and you can`t get access to the menu. You have to power down and up to make it work. The new 32k board will correct the problem. I accept paypal or money orders.
The DO IT YOURSELF manual will take more time.

Dan…
newbury(at)planetkc.com
E-mail amendment from Dan Newbury

Hi, Nigel (Commodore FREE)
My name is Dan Newbury, creator of the Megabit Internal ROM Adapter. Mark Brown aka airship, has told me that you plan to publish Mark's review of the Internal ROM Adapter. If so, I would appreciate it if you would correct a wrong statement in the review. Mark stated that the price included PPD (postage paid). That is not true, below is the correct pricing:

Adapter without EPROM is $15.00 + shipping & handling USD.
Adapter with EPROM is $20.00 + shipping & handling USD.
I accept PayPal or money orders.

PayPal - USA & International
Money orders - USA only
Will ship anywhere USPS can go.

Thank you, Daniel C. Newbury

Simon Quernhorst Games Cartridge

http://www.quernhorst.de/atari/r8.html
http://www.quernhorst.de/atari/

I'm happy to announce the release of the limited edition of my 2nd C64 Cartridge called "R8ro!". The cartridge contains 8 minigames including the winners of the 1k and 2k minigame competitions 2008. 30 copies of this limited edition are made and they consist of cartridge, box, manual, postcards, serial number and signature.

I will update the website http://www.quernhorst.de/atari/r8.html today or tomorrow. Please visit it on the weekend for more information, pictures, prices and instructions how to order.

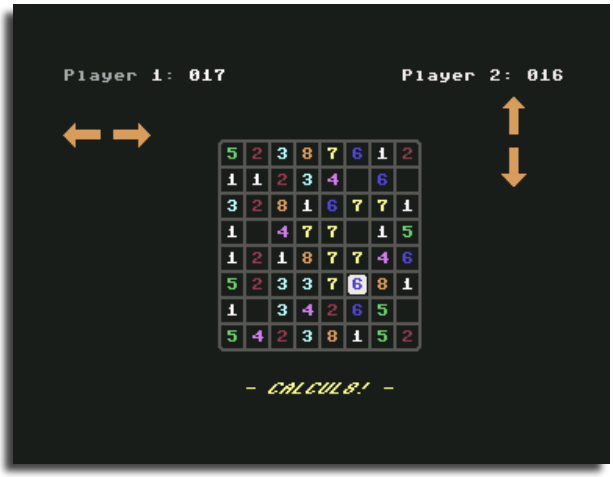
Best regards and have a nice weekend! Simon"

R8ro! (R-Eight-tro)
Eight new mini games for the Commodore 64.

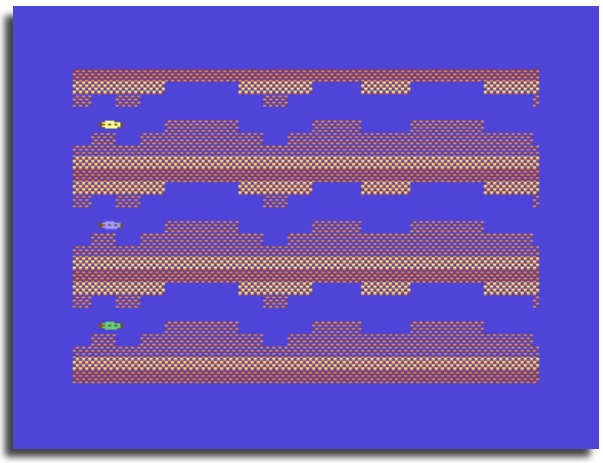
The story of these Games
An annual Mini Game Competition on the internet is held in the three categories of 1, 2 and 4 kilobytes. While getting to know the C64 better again (after years of coding for the Atari VCS and about seven years since my last C64 program) I had some ideas for possible games in the lower size categories. I thought that multiplayer games, scrolling multicolour landscapes, hectic puzzle games, showing sprites in the borders and other things would be possible in even the lowest amount of kilobytes. To have a goal to strive for, I plan to compile eight different mini games to a complete disk and cartridge called "R8ro!" which should give the owner a variety of different game concepts to choose from.

Elimin8! (Eliminate)
This is a game for one player. Nine cities around a radioactive plant have been polluted. The population was evacuated and the useless and dangerous buildings have to be exterminated now. You fly over the scrolling skylines and drop laser shots to neutralize columns of buildings. You can only move left and right while the gravity pulls you towards the buildings. You get more points if you manage to hit buildings without missing (combo scoring).

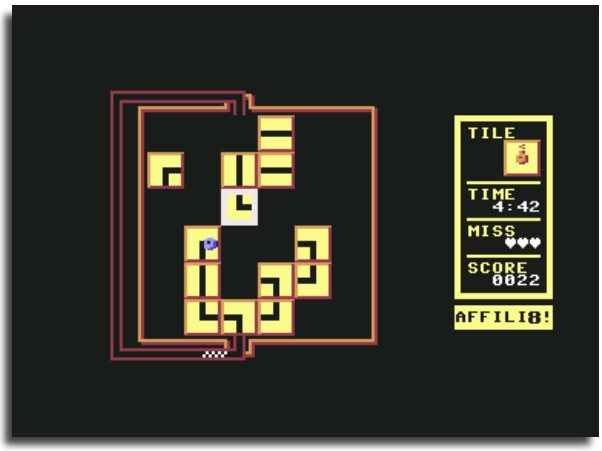
Calcul8! (Calculate)
This game for two players shows a playfield of 8 by 8 numbers. Both player alternating choose a field and try to get the highest points and leave less points for the opponent. Player 1 can only move his cursor horizontally while player 2 is only able to move vertically. So choose your fields wisely and take a look which fields your opponent may take afterwards.



Extric8! (Extricate)
This game for one, two or three players generates scrolling random tunnels which the players have to pass with their submarines. Two players control their sub with a joystick while the third player uses the keys F1 and F7. The distance is counted for every player. - so who is going to survive the longest distance?

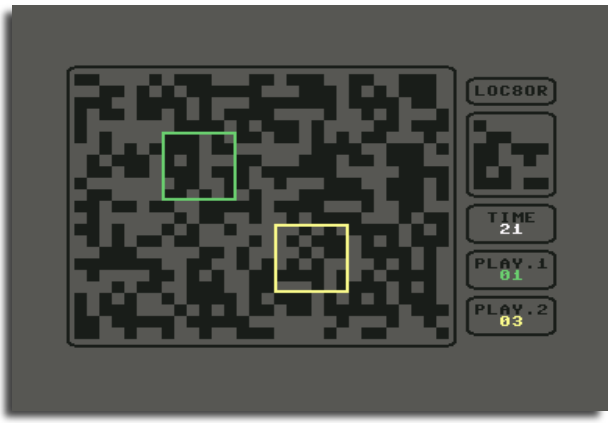


Affili8! (Affiliate)
This game for one player offers a playfield of 6x6 squares where you have to place randomly generated tiles. After a few seconds a marble starts rolling and it will follow the trail of your placed tiles. The marble removes passed tiles from the field and there are also bombs to remove misplaced tiles from the screen. The game is over if the marble crashes or if you didn't manage to place some tiles within the given countdown time. Try to prepare alternative routes as you never know which tile may appear next...



This 2kb game for one player offers horizontal scrolling in both directions with multicoloured graphics. The level design is about 30 screens wide.

Loc8or (Locator)

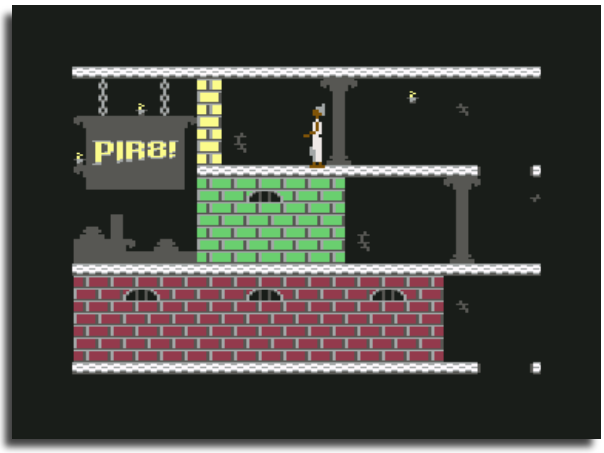


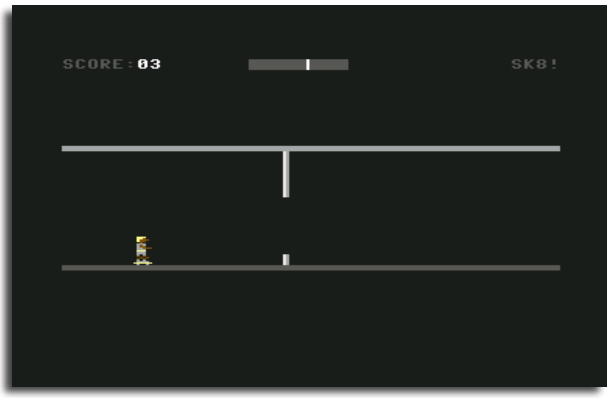
One or two player have to search for a small pattern of 6x6 squares in a randomized playfield. Finding a pattern increases your score and adds 10 seconds to the countdown timer. Try to find as many pattern as possible - and more than your opponent of course...

Sk8! (Skate)

Pir8! (Pirate)

Man, what a night! You hardly remember anything and try to find your way out of the dark dungeons of what you believe is the cellar of the bar where you spent the last evening. Jump and run along different paths and avoid the dead ends, falling too deep or falling into the pits...





Obstacles appear in your way and you have to jump to avoid them. This game for one player is only controlled by the fire button of your joystick. Holding fire raises the jump bar and releasing fire makes the player jump as high as the bar was raised before.

Elev8! (Elevate)

Technical Information

All games are written for a PAL Commodore 64. The total size of Elimin8!, Caclcul8!, Extric8! and Loc8or is 1 kilobyte (1024 bytes) each - containing all code, graphics and sounds. All code, graphics and sounds of Affili8! and Pir8! fit into 2 kilobytes (2048 bytes) each. All code, graphics and sounds of Sk8! and Elev8! fit into 0.5 kilobytes (512 bytes) each.

Production of original Disks and Cartridges

As soon as all eight games are finished, they will be compiled and released on disk and cartridge afterwards. It is unsure when these items will be ready, so please watch the web site for news. The price and amount of produced items is unsure right now.

BYTE BACK 2009

A report for Commodore Free
By Andrew Fisher

INTRODUCTION

Mat Corne went to the Fusion event in 2008, and vowed to put on his own retro gaming show. The end result was that over 200 people went to Bidds Live Music Club in Longton, Stoke-on-Trent to play games and listen to some fascinating talks on the weekend of the 7th and 8th of March 2009.

The venue was split into two rooms. In the main club were the bar and the food counter, with the rest of the room taken up with gaming stands and arcade cabs. Up on the stage were a couple of projectors, plus another at the other end of the room that was mainly used for Guitar Hero/Rock Band.

Among the stalls selling things were Console Passion and Keith Ainsworth from Retrogamer fanzine (who had a large number of Commodore tapes and some cartridges for sale). Commodore hardware was also present on a couple of the other stalls. In the main hall were a couple of Commodore 64's, one that was running various demos and games during the day, and the other that was mainly used for Shredz64 (the Guitar Hero clone). Steve (StarshipUK) had also bought along his C64G, which is used as a keyboard for his emulation PC.

In the "executive suite" was the homebrew showcase and the stage area where the Q&A sessions took place. The Attic Bug were here with many boxed and original games for sale - seeing a big pile of those classic Thalamus boxes, still shrink-wrapped, was impressive!

Jason Mackenzie was also there with his Psytronik/Binary Zone stall. The main machine on here was his hybrid "Specadore 64", a Commodore 64 repainted in black to resemble a 48K Spectrum. He also demonstrated The Last Amazon (soon to be released by Psytronik with a new third game to make it a trilogy) and the C64 DTV (the plug-in joystick). The Shredz64 set-up transferred here during the Sunday for a while before being packed up.

Also in the executive suite was the Commodore Computer Club UK. Among the machines on this stand were a Commodore 116 (a rare cousin of the C16/Plus4 that was loading games from Compact Flash) and a C64 running a SuperCPU and CMD drives. This C64 was actually an old "breadbox" machine but in a slimline C64C case. Metal Dust was on show, along with the patched version of Driller.

HAPPY TALKING

There were four different Q&A sessions on the Saturday, the first being Jamie Woodhouse (Amiga coder) and the last being Paul Drury interviewing arcade high score champions Jon Stoodley and Tony Temple. The other two talks had more relevance for C64 fans. "Ocean Reunited" saw four well-known people take to the stage. Spectrum programmer Joffa Smiff admitted beforehand he was very nervous, and did not take full part in the discussion. Graphic artists Simon Butler and Mark R. Jones (most known for his Spectrum work) were joined by programmer Jim Bagley who spent several years at Special FX working under contract to Ocean.

It was a very interesting talk, from tales of working in the "dungeon" with a Quaker graveyard under the car park, to the implications that certain games had high review scores "bought" for them. Simon Butler in particular had a lot to say about Total Recall, middle management and the lack of creative thought in the industry.

The other Q&A session on the Saturday took place at 5pm... and I went on to introduce "The Sensible Guide To Jon Hare". I have met Jon at several retro events, and jumped at the chance to talk to him live with an audience. I did have a list of pre-prepared questions, with a slideshow on the projector introducing each topic. But as the talk went on I got more into it, asking extra questions and debating key points with Jon. There were lots of interesting anecdotes, from the £5000 contract for Parallax to the X-rated scenes in the cancelled Sex & Drugs & Rock & Roll.

As well as the talks, there were various game competitions running over the two days and a charity auction on the Sunday afternoon. This included a reproduction poster of the Wizball artwork, signed by artist Bob Wakelin (who was there on the Saturday), designer Jon Hare and Spectrum graphic artist Mark R Jones.

SHOW TIME

This was a good weekend, although there were minor niggles. The venue was not that easy to get to, the food service was slow (I queued for an hour on the Saturday) and the state of the toilets was atrocious. Still, it was great to see so many old friends and play so many games. While organiser Mat is unsure about putting on another event, there is no doubt the weekend went well. And best of all, well over 1500 GBP was raised for charity - the RSPCA and a local hospice.

READERS CODE

AA TIMER

By JOHN FIELDEN

One of Commodore Free readers sent this application in it's for the commodore 16/plus 4 range of computers called AA timer, if you like these type in programs then let me know as the reader has some more, if space permits you will see the application on the disk Image (d64) of the magazine

AA Timer notes
lines:-10 - 110 REMarks about the program. Clear & Initialise Screen. Set max. Volume, people can then adjust their sets accordingly. Print instructions to test everything is working properly, Wait for user input.

120 - 170 Discern user response and act accordingly. Self explanatory really, from above PRINT statements.

180 - 260 Pressing any key in the menu above sends the user to this menu. As there is a sound test anyway, it isn't really necessary. It was a precursor to figuring out timings as I had to discover again from memory, due to the intro. to basic manual being long gone. I still seek this, and the sequel/s...I left it in as it is in line with the spirit of the program. May be useful for further study.

SERIOUS NOTE: Here I must make another point that it is dangerous to watch time. Not just because fellow workers consider it "skiving off work" but due to its effects on the mind. People have become seriously ill due to watching time. ...so I took up programming!

DEBUGGING: With the above in mind here is a tip to stop the computer showing the count.

LIST310 put the cursor over the 0. And type "1" and press RETURN repeat with "2", 3, 4, 5,6,7

LIST310-318 (I always add one in the list for good measure)

310 All you want is the first command. So delete "Inst/Del" or space out all the rest after. and press RETURN.

311 You already have the first command in the previous line, so delete it here. -And in all the following lines!) The first command now should read x$=ti$. scrub out the ones after it. Again you don't need the command after this line so delete all the others. The command gives the computer something to calculate with and updates according to the current time which got reset to midnight "000000" on the appropriate key press to start the test.

312 Here is nowhere we need to deviate from the pattern. Space/scrub out the whole line. And from the usual place type the following command and press RETURN.
IFX$=>3 ANDX$<7 THEN314

313 we only need the print statement here so delete the rest. By now there should on be the NEXT statement. And..

314 we only need the NextJ. You know what to do!

Testing the debugging: Here we have 3 extra lines. The computer must ignore these for the program to work. And as we're only testing our efforts thus far, we may need them later. Hold shift down on the P of the PRINT statement. And press the key for inst.del. 4 times. Let go of these. Type REM and press return key. On all the lines.

Run program.

The first thing I found was that I forgot to put inverted commas around the "3" and the "7". And I think if I live to be programming at 120 years old, I'll still be forgetting this!

The second thing I found was that the computer takes no notice unless it is written exactly to original form. That is "000003" ... "000007" because it's a string not just a number! Finally it now takes 12 seconds because of the extra command. It would be easier to change the printed statement from 10 to 12 than to mess about for next loop. Or would it? Try changing this to a DO Loop! In the first part of debugging we have found The proof of the pudding is in the eating, but in this case the eating proved the programmer to be the pudding! But no matter as this teaches us that you can't always get it right first time. So lets continue.

Q. Will it work using ti$ instead of x$?
The answer is yes, only we're up to 15 seconds on the count.

Now. LIST310-319 again change the following lines accordingly.

310 DO

317 LOOPUNTILX$=>10
Now we need to take out the NEXT STATEMENT IN 314. We will need to repeat line 313 with what I have in mind so go over to 313 and 314.

Now relist the required lines 310-319

312 ifx$=>"000003 and x$<"000007" then ja%=1;elseja%=0
Use the shift and inst. del keys to put the following in front of the existing lines 313 & 314

313 ifja%=0then...

314 if ja% =1 then ... (now delete everything after ";", in its place type "counting" and press return to store it.
I don't ever have any luck with do loops. I'm fully expecting the program to ignore my request to stop at 10, and proceed to infinity! There is a way out, firstly with run stop. So it is safe to see if this will happen. So run the program. Again the mismatch error in 317, corrected accordingly. It works, albeit requiring a slight modification.
ADD to the end of 313, the semicolon ";" symbol, followed by inverted commas. And 9 spaces, followed by 9 arrow presses to the left. This takes away the horrible " ing" on the screen at the end etc. Now the programmer is not so much the pudding! ie. Run the program to test it. It works! exit program and Save it. Small note:

line240 refers to the original for/next loop, not the new do/loop.

<**OPTIONAL>** LIST310-325 at the ready prompt type 315 & press return, same for 316. As these lines aren't needed. Had the UNTIL command been missed, as I have been known to get into a mess with these. I would've put in: 315 ifx$=>"000010"then320 The > symbol prevents any slight variation, more likely if using ti$ from causing an endless count.

<REMEMBER> WARNING: DO NOT PLAY WITH NOR WATCH TIME! It is only safe to glance on such rare occasions' as are needed.

360 - 550 when the user escapes the previous menu and gets to here. There are 3 options.
In the lines 500 to 530 these are reserved in case I ever want to add to the program. Lets look at the options. Print statements 420&430

go along with this but serve the dual use of putting a space between functions and exit key. This could easily be achieved in 440 with the down arrow.

390 & 470 The instruction to go to egg timer at line 1210 on pressing the corresponding key. The code also gives away the order in which the program was written. The original idea was to provide a system to count for any-one undertaking exercises like Pilates or Yoga or whatever

Mute Point: Much has been said in recent times about the annoyance of "Spaghetti coding", in this case the ability to go with how I felt was liberating. Giving a sense that I am doing this for fun, not out of some laborious obligation. And so what if no-one else can read it, they'll have to come to me!

Now though I'm in two minds. Do I follow the paper page for page or jump two pages to 1210 the way the choice is listed? ...It makes more sense to keep to the same page for this purpose.

550 tells the computer if you have any other key press than the options given. Ignore it and go back to wait for another one.

560 Interval beeps (key press 3)

570 - 640 Setting up sub menu, self explanatory.

650 Sets time to midnight, the only logical start point. oddly enough there is a DO without a LOOP. Can You see why 1. The computer never notices, I haven't once received a DO WITHOUT LOOP message! And 2. What would need changing to incorporate this as a full LOOP? (And don't say "get a better programmer" or I'll cry!)

660 look for a key press, but don't wait for one! The difference being the omission of the IFA$="" (NOTHING, No Key Press) Then Jump back to this line. The omission is quite correct for this purpose.

670 if random isn't chosen. or switched off there are specifics for the sound command at the end of the count. And so the computer must skip the following three lines that give random parameters. (experiment with this all you want). Of course if random is on then the reverse is true.

705 In testing I seemed to get all kinds of numbers -testable by adding a print statement at production of sound. ?"a";a ...etc.

710 makes sure Random is on or off, and so only the parameters 0 and 1 are used.

720 mute function, if phone rings to save stopping program. This will order the program to jump to 880 where VOLume on/off is decided, followed by the appropriate print. then returned at 900 to -

730 - 910 checks relevant parts to see if they need altering, carries them out and returns for more. The engine room so to speak. Can you find where the duration is set?, bear in mind that this would be at 0 therefore produce a constant and annoying beep, if not set ahead of the beep. 800&810 make it possible for the alarm to go off at the correct moment.

<DEBUGGING> 840 Why not goto650 and omit the extra TI$ statement FROM 840?Too many RETURNs, change the preceding returns of the sub-menu to go to the last RETURN of the type. In this case change870 to goto910 This way it will be easier to see where the menu of type ends. More useful in larger programs. Reduces "Spaghetti".

850 Appears to be a line left for debugging in building. As nothing I can see refers to it, and everything skips it. It can be deleted to avoid later confusion.

DO's and Dont's! If you wished to LOOP as mentioned earlier. Instead of Goto660, which if you've followed the above is now only820 as this does not reset time. In this case it may be better just to delete the DO as this menu isn't really designed for it.
Finally, don't confuse keypress "X" with variable x.

1000 The STOP WATCH menu.
A simple Start - Stop sub-menu. It's easy to press "S" to show time even with the line added to show when the counter is started. The rest seems self explanatory.

NOSTALGIA: I remember the "An Introduction to BASIC part 1" has a program with a timer that uses TI to test and measure reflexes (Do not confuse with the clock variable TI$), I've been longing to find a copy of this book and the sequel which I would've loved to own. Not least because it would enhance this and future programs in this field. In trying to recreate this from dead reckoning and "memory" (hilarious after 20 odd years!), I came up with the following which can stand alone or be added to the existing program. If you do add it. Also change lines

```
420 print" 4 - Reflex Timer"
            and
500  IFa$="4"then1540
            and
1720 GOTO370
```

```
1550 REM *REFLEX TIMER*
1560 VOL8
1570 PRINT"{clrhome}      REFLEX TIMER "
1580 PRINT"{down} WAIT FOR BEEP"
1590 X=INT(RND(100)*750):REM PRINTX
1600 FOR J=0TOX:GETA$
1610 IFA$<>""THENPRINT"NO! WAIT FOR BEEP
!":A$="":GOTO1590
1620 NEXT J
1630 SOUND1,300,10
1640 TI$="000000":REM PRINTTI
1650 GETA$
1660 REM PRINTTI
1670 IFA$=""THEN1650
1680 PRINTTI
1690 PRINT"ANOTHER GO Y/N?"
1700 GETA$:IFA$=""THEN1700
1710 IFA$<>"N"THEN1550
1720 END
```

A challenge would be to make it two player using specific keys, and saying who wins.
Also try adding a hall of fame. I managed a fluke of 6 once. Averaging about 19 or 20. occasionally being as slow as 27.

1590 randomises the wait for a beep. After REM is for debugging only.

1600 set length of wait according to above.

1610 check that nothing is pressed before beep. If it is say so, and reset and importantly change time to wait. This way person doesn't get used to delay.

1620-30 obvious

1640 This resets TImer! Remembered late in build.

1650-1680 Waits for key response and measures delay using TI. It's not seconds, and as I don't have the book I don't know the rules of the timer. A bit of fun nonetheless.

1690 - 1720 self explanatory.

1210 Egg Timer
Similar to stop watch, except set in advance how well you want your egg cooked. I can manage a full 9 minutes!
When time is up the program will beep until the user shows it due care and attentions by turning it off again.

Listing of: AAtimer.prg

(C) JOHN FIELDEN

```
10 REM *TIMER**(C) JOHN FIELDEN **YAPE*
20 SCNCLR:PRINT"{black}          TIMER       "
30 VO=8:VOLVO:PRINT:PRINT
40 PRINT" PROGRAMMED USING YAPE A C=16 EMULATOR  FOR PC, XP"
50 PRINT" SETTINGS, CLICK DOWN TO SOUND SETTINGS"
60 PRINT"CLICK ON ENABLE SID SOUND,":PRINT"  AND ENABLE TED SOUND AT"
70 PRINT"CLICK ON OK AND YOUR READY TO GO."
80 PRINT" ***** TURN UP VOLUME ON COMPUTER ***** "
90 PRINT"(NOTE: ALSO TESTING CLOCK, ON SAME MENU)"
100 PRINT:PRINT"PRESS SPACE TO TEST SOUND."
110 PRINT:PRINT"PRESS ANY OTHER KEY FOR TIMER OPTIONS"
120 GETA$:IFA$=""THEN120
130 IFA$=" "THEN150
140 GOTO180
150 REM -TEST SOUND-
160 SOUND1,300,55
170 GOTO120
180 REM *************************
190 REM *              T I M E R              *
200 REM *************************
210 SCNCLR:PRINT"    TEST TIMER ":PRINT
220 PRINT"SOUND TAKES 10 SECONDS TO OCCUR USING       TED CHIP IN YAPE"
230 PRINT:PRINT" SPACE FOR TEST ":PRINT"ANY KEY FOR MAIN MENU "
240 PRINT:PRINT"THE SAME LOOP TAKES TWO SECONDS WHEN     CLEARING TIME SHOWN "
250 PRINT"{lred} WARNING: DO NOT WATCH TIME!!"
260 PRINT"{brown} PRESS A KEY ..."
270 GETA$:IFA$=""THEN270
280 IFA$<>" "THEN360
290 PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{black}    KEY PRESS"
300 TI$="000000"
310
FORJ=1TO500:X$=TI$:PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}";RIGHT
$(X$,2):NEXTJ
320 SOUND1,300,55
330 PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}                "
340 FORJ=1TO500:NEXTJ:PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}     "
350 TI$="000000":GOTO270
360 REM *MAIN MENU*
370 SCNCLR:PRINT"           T I M E R"
380 PRINT"    MAIN MENU "
390 PRINT:PRINT"1 - EGG TIMER "
400 PRINT"2 - STOP WATCH"
410 PRINT"3 - INTERVAL BEEPS"
420 PRINT
430 PRINT
440 PRINT" X = EXIT"
450 PRINT:PRINT"  PRESS A KEY"
460 GETA$:IFA$=""THEN460
470 IFA$="1"THEN1210
480 IFA$="2"THEN1000
490 IFA$="3"THEN560
500 REM
510 REM
520 REM
530 REM
540 IFA$="X"THENEND
550 GOTO460
560 REM *INTERVAL BEEPS*
570 SCNCLR:PRINT"{black}           T I M E R"
580 PRINT"{black}  INTERVAL BEEPS ":PRINT
590 PRINT:PRINT"ARROW UP = INCREASE DELAY"
600 PRINT"ARROW DOWN = DECREASE DELAY"
610 PRINT"ARROW RIGHT TO DISPLAY COUNT SO FAR"
615 PRINT"ARROW LEFT TO CLEAR COUNT DISPLAY"
620 PRINT:PRINT"PRESS 'R' TO RANDOMISE BEEPS"
630 PRINT"PRESS 'SPACE' TO MUTE  BEEPS"
640 PRINT:PRINT"PRESS 'X' TO RETURN TO MENU"
650 TI$="000000":DO
660 GETA$
670 IFR=0THENA=1:B=300:C=25:GOTO710
680 A=INT(RND(1)*3)+1
690 B=INT(RND(10)*950)
700 C=INT(RND(15)*35)
705 IFC<15THENC=C+15
710 IFA$="R"THENR=R+1:IFR>1THENR=0
720 IFA$=" "THENGOSUB880
730 IFA$="{up}"THENX=X+1:GOSUB860
740 IFA$="{down}"THENX=X-1:GOSUB860
750
IFA$="{right}"THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{do
wn}COUNT: ";Y
```

```
755
IFA$="{left}"THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{do
wn}                "
760 IFA$="X"THEN370
770 IFX<1THENX=15
780 IFX>15THENX=1
790 PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}    DURATION  ";X;" SECONDS"
800 X$=TI$:K$=RIGHT$(X$,2):Y=VAL(K$)
810 IFY=>XTHEN830
820 GOTO660
830 VOLVO: SOUNDA,B,C:TI$="000000"
840 GOTO660
850 STOP
860 PRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}                                    "
870 RETURN
880 VO=VO+8:IFVO>8THENVO=0
890
IFVO=0THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{
down}{down}{down}{down}{down} MUTE        "
900
IFVO=8THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{
down}{down}{down}{down}{down} SOUND ON"
910 RETURN
1000 REM *STOP WATCH*
1010 SCNCLR:PRINT"           STOP WATCH"
1020 PRINT:PRINT" PRESS SPACE TO START, THEN PRESS SPACE"
1030 PRINT" AGAIN TO STOP":PRINT"PRESS 'S' TO SHOW TIME":PRINT"'X' TO EXIT"
1040 GETA$:IFA$=""THEN1040
1050 IFA$="X"THEN370
1060 IFA$=" "THEN1080
1070 GOTO1040
1080 TI$="000000":PRINT"STARTED!"
1090 GETA$
1100 IFA$="S"THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}";TI$
1110 IFA$="X"THENPRINT"{clr}           STOP WATCH":PRINT" PRESS SPACE TO STOP WATCH!"
1120 IFA$=" "THEN1140
1130 GOTO1090
1140 PRINT"TIME STOPPED AT :";TI$:PRINT
1145 PRINT"SORRY, TENTHS OF A SECOND":PRINT" NOT YET AVAILABLE"
1150 PRINT:PRINT" PRESS 'X' FOR MAIN MENU"
1160 PRINT" OR SPACE BAR (' ') FOR RESTART"
1170 GETA$:IFA$=""THEN1170
1180 IFA$="X"THEN370
1190 IFA$=" "THEN1010
1200 GOTO1170
1210 REM *EGG TIMER*
1220 PRINT"{clr}           EGG TIMER"
1230 PRINT:PRINT"SELECT A TIME FOR YOUR EGG WHEN YOU ":PRINT" START BOILING"
1240 PRINT" '1' (YUK!) 1 MINUTE - '9' 9 MINUTES":PRINT" 'X' TO MAIN MENU"
1250 GETA$:IFA$=""THEN1250
1255 IFA$="X"THEN370
1260 IFA$<"1" OR A$>"9"THEN1250
1270 IFA$="1"THENT$="000100":T=1
1280 IFA$="2"THENT$="000200":T=2
1290 IFA$="3"THENT$="000300":T=3
1300 IFA$="4"THENT$="000400":T=4
1310 IFA$="5"THENT$="000500":T=5
1320 IFA$="6"THENT$="000600":T=6
1330 IFA$="7"THENT$="000700":T=7
1340 IFA$="8"THENT$="000800":T=8
1350 IFA$="9"THENT$="000900":T=9
1355 PRINT" YOU CHOSE ";T;" MINUTE/S"
1360 TI$="000000":PRINT" ' ' FOR TIME":PRINT" 'X' FOR RESTART"
1370 GETA$
1380 IFA$=" "THENPRINT"{home}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down} ";TI$
1390 IFA$="X"THEN1210
1400 IFTI$=>T$THEN1420
1410 GOTO1370
1420 PRINTTI$:PRINT"TIME UP!!!":VOL8
1430 PRINT" ANY KEY TO TURN SOUND OFF"
1440 GETA$
1450 SOUND1,500,15
1460 IFA$<>""THEN1480
1470 GOTO1440
1480 PRINT" ENJOY YOUR MEAL."
1490 PRINT" PRESS 'X' MAIN MENU"
1500 PRINT" PRESS ' ' MORE EGGS"
1510 GETA$:IFA$=""THEN1510
1520 IFA$="X"THEN370
1530 IFA$=" "THEN1210
1540 GOTO1510
```

In the Beginning Outro 1

By Lord Ronin

Well we finally reached the end of the 12 parts of our look at the Commodore user guide. I Hope that you were able to gain a something from it, even if it is the fact that I am a fanatic for the C= system.

If the user's manual was a fantastic book that taught you all you needed to know for doing basic programming, then not only would I have not written this series, uncounted authors would have not written books, articles for mags, and even some disk programmes to teach Basic. There isn't one book to teach the idea. Everyone has their own way of presenting the same information. In out local group for example we found that this diverse way is the one that worked. Different members saw the information, but one type of writer made it spark for them; different people and different writers. I would encourage you to look online as well as locally for books on the C=. Thousands upon thousands of books and magazines were made for the C=. Not all of them are in the landfill today many actually are now on line in different formats for use. Some have been put on CDs for use, Can't say the contact places. What I can say is that if you sign up for the homestead list at vcsweb.com. You will be in the same place as my group and able to reach the same sources. From that point you can grow in your own direction. Saying all of that, the users manual is still the first book to use for a starting point. You may not get any farther because you aren't interested in programming at that time. Yet it is the start of how to at least use the C=.

Many times I have said that I am not really a programmer. Sure I wrote this series. Wrote it from a lamers or new users point of view. Because it makes me only a few steps in experience than the guy that is just starting off I am not light years ahead of you. I Freely admit that some of my interpretations can and probably are wrong, my experience has been the book of this series. The Commodore Basic trilogy, one that was used at space camp. A few hundred type in programmes. Picking other peoples brains and trying to help the members of my local group.

In my work on this series a few things came to light; speaking with some programmers of note and long experience, All of them started on the same first book: that being the users manual. From there they did things like type in programmes from magazines, bought other books to learn programming, found a users group, or created a group of C= users. Then they learned together. Apparently one theme of feed back was the fact of programming is easier to learn with others. Face to face is great. I heard stories about phone calls and logging onto boards for many hours. Asking and answering questions.

I also have been told and many times over, that to do the programming of big games and demos and most stuff. You will need to study and practice with Machine Language. Don't know this language. I have seen some very good and enjoyable programmes that were done in Basic, as well as the same in ML. Not experienced enough to say what you should or should not do in this part. ML is said to be faster than Basic. It is also said to be smaller in space than Basic for the same programme actions. I do know that you can have Basic and ML mixed in one programme. There is an argument about the validity of learning Basic before going to ML. Some say that you need the understanding of Basic, to comprehend the actions of ML. These are the majority of those who have discussed this with me. There are some that say they wish they had started on ML and never messed with Basic. Reasons on the lines of having to re-learn concepts. I can't tell you what is the best. Or if you should attempt ML.

In fact you can not bother to programme anything and enjoy this PC. What I have done in the 12 parts is show you with my comments the users manual. They go into the basics of Basic for most of the book. Personally I find that the little I know on programming to be fun and makes me desire to learn more, as it is a creative and mind working subject. But I admit that it isn't a something for everyone. In fact I never thought that I would be interested or good enough to even attempt the things in the manual for several years. I only played games and went online and started to write for the local group and my game guild. Things sort of grew on me.

How many millions of programmes that are out there is anyone's guess. Commercial, freeware, public domain and the lot. Magazine type-ins, book type-ins and stuff that a guy made for himself. Floating around in collections, like what you may have. Plus in thousands of online web sites today. Most of them you won't be interested in at this moment. As you experience the C= more and more. Your interests will expand. OK that is because you will know more on the spectrum of things for the C=.

Overall point that I want to make at this time is now that you have seen a taste of programming, you can see that not only is this machine a very powerful device but it is friendly to the user. Most of all you are in control. You can create, You are doing the programming of the machine. Not being programmed by a something that you have no understanding of the workings or operation. Why in the industry that system is called "plug and pray".

I know of people, one of them the President of a users group. Who are not programmers They collect and use the C= hardware and software. But to programme something, they are worse than me. Yet that fact of their interest in programming, in no way diminishes their love or interest in the C=. In fact the one man above, started a convention/expo for the C=.

By going through the 12 parts you gained some understanding of the C=. Even if you didn't type in single program you will have seen the difference between the C= and the IBM PC platform. I hope that it was a positive experience. This PC and its users have outlasted the company, you don't see that for other PC platforms. But no I can't tell you the why of this fact. Why people are making software and selling it for low amounts or just giving it to the C= community, or why others are making hardware and passing out the technical detail for others to make. Like the vid conference thing, and the Ethernet connections. Nor way on eBay and other places, the C= is commanding high prices when compared to other PC items. The system is still loved 20 plus years after it was the worlds #1 PC platform. May you too find the pleasure and enjoyment that millions of others have throughout the world playing around with the C=.

As a subnote before ending we have dealt with the Commodore 64 in this series. There are other models. You may have the flat creamish coloured one called the Commodore 64c. You may have the Commodore 128, which is the flat one or the Commodore 128Dcr, that is the one that has the built in drive. What has been said in this series works with those machines. There is a slight difference with the sound chip but the rest of the information, in 64 mode on the 128 models all works the same. In fact all the programmes from the user's manual that are in this series where tested on a 64c and a 128Dcr in 64 mode. OK it was also written on the same 128Dcr in 64 mode.

There are other C= PCs. The Commodore PET came first in the mid/late 70s. Around 1980 was the introduction of the VIC-20. The 64 came out in 1982. Later they came out with the Commodore16. which in my limited experience is a very cut down version of the 64 haven't messed with mine much. Now there is also a black C=. This is called the Plus/4. I have two of them one is NTSC and the other is PAL I don't use them. They seem to be a great idea that failed. Great

because they added 4 built in things. Don't worry about those things. If you have a Plus/4. I can't tell you how much of this series worked for you. Never dealt with it at all. But I can tell you that the reason the Plus/4 failed is that it was not 100% compatible to the hardware and software for the 64. This from other users, and the fact that I could not read a disk that was created on the Plus/4, using a real 64. I find it an amazing thing that C= users killed off a C= system because it wasn't compatible. Yet windows users accept changes that make their hardware and software incompatible, as it was the normal and acceptable thing.

So saying all the above. Please enjoy your C= system. Millions have for decades.

Lord Ronin from Q-Link
Chancellor of the
Anything Commodore Users Group #447
23/Nov/2007ce

Short Additives Or some of the stuff that I said I would get to and didn't. Getting more information: I said many times that there was a titanic amount of things written for the C= PC. Tonnes of that have been cast aside. But there are collectors out there, like my users group, who preserve the information. Many web sites exist with C= information. There is even a Commodore web ring. Yet one has to start at someplace. So I strongly suggest that you go to http://www.vcsweb.com as your start. Rod and Gaelyne are very friendly to C= users. Plus you can collect not only files but you have access to C= mail lists, and therefore the people that are active in the scene. Who are much better at helping you than this old lamer.

I strongly suggest that you go to http://www.vcsweb.com as your start. Rod and Gaelyne are very friendly to C= users.

ON-LINE FILES: There are two types of "online" file sources. One is the new web and internet. The other is the older BBS. Files found on these sources are most probably compressed, in some form. They should end with an extension, such as "filename.XXX". Here the "XXX" represents some form of compression. There is another ending, but that is the next section.

In most of the C= compression files you will see the following.

SDA; that file will download for you. All you need to do is run it, and have a blank disk ready. The file will open up onto the blank <formatted> disk.

SFX; pretty much the same as above except that this one will go to any drive in the stack. While the SDA is limited to just drive #8.

LBR; stands for library. Not really a compression. More on the lines of putting a collection of files into one big file. There are many versions or editions of this one. You need a tool called "library" to open this type of file up. I use version 9 most of the time except for the Centipede BBS, which is a 128 programme and has its own library tool.

LNX; for this level of discussion. This "lynx" thing is similar to the above. Although I have been told that it has some compression. This is an undocumented statement to me. Again there is the need for a tool to open the file. Also there are several versions to this tool.

ZIP; Why yes we can do the PK1 version and we can also do the PK2 version. There are tools online for us to make and dissolve the PK zip.

CVT; well, now, this isn't a specific form of compression. But in your collection of files. Both at hand now and in the future. You may see this as an extension. Means "convert", and is used for Geos specifically. As the Geos USR file is not able to be transmitted raw over the phone lines, or copied with any thing else but Geos. So storage and on BBS's etc. The files are converted. Highest version of convert that I have is v3.1. Most people I know use v2.5. Meaning that if you see a file with this extension. Don't worry if it won't open up for you at this time.

XYZ; no not really a compression or anything that you will find. What do I mean by this heading? Well is that there are other extensions that you can see on files. Not all of them are storage compression systems. Here it is complex and past the scope of this project to explain fully. First some extensions are compression. Short lived or specific to one programme or source. Let's not ask me how many disks I destroyed of files from q-Link that I thought were bad. Because there was a small amount of compression used there and I didn't know about it at the time. All of them that I have found will require tools to open them up. I suggest that if you find these to hang onto them and ask on a list for assistance. Could be a treasure hidden on the disk. Second there are some programmes, like art programmes or example that add an extension to the name of the file. Granted that there are some that also make a prefix to the name of the file. These files will not run on their own, they are a part of the main program. So if they don't run, they aren't bad. You just need to find the right program for them to run.

IDIOTS; Someplace I have a box into which I toss disks that have programmes that won't work. Some of these I have over time saved from blanking. Why don't they work? Well the title of this section explains the who and not the why. OK the who is idiots and the why is the fact that they compressed or other things to store the program on a save disk. Really it would have been nice if they had put the extension on the file name! You see not all programmes will do that part for you. Our zip programs don't, or at least the three that I have used. I have to name it "lost cat.zip". Some versions of lnx and lbr will put the extension in for you.

There was a BBS called MudPit it ran out of Dallas Fort Worth area. I scored up the hard drive from 2nd hand. Files where still on the hard drive, so copied them. Would have been nice if the Geos files had the ".cvt" at the end. Many of the preserved GEnie C= areas on web sites today. Don't have that ".cvt" in the name. My point is, that if the file doesn't work. May be that the extension wasn't put on it. Had some zip ones that way and it isn't seen by the program then don't toss or blank them. Stick them off to the side and as you grow in understanding of the C=. You can try to rename the files with extensions and slowly find out what they are over time.

.D64: This is that next section I mentioned. Most of the files that I see on web sites and sent to me as attachments to email. Just happen to read sort of like "filename.d64". OK I must add here that I will zip these when I send them or put them on the BBS. Yeah that can all be done with a Commodore. Right then what is a .D64? Simply it is an electronic image of the entire side of a 1541 disk. Bummer is that it is 689 blocks. Not going to fit on a 1541 disk of 664 blocks. That is why I zip them for my work. These are used for emulators. Wait a moment, I have to add here that the file can be reverted back to normal for a C= PC. OK I have also been told that a .D64 does help with some copy protection stuff as well. Anyway, you can find mega amounts of these online but from my experience, I always zip them in my ram because it is easier for me with the tools at hand to just run the tool that unzips and reverts the .D64 into a real 1541. Since the tool does both at the same time.

You may also see files that are listed as .D71 or .D81. These are image copies of the 1571 double sided disk, and the 1581 3 1/2" disk respectfully. These are also too large to be put on their original disks. The tools that I scored online for the C= will also open these up for you. I have been told that there is also an image copy of at least the FD-2000 disk. But that is undocumented at this time.

Jiffy Dos: Found through http://www.cmdrkey.com at the time of writing. This is a chip that goes into your C= PC as well as into your drives, it comes complete with a small booklet manual. Way too much to write here on this one. If you can get it, then do so, if you

have it, you are lucky. Loading from a disk is fantastically faster. Keys are shortcut.

For example
F1 will give you the directory of a disk.
F4 will read on screen a text file.
F2 will present a Basic program on screen or you.

All your commands for file work start with the "@" symbol. Scratch as an example is open15,8,15,"s0:filename":close15. In Jiffy Dos it is just @s0:filename. Yet that error channel of #15 is still activated. Suffice to say that it is faster and things are easier with Jiffy Dos. No going into the tech stuff on interleave of the disk. But will say here are a couple of simple and fast copiers in Jiffy Dos. Get it, is my suggestion.

Press play and get the C= to send some power to it with the load command. You may have to press that in several times to do the job. OK when it is running, place the swab on the right hand side of the pinch roller. Ah that is from the perspective of facing the keys. Place it against the pinch roller and that metal thing that looks like a needle, called the capstan. Do this from he right side. If you do it from the left it will try to; and probably will tear off things from the swab. You want all that brown oxide stuff off the roller. So it may take some time. That brown stuff is oxide from the tape. Has a great deal of fun in making the tape wind around the pinch roller and the capstan. Or what is called "eating the tape".
Got that done and now it is the head cleaning time. No power for this one. Just press the key so the heads come into view. New swab of course. Wet it and now it is time to do the heads. Actually every metal looking thing that you see, where the tape would make

Languages: This entire thing was on the users manual with the C=64. Where we spent time with Basic v2

Languages: This entire thing was on the users manual with the C=64. Where we spent time with Basic v2. There are other forms of Basic out there. As an example the 128 has Basic v7. Some add on carts will give you Basic v4. But also I mentioned slightly that there is Machine Language <ML> and Assembly <ASM>. These two go together, from what I have been told. There are other computer languages that can be used on the C= PC. Comal, Cobalt, Fortran and C. This is not an extensive list of computer languages for the C=. Only the ones that I have seen or have in my collection. Loading from the disk: There is a bit more than what I wrote in the main section. Your friend is the "*" in this part. This is not just for loading a program but for hunting in a directory for a program. Starting with the directory, and that loading with the "$" they really could have made a specific key for that one Lets do the trick.

lO"$l*",8
lI

You will get all the files that start with l in the directory. Rather than all the files on the disk.

lO"l*",8
rU

This will load the first l file on the disk. Let's hope that it is a program and not a sequence file

Works also with more than one letter. Like lO"lost*",8 or lO"$lost*",8. First one will load the first lost anything program on the disk. Second one will set you up to see the listing of all the files that start with lost. Just did that with my set of lost cat stories.

128: Not the topic of this series. Yet if you have one you have a great PC. This unit not only has the C=64 inside. This unit has 40 column Basic v7. 80 column Basic v7. This PC also does several of the CP/M languages.

Cleaning: note that if you do this, it is at your own risk, However saying that it is a good idea to run you over some simple tips. Now there have been many a list, forum and magazine article on this task. So what I write here is just the real simple stuff. You need the swabs and alcohol that I mentioned in the series, plus a couple of other items that you may or may not have.

Lets do the tape machine first.
Forget the tape head cleaners and demagnetisers because they are a waste of money they look like a cassette and you run it in your machine. This one I can speak from years of in the field experience, I used to repair and service tape machines for a living! These things are not worth the plastic they are made from! What you need to do is flip open the lid of the tape machine. Take a look at the black rubber looking wheel. That wheel should be black, not a rust coloured band around the middle. This is the problem with head cleaner things. They don't clean this thing called the pinch roller. First dip your swap in the alcohol you don't want it dripping though. contact. But the heads are the most important at this time. That main one is the silvery looking little box. Well most of the time it looks that way. There are a couple of dark spots on it. Ah these you are not going to remove. as they are the part that actually puts the information on the tape, and reads it. But you may find what look like scratches that is really crud from dirty tapes. Bottom line is that you want that to be as shiny as possible. Here is a question of techniques. Some say you should move the swab in the direction of the tape flow. Others say that you should go up and down. Personally I do both for the movements of the cleaning swab. OK there is one other thing in there. Should be to the left of the head. Maybe black in colour. This is the erase head. Just sends out a pulse to erase the tape when you are recording. This too needs cleaning. Do it the same way. Oh yeah when the swab is dirty, toss it in a safe place from kids, pets and your smoke.

Demagnetizing time: remember science class in school? Where you took a magnet and ran the screw driver over it to magnetise it? Well the principle is the same for the tape machine. Tape is magnetic, the head is metal. You use the machine; you build up a magnetic layer, called gauss, this happens as the tape moves over the head. Well my head demagnetiser has to be plugged into the wall socket in order to create enough of a field to remove the gauss. Don't see how a small spinning magnet in a cassette case can do the same. Anyway, if you still have one of these tools. Then you already know how to use it. Most readers will not have this tool. Hobbyists, electronic stores and if you can find one, a repairman. Most probably do have the tool. You will need to go see these sources for a demag. If they just let you use the tool. Here are the guidelines. Take off your watch first. You can magnetise your watch. Keep the device away from all magnetic things. Like the box of disks, the monitor etc. After plugging it in, press the on button (if it has one) slowly bring it to the heads and move it up and down. If it is real bad you will hear a humming noise. Do the same to every metal part that is exposed to the tape. Do not touch the device to the metal though. When you feel it is right, slowly pull away to arms length and turn off the device. FWIW: in audio work, if your tape sounds muffled. You need to demag your system. I used to get $20 for a clean, test and demagnetized of tape machines. Save yourself bread and do it yourself.

Disk Drive guys: Well it is a bit easier for you. There are disk drive cleaners. A simple disk that has a semi abrasive cloth inside. Place the alcohol on it and run it around 30 seconds in the machine. I tell it to validate on the computer and just keep hitting that line again and again. Now the cleaning disks have a chart for number of uses. Not that I ever paid attention to it. Had a girl friend that knew of a fabric at the local fabric shop. That was the same roughish gauze stuff. She cut it to fit and inserted that into the sleeve. Works fine for me. But lost what it was called and the girlfriend. However at the time of writing, you can still get them from Radio Shack, the cleaning disk I mean not the girlfriends! 5 1/4" cleaners as well as 3 1/2". There are also online places to buy them. I'll do cleaning of the keyboard and a few other things in the 2nd part of the outro.

The Evolution of Forth

Commodore Free; would like to thank "Forth Inc" for its help with articles about Forth language. The original article can be found here, reprinted below is just a small section of what is available online http://www.forth.com/resources/evolution/index.html

Forth Inc also have a very nice online version of the classic book Starting Forth this is still considered a classic book about the Forth language and you will see its name pop up many times the full online book is available from here:
http://www.forth.com/starting-forth/

Abstract
Forth is unique among programming languages in that its development and proliferation has been a grass-roots effort unsupported by any major corporate or academic sponsors. Originally conceived and developed by a single individual, its later development has progressed under two significant influences: professional programmers who developed tools to solve application problems and then commercialized them, and the interests of hobbyists concerned with free distribution of Forth. These influences have produced a language markedly different from traditional programming languages

Authors
Elizabeth D. Rather
FORTH, Inc.
5959 W. Century Blvd..
Suite 700
Los Angeles, CA 90045

Donald R. Colburn
c/o Digital Media Magic
14712 Westbury Rd.
Rockville, MD 20853

Charles H. Moore
Computer Cowboys
40 Cedar Lane
PO Box 127
Sierra City, CA 96125

Presented at the ACM SIGPLAN History of Programming Languages Conference (HOPL II, April, 1993). Published in ACM SIGPLAN Notices, Volume 28, No. 3 March 1993.

Chuck Moore's Programming Language Early Development

Moore's programming career began in the late 1950's at the Smithsonian Astrophysical Observatory with programs to compute ephemerides, orbital elements, satellite station positions, etc. [Moore, 1958], [Veis, 1960]. His source code filled two card trays. To minimize recompiling this large program, he developed a simple interpreter to read cards controlling the program. This enabled him to compose different equations for several satellites without recompiling. This interpreter featured several commands and concepts that survived into modern Forth, principally a command to read "words" separated by spaces and one to convert numbers from external to internal form, plus an IF … ELSE construct. He found free-form input to be both more efficient (smaller and faster code) and reliable than the more common Fortran practice of formatting into specific columns, which had resulted in numerous re-runs due to mis-aligned columns

In 1961, Moore received his BA in Physics from MIT and entered graduate school at Stanford. He also took a part-time programming position at the Stanford Linear Accelerator (SLAC), writing code to optimize beam steering for the (then) pending two-mile electron accelerator, using an extension of some of his prior work with least-squares fitting. A key outgrowth of this work was a program called CURVE, coded in Algol (1964), a general-purpose non-linear differential-corrections data fitting program. To control this program, he used an enhanced version of his interpreter, extended to manage a push-down stack for parameter passing, variables (with the ability to explicitly fetch and store values), arithmetic and comparison operators, and the ability to define and interpret procedures.

In 1965, he moved to New York City to become a free-lance programmer. Working in Fortran, Algol, Jovial, PL/I and various assemblers, he continued to use his interpreter as much as possible, literally carrying around his card deck and recoding it as necessary. Minicomputers appeared in the late 60's, and with them teletype terminals, for which Moore added operators to manage character input and output. One project involved writing a Fortran-Algol translator and file-editing utilities. This reinforced for him the value of spaces between words, which were not required in Fortran source.

Newly married and seeking a small town environment, Moore joined Mohasco Industries in Amsterdam, NY, in 1968. Here he developed computer graphics programs for an IBM 1130 minicomputer with a 2250 graphic display. This computer had a 16-bit CPU, 8K RAM, his first disk, keyboard, printer, card reader/punch (used as disk backup!), and Fortran compiler. He added a cross-assembler to his program to generate code for the 2250, as well as a primitive editor and source-management tools. This system could draw animated 3-D images, at a time when IBM's software for that configuration drew only static 2-D images. For fun, he also wrote a version of Spacewar, an early video game, and converted his Algol Chess program into the new language, now (for the first time) called FORTH. He was impressed by how much simpler it became.

The name FORTH was intended to suggest software for the fourth (next) generation computers, which Moore saw as being characterized by distributed small computers. The operating system he used at the time restricted file names to five characters, so the "U" was discarded. FORTH was spelled in upper case until the late 70's because of the prevalence of upper-case-only I/O devices. The usage "Forth" was generally adopted when lower case became widely available, because the word was not an acronym.

Moore found the Forth-based 1130 environment for programming the 2250 superior to the Fortran environment in which the 1130 software was developed, so he extended it into an 1130 compiler. This added looping commands, the concept of keeping source in 1024-byte blocks and tools for managing them, and most of the compiler features we recognize in Forth today.

Most important, there was now a dictionary. Procedures now had names, and the interpreter searched a linked list of names for a match. Names were compiled with a count and three characters, a practice learned from the compiler writers of Stanford and which prevailed in Forth until the 1980's. Within a dictionary entry was a "code field" containing the address of code to be executed for that routine. This was an indirect threaded code implementation and was in use five years before Dewar's paper on indirect threaded coded appeared in Communications of the ACM The use of indirect threaded code was an important innovation, since an indirect jump was the only overhead once a word had been found. Dictionary entries could consist either of pointers to other "high level" routines or of machine instructions.

Finally, in order to provide a simple mechanism for nesting routines, a second stack called the "return stack" was added. The benefit of having a stack reserved for return addresses was that the other stack could be used freely for parameter passing, without having to be

"balanced" before and after calls. The first paper on Forth was written at Mohasco In 1970 Mohasco assigned Moore to an ambitious project involving a new Univac 1108 handling a network of leased lines for an order-entry system. He ported Forth onto the 1108, and arranged for it to interface to COBOL modules that did the transaction processing. The 1108 Forth was coded in assembler. It buffered input and output messages and shared the CPU among tasks handling each line. It also interpreted the input and executed the appropriate COBOL modules. This version of Forth added mechanisms for defining and managing tasks, and also an efficient scheme for managing disk block buffers similar to schemes in use today. Unfortunately, an economic downturn led Mohasco to cancel the 1108 project before completion. Moore immediately gave notice, then wrote an angry poem and a book on Forth that was never published. It described how to develop Forth software and encouraged simplicity and innovation

Philosophy and Goals
To Moore, Forth was a personal response to his frustration with existing software tools, which he viewed as a sort of "tower of Babel"

The software provided with large computers supplies a hierarchy of languages: the assembler defines the language for describing the compiler and supervisor; the supervisor the language for job control; the compiler the language for application programs; the application program the language for its input. The user may not know, or know of, all these languages: but they are there. They stand between him and his computer, imposing their restrictions on what he can do and what it will cost.

And cost it does, for this vast hierarchy of languages requires a huge investment of man and machine time to produce, and an equally large effort to maintain. The cost of documenting these programs and of reading the documentation is enormous. And after all this effort the programs are still full of bugs, awkward to use and satisfying to no one.

Moore conceived of Forth as replacing the entire "vast hierarchy" with a single layer, requiring only two elements: a programmer-to-Forth interface, consisting of minimal documentation (minimal because the interface should be simple and natural), and the Forth-machine interface, consisting of the program itself. His view was entirely personal, considering his own needs in the light of his own experience. The following excerpts from his unpublished book describe this view:

I've written many programs over the years. I've tried to write good programs, and I've observed the manner in which I write them rather critically. My goal has been to decrease the effort required and increase the quality produced.

In the course of these observations, I've found myself making the same mistakes repeatedly. Mistakes that are obvious in retrospect, but difficult to recognize in context. I thought that if I wrote a prescription for programming, I could at least remind myself of problems. And if the result is of value to me, it should be of value to others….

Above all, his guiding principle, which he called the "Basic Principle," was, "Keep it simple!" Throughout his career he has observed this principle with religious dedication.

As the number of capabilities you add to a program increases, the complexity of the program increases exponentially. The problem of maintaining compatibility among these capabilities, to say nothing of some sort of internal consistency in the program, can easily get out of hand. You can avoid this if you apply the Basic Principle. You may be acquainted with an operating system that ignored the Basic Principle.

It is very hard to apply. All the pressures, internal and external, conspire to add features to your program. After all, it only takes a half-dozen instructions, so why not? The only opposing pressure is the Basic Principle, and if you ignore it, there is no opposing pressure.

The main enemy of simplicity was, in his view, the siren call of generality that led programmers to attempt to speculate on future needs and provide for them. So he added a corollary to the Basic Principle: "Do not speculate!"

Do not put code in your program that might be used. Do not leave hooks on which you can hang extensions. The things you might want to do are infinite; that means that each has 0 probability of realization. If you need an extension later, you can code it later — and probably do a better job than if you did it now. And if someone else adds the extension, will he notice the hooks you left? Will you document this aspect of your program?

This approach flew in the face of accepted practice then as now. A second corollary was even more heretical: "Do it yourself!"

The conventional approach, enforced to a greater or lesser extent, is that you shall use a standard subroutine. I say that you should write your own subroutines.

Before you can write your own subroutines, you have to know how. This means, to be practical, that you have written it before; which makes it difficult to get started. But give it a try. After writing the same subroutine a dozen times on as many computers and languages, you'll be pretty good at it.

Moore followed this to an astounding extent. Throughout the 70's, as he implemented Forth on 18 different CPUs he invariably wrote for each his own assembler, his own disk and terminal drivers, even his own multiply and divide subroutines (on machines that required them, as many did). When there were manufacturer-supplied routines for these functions, he read them for ideas, but never used them verbatim. By knowing exactly how Forth would use these resources, by omitting hooks and generalities, and by sheer skill and experience (he speculated that most multiply/divide subroutines were written by someone who had never done one before and never would again), his versions were invariably smaller and faster, usually significantly so.

Moreover, he was never satisfied with his own solutions to problems. Revisiting a computer or an application after a few years, he often re-wrote key code routines. He never re-used his own code without re-examining it for possible improvements. This later became a source of frustration to Rather, who, as the marketing arm of FORTH, Inc. often bid jobs on the assumption that since Moore had just done a similar project this one would be easy — only to watch helplessly as he tore up all his past code and started over.

Today, Moore is designing Forth-based microprocessors using his own Forth-based CAD system, which he has re-written (and sometimes rebuilt, with his own hardware) almost continuously since 1979.

Moore considered himself primarily an applications programmer, and regarded this as a high calling. He perceived that "systems programmers" who built tools for "applications programmers" to use had a patronizing attitude toward their constituents. He felt that he had spent a great fraction of his professional life trying to work around barriers erected by systems programmers to protect the system from programmers and programmers from themselves, and he resolved that Forth would be different. Forth was designed for a programmer who was intelligent, highly skilled and professional; it was intended to empower, not constrain.

The net result of Moore's philosophy was a system that was small, simple, clean — and extremely flexible: in order to put this philosophy into practice, flexible software is essential. The reason people leave hooks for future extensions is that it's generally too difficult and time-consuming to re-implement something when requirements change. Moore saw a clear distinction between being able to teach a computer to do "anything" (using simple, flexible tools) and attempting to enable it to do "everything" with a huge, general-purpose OS. Committing himself to the former, he provided himself with the ideal toolset to follow his vision

The Forth Interest Group

In the late 1970's, Northern California was afire with the early rumblings of the Computer Revolution. Groups of interested individuals such as the "Home Brew Computer Club" were meeting to share interests and experiences. Magazines such as Radio Electronics published step-by-step instructions on how to build your own video display terminal, and even how to build your own microcomputer system.

Due to the high cost of memory and low level of VLSI integration, typical "homebrew" computers were very resource-constrained environments. Echoing back to the first generation computers, there was insufficient memory to concurrently support an editor, assembler and linker. Mass storage was slow and expensive, so many homebrew systems used paper tape or audio cassette tapes for I/O. Although some BASIC language products were available, they were typically very slow, and incapable of supporting significant programs. The stage was thus set for something else to meet the expanding needs of these hardy explorers and "early adopters."

Forth had been born and bred to exploit the minimal facilities of resource-constrained systems. It carried neither the excess baggage of a general solution nor a requirement for an existing file or operating system or significant mass storage. As Forth was used to tackle more and more difficult embedded computer applications, it started to claim the attention of the Northern California homebrew computer enthusiasts.

Bill Ragsdale, a successful Bay Area security system manufacturer, became aware of the benefits of microFORTH, and in 1978 asked FORTH, Inc. to produce a version of microFORTH for the 6502. FORTH, Inc. declined, seeing much less market demand for microFORTH on the 6502 than the more popular 8080, Z80 and 6800 CPUs.

Ragsdale then looked for someone with the knowledge of microFORTH and intimate familiarity with the 6502 to port a version of microFORTH to the 6502. He found Maj. Robert Selzer, who had used microFORTH for an AMI 6800 development system on an Army project and was privately developing a standalone editor/assembler/linker package for the 6502. Selzer wrote a 6502 Forth assembler, and used the Army's microFORTH metacompiler to target compile the first 6502 stand-alone Forth for the Jolt single board computer.

Selzer and Ragsdale subsequently made substantial modifications and improvements to the model, including exploitation of page zero and stack-implicit addressing architectural features in the 6502. Many of the enhancements that characterized the later public-domain versions were made during this period, including variable-length name fields and modifications to the dictionary linked-list threading. A metacompiler on the Jolt could target a significantly changed kernel to a higher address in memory. A replacement bootable image would then be recompiled by the new kernel into the lower boot address, which could then be written out to disk. At this point, Ragsdale had a system with which to meet his professional needs for embedded security systems.

During this period the Forth Interest Group (FIG) was started by Ragsdale, Kim Harris, John James, David Boulton, Dave Bengel, Tom Olsen and Dave Wyland. They introduced the concept of a "FIG Forth Model," a publicly available Forth system that could be implemented on popular computer architectures.

The FIG Forth Model was derived from Ragsdale's 6502 system. In order to simplify publication and rapid implementation across a wide variety of architectures, a translator was written to convert Forth metacompiler source code into text that, when input to a standard 6502 assembler, would replicate the original kernel image. In this way, neither the metacompiler nor its source code needed to be published. This is an important point. Forth metacompilation is a difficult process to understand completely. It requires the direct manipulation of three distinct execution phases and object areas, and is not something that a casual user wanted or needed.

By publishing assembler listings, the Forth Interest Group was able to encapsulate a Forth run-time environment in a manner that could be easily replicated and/or translated to the assembly language of a different computer architecture. It was the intention of the original team of implementers to thus stimulate the development of compatible Forth systems and the appearance of new vendors of Forth products.

After the 6502 FIG Model was published, FIG implementers published compatible versions for the 8080 and 6800 microcomputers and the PDP-11 and Computer Automation minicomputers. Over the years, volunteers added other platforms and documentation. The 1982 Forth Encyclopaedia by Mitch Derick and Linda Baker provided an exhaustive 333-page manual on FIG Forth, with flow charts of most words. In 1983 an ad in Forth Dimensions, the FIG newsletter listed: RCA 1802, 8080, PACE, 6502, 8086/88, 6800, 6809, 9900, Nova, Eclipse, VAX, Alpha Micro, Apple II, 68000, PDP11/LSI11 and Z80.

Today there are several thousand members of the Forth Interest Group in over fifteen countries. Since 1980, FIG has sponsored an annual conference called FORML (Forth Modification Laboratory), an educational forum for sharing and discussing new or unproven proposals intended to benefit Forth, and for discussion of technical aspects of Forth. Its proceedings are available from the Forth Interest Group

To read more about Forth and Forth Inc click here
http://www.forth.com/resources/evolution/index.html

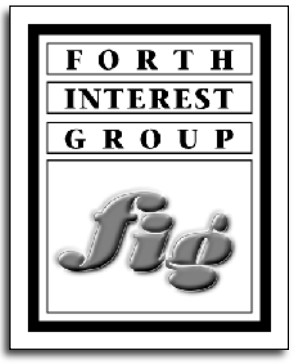
TO COMMODORE FREE MAGAZINE
FROM FORTH, Inc.

http://www.forth.com/resources/evolution/index.html

As Commodore Free states its purpose as "A free to download Magazine dedicated to Commodore computers" that satisfies the requirement for no commercial advantage so you have permission.

Please contact me if you have any other questions about Forth, our company, or its history.

Best regards,
Leon H Wagner
President
FORTH, Inc.
Dave Jaffe



INTERVIEW WITH
Silicon Valley Forth Interest Group (SVFIG)

COMMODORE FREE
Please introduce yourself to our readers

KEVIN APPERT
I'm Kevin Appert Vice Chair and Program Chair of the Silicon Valley Forth Interest Group (SVFIG).

AK> Andy Korsak, Forth user enthusiast since 1977.

PG> We're the peanut gallery. We make comments which are witty or instructive but anonymous. Some of the PG comments are from the February SVFIG meeting.

CF> Can you tell our readers about FIG (Forth Interest Group)?

KA> FIG has disbanded. The Silicon Valley Forth Interest Group (SVFIG) maintains as much of what FIG did as practical and appropriate including FIG's website at http://www.forth.org We practice and promote the Forth Programming language in our area and worldwide.

PG> FIG was a worldwide organization of Forth users started in the 70s with the creation of FIG-Forth.

KA> FIG-Forth was a publicly available implementation on every common microprocessor of the time. This brought Forth to the masses and precipitated a long era of Forth popularity.

PG> We'll quibble about the "EVERY common microprocessor".

KA> There were some obscure ones which weren't implemented but ALL the microprocessors that were commonly used at the time were implemented. If anyone can come up with a counter-example, I'd be pleased to hear about it.

CF> Where do you meet?

KA> SVFIG is currently meeting once a month at Stanford University in Palo Alto, California. We maintain a continuous "virtual meeting" through our email list and web site. Once per year SVFIG hosts Forth Day, the gala Forth festival notable as the venue for Chuck Moore's "Fireside Chat". Video from this past Forth Day will be on the site within the next month or so.

CF> Are there FIG groups in other countries?

KA> There are other communities of Forth devotees on the web and in other countries listed under "Other Forth Groups" on our home page at http://forth.org/ If you're interested in one of these groups you can contact them directly. There is a longstanding community on news:comp.lang.forth which can be reached through http://groups.google.com/ Our website is a member of the Forth Web Ring which you can navigate with the strip near the bottom of the home page.

CF> How could someone join?

KA> There is no formal membership. One participates by subscribing to the mailing list or coming to a meeting.

CF> Can you tell our readers how Forth as a language started and the approximate year?

KA> The year was 1968. Forth was invented by Chuck Moore to facilitate the design graphics software he was writing. You can read about it in more detail at some of the sites listed below...

History of Forth:
http://www.colorforth.com/bio.html
http://www.forth.com/corp/background.html
http://www.forth.com/resources/evolution/index.html

http://en.wikipedia.org/wiki/Forth_(programming_language)#History
http://en.wikipedia.org/wiki/Chuck_Moore

AK> The genius inventor, Charles H. Moore, was just interviewed and tells the story:
http://www.computerworld.com.au/article/250530/-z_programming_languages_forth?pp=1

CF> Do you know why there was a need for such a language to evolve?

KA> From the very start, Forth was the solution to any number of problems. It's very effective and fun to use.

AK> When Bill Ragsdale (look him up at http://qrz.com -- he's now a radio ham, as I am) and Dave Boulton gave us a brief talk about their setting up the first FIG back around 1977 at a Homebrew Computer Club meeting at Stanford University.

PG> The three factors which most fertilized the birth of Forth were:

1) Portability - an instruction-set Diaspora was making it necessary to have some way of moving easily from one machine to another.

B) Interactivity - computing was changing into a process where a programmer could interact in real time with the computer.

III) Simplicity - a simple and consistent architecture and syntax for human-machine interaction.

CF>When did you first come into "contact" with Forth as a language?

KA> I first learned about Forth from the Byte magazine "Forth Issue" and an inexpensive Forth which was available at the time for my 6502-based OSI-C1P computer.

PG> Forth, Inc. had a booth at Wescon in San Francisco in 1975.

CF> Why in your opinion is the language still in use and who still uses the language?

KA> Forth has much to recommend it. You can roam the web for hours reading about its benefits.
The most important aspect of Forth to me is the capability of producing reliable programs. There are two reasons for this:

1) Proper Forth source is small, modular and simple. The code is frequently "correct by inspection". You can look at it and see problems or their lack.

B) Modules can be incrementally and interactively tested in the Forth environment.

Please note that there are no guarantees of producing reliable code with ANY language. There are few "safety guards" in Forth. Like a sharp tool, Forth can be used to great effect or when used improperly it can lop off fragments of anatomy.

Mitch Bradley writes: "Forth is weird compared to most popular computer languages. Until you learn how, it is hard to read because it is not based on the syntax of algebraic expressions."

"But it is worth learning because a running Forth system gives you an extraordinary degree of low-level control over the system. Unlike most other programming environments that put up walls to hide or block access to "unauthorized" things, Forth makes it easy to get at anything, at any level from low to high."

CF>Who still uses the language?

KA>Forth programming, as distinct from using things written in Forth, is still popular in certain quarters. Here are a few users:

* There are a lot of hobbyists and enthusiasts.

* Forth Inc. is the largest commercial Forth house. Some of their successes are described on their website.

* IntellaSys has a chip with an array of 40 Forth processors.

* A lot of Sun machines start up with Open Firmware, a dialect of Forth. The One Laptop Per Child project hardware also uses Open Firmware.

* MPE in England is another Forth business.

* An MPE customer, Construction Computer Software (CCS) in Cape Town produce the MARS and CANDY applications which are a standard all over the world.

CF>What are the main benefits of the Forth language?

KA> How much time do you have? I could go on for hours! Here are some quick thoughts:

* It's a tool for the production of a RELIABLE product

* It's extensible. You add on to the language at will.

* It's quick to code. You can put together a working prototype in the blink of an eye and iterate it into a finished product with astounding speed. A do-over is frequently necessary even in the best-planned implementation effort and with Forth you can come to this realization sooner and re-implement within the time available instead of duct-taping your first try.

* It executes quickly. Small pieces are easy to optimize and an assembler is built into the language when necessary for machine-language speed.

* It's a programmer amplifier. It makes a good programmer better. Of course there are bad programmers and the attendant downside.

* It's frugal with machine resources. Some might tell you the era of the resource-limited computers is over but there are still plenty of applications where battery, memory, or other resources are constrained. Note that size also figures into my comments about reliability.

* It's portable. I'm not aware of any general-purpose processor that has no Forth. Implementation is comparatively simple and usually straightforward. I can demonstrate this by noting the vast array of available Forth's and Forth-like Languages available for the asking (and the buying, of course) for any computer you'd care to name.

* It can be its own operating system for embedded applications including a cooperative multi-tasker.

* It's the native language of custom or FPGA Forth processors like the IntellaSys SeaForth-24 chip, an array of 24 Forth processors with spectacular potential. More about this later on.

* It's interactive. You can test a module immediately after writing it, while you still have all aspects of it in your head.

* It's the language of choice for interacting with and bringing up new hardware. My favourite examples of this are the Mac and the Atari ST. Both of which had Forth as their 'milk language' in the early times on the bench.

CF> Is Forth an abbreviation like B.A.S.I.C (beginners all purpose symbolic instruction code)?

KA> No, it's just a word, not an acronym. Chuck Moore was using what he perceived to be the fourth generation of computing and had to throw one letter overboard because his OS only had five-character strings.

CF> Is Forth a compiled language? For the benefit of the readers unsure about the term "compiled" can you explain its meaning.

KA> As it says on some social networking sites, "It's complicated". AC-language compiler, for example, runs and translates source code into the directly-executable machine instructions of a given microprocessor or computer. When you compile Forth source it is translated by a simple compiler into instructions for Forth's "Virtual Machine". When you run a compiled C program the computer executes its instructions. In traditional Forth, an Interpreter takes the Virtual Machine instructions and gives them to the CPU as machine instructions. Some modern Forths have evolved into other approaches including pure compilation and something called subroutine-threaded code but old-school Forth had this method of two-phase compile-time and run-time execution.

CF> Do you know anything about the Forth implementation on the Commodore range of machines?

KA> Here's somebody's list:
http://www.npsnet.com/danf/cbm/languages.html# FORTH I recall using Forth on Pets' and 64s many years ago. If I wanted a Forth now for a Commodore machine I'd poke around on the Internet, ask around on the SVFIG email and comp.lang.forth then I'd try to port 6502FIG-Forth.

Here's an example of an Internet download:
http://cbmfiles.com/genie/C64-128ToolkitListing.html
1348 BLAZIN'FORTH MAZAX 860610 25200
Desc: Full Fig-83 Forth for C=64

CF>I really should ask "How do you get started programming in Forth"

KA> There are downloadable Forth packages on the Internet. I'd suggest you start with a PC and WIN32Forth. It has a community of users for questions and encouragement. You can join the SVFIG email list for more interaction with other Forth users.

CF> In every language the programmer comes out with a line like "I wish the language could do.." Is there something missing in Forth?

KA> Forth's extensibility makes you responsible for your own destiny in this regard. Mostly we add on things they perceive to be missing. Sometimes we appropriate other's additions.

CF> Is Forth A high or low level language?

KA> YES! Really, both... see below. This a great question for Forth.

From Computer Hope: High Level Language A type of advanced computer programming language that isn't limited by the type of computer or for one specific job and is more easily understood. Today, there are dozens of high-level languages; some commonly used high-level languages are BASIC, C, FORTAN and Pascal.

A low level language like Assembly Language has a one-to-one correspondence with the instructions of the machine. There is no abstraction of the tedious shuffling around of individual memory locations and registers. A moderate-level language like FORTRAN or C invokes functions and operations by name instead of having to have each small step spelled out and takes care of some of the book-keeping and busy-work for you so that you can express your program in something closer to English and equations.

Forth bridges these levels of abstraction. We start at the low-level simple operations like addition, subtraction and memory manipulation. Occasionally it may be necessary to go below even this low level into the built-in assembler. We use these simple pieces in combination with built-in higher-level constructs to build tools of increasing abstraction with which we solve the problem put before us.

CF> Does the language work via line numbers?

KA> There are no line numbers. There are no GOTOs, because although they have their uses we generally still consider them harmful.

CF> Can you give an overview of how the language works?

KA> To write a program you define modules. We call them "words". We keep them in a "dictionary". Forth starts out having some words and you define more words in terms of the existing ones until you have one word or a small set of words which performs your application.

For example, I can define a word to print "Hello world!" with the following:

```
: say_hi  ." Hello world!" ;
```

Then, I can make a word to it ten times like this:

```
: 10_hi  10 0  do say_hi  cr loop  ;
```

From the command line, I would just type 10_hi and the result would be:

```
10_hi     Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

.

Here's another example...

Let's say I wanted to put numbers in front of each line in the example above. I'd define a new word:

```
: Numbered-hi
   cr   10 0 do i 1+ .  ." "  say_hi  cr loop  ;

Numbered-hi
1  Hello world!
2  Hello world!
3  Hello world!
4  Hello world!
5  Hello world!
6  Hello world!
7  Hello world!
8  Hello world!
9  Hello world!
10  Hello world!
```

CF> How fast if Forth compared to other languages like ASSEMBLER or BASIC, and also how compact is the code?

KA> Forth tends to be much faster than BASIC, It can be as fast as assembly when it needs to be by good implementation and/or actual use of assembly code. It tends to be much faster than poorly written assembly. The key to this is spot optimization and good design. There are actually some Forths with optimizations but in general the onus is on you to obtain the desired performance.

CF> Are there any books our reader should look out for while starting his journey programming in Forth?

KA> Forth books are generally out of print. You can get used ones with fair ease. There are some hits on Amazon.com
Although it isn't in print on paper, I recommend "Starting Forth" by Leo Brodie. It's available online here:
http://home.iae.nl/users/mhx/sf.htmland on Forth, Inc.'s web site.
For the more advanced users, Forth, Inc. has a couple of good books here: http://www.forth.com/forth/forth-books.html

"Thinking Forth" isn't for beginners but it is highly recommended once you've gotten started. It's available from Amazon. They're using print-to-order technology to produce one copy at a time. It's online as well as in dead tree format.

CF> Is there a question you would have liked to have been asked?

KA> Where would I come across Forth in my daily life?
There are many applications of Forth in use today. One of the most prevalent is the FedEx "wand" used by all of that firm's couriers. PostScript and PDF are Forth-Like Languages. Forth, Inc. and MPE's websites have lots of interesting success stories. Ask Leon at Forth, Inc. if he'll do one of these interviews and tell you about some of Forth's victories.

CF> Do you have any comments you would like to add?

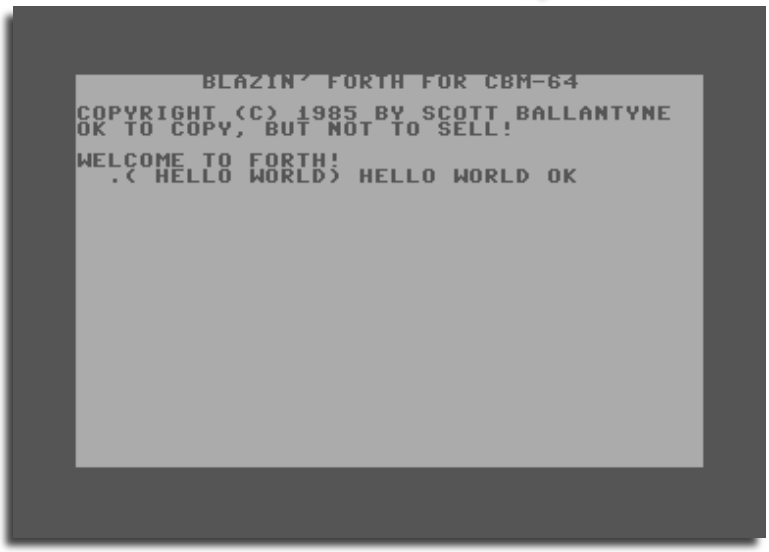
KA> There is lots of Forth information on the Internet. It's like drinking from a firehouse sometimes. http://www.forth.org and http://www.forth.org/svfig are good places to start. If you're interested in Forth and SVFIG, subscribe to the email list and read comp.lang.forth The IntellaSys chips will soon be available to hobbyists in small quantities. Look for announcements on http://www.forth.org/svfig andcomp.lang.forth

Forth Incorporated is the world's leading commercial Forth firm as you might expect. They have email lists for discussion of their SwiftForth and SwiftX products to which anyone may subscribe.

Wikipedia has an article on Forth here:
http://en.wikipedia.org/wiki/Forth_(programming_language)

Alternative Programming Languages Forth

By Paul Davis



Most people interested in programming the Commodore 8-bit machines start with BASIC. Eventually, however, the shortcomings of BASIC become all too apparent and this is when many programmers turn to machine code. But this is not the only option. There are several alternative high-level languages available for the Commodore. In this article we will take a look at Forth.

Forth is an unusual language but it is capable of producing extremely compact and efficient code. This makes it an ideal choice for use on micros with limited resources such as the Commodore.

In this introductory article I will attempt to give an overview of the language and the process of creating programs with it. I hope to whet your appetite enough that you will want to discover more. If the article is well received I will continue with a more in-depth tutorial. Please write in to the magazine and leave your feedback, even if it`s just to say "this sucks, you should have done C instead!"

Loading the Forth Language

I`ve made it as easy as possible to get started by preparing a tutorial disk that Nigel has kindly agreed to host in the download section of the Commodore Free web site. We will use a freely-distributable version of Forth called Blazin` Forth. This is a good implementation with several useful enhancements and is available for the C64 and Plus/4 (or expanded C16) although be warned the Plus/4 version is a bit buggy.

Insert the tutorial disk into drive 8. Plus/4 users press Shift+Run/Stop. C64 users enter the command

```
LOAD "BF",8,1
```

The Blazin` Forth environment will load and start automatically. Once loaded, the system will show a title screen and welcome message. Forth is waiting for you to type an instruction.

First steps

Let`s begin with the ubiquitous hello world. Type in the following exactly as shown, including all punctuation and spaces, then press Return:

```
.( HELLO WORLD)
```

That`s a dot, open bracket, space, hello world, close bracket, then press return. Forth responds by printing

```
HELLO WORLD OK
```

to the right of the command you just entered. And now you`re thinking 'what the hell kind of command was that!` right?

Forth treats any line you enter as a sequence of words separated by spaces. In the command above this means the first word is the dot-bracket combination. Yes, you read that right, .(is a word! In Forth any sequence of characters may be used as a word because the only delimiter that Forth uses to separate words is a space.

The behaviour of the dot-bracket word is to print out the text following it up to the closing bracket. The space between the dot-bracket and the text is required to make Forth recognise it as a word.

When Forth interprets a line without errors it displays the message 'OK` at the end of the current line. What if you type a word that Forth doesn`t understand? Enter the following line:

```
HELLO
```

The response you get varies between different dialects of Forth. BForth responds with the message 'not in current search order` and shows arrows pointing to the part of the line causing the problem. The error message is a bit cryptic to a beginner but can be taken to mean 'word not found`.

Now enter this line:

```
123
```

Forth responds with OK. Does this mean that 123 is a known word? Not exactly. What happens is Forth tries to interpret an unrecognised word as a number before giving up and printing the error message. In this case it succeeds and the number 123 is put in temporary storage ready for future use by another word.

There is a word to print out numbers in the storage area. It is simply a dot character. Enter the line

```
.
```

Forth displays the number 123. So, what is this area where the numbers are stored and how can we use it? The answer to that question is the most fundamental aspect of the Forth language. Values are stored on the 'stack`.

The Stack

If you are familiar with data structures you probably already know what a stack is and how it works. If the concept is new to you I would recommend reading chapter 1 of the online 'Starting Forth` book listed at the end of this article. If you just want to jump straight in to using Forth and fill in the details later, the most important points to understand about the stack are as follows. Values are placed on the stack one at a time and this is called 'pushing` a value. The most recently pushed value is said to be the 'top` of the stack. Values are retrieved one at a time from the top of the stack and this is known as 'popping` a value. The effect of these rules is that the values are retrieved in the reverse order that they are stored. The most recently stored is the first to be retrieved.

Let`s try an example to show how this works. To place the values 1, 2 and 3 onto the stack enter this line:

```
1 2 3
```

Forth responds with OK. There is a very useful word that shows the current contents of the stack. Enter this line:

```
.S
```

That`s dot-S. Forth responds 1 2 3 OK. The .S word lists the items in the order they were pushed. The last value in the list before OK is the top of the stack.

To pop each value off the stack and display it, enter this line:

```
. . .
```

That`s dot, space, dot, space, dot. Forth responds with 3 2 1 OK. The values are popped off the stack in the reverse order they were pushed.

Plus/4 owners will have noticed an extra number on the stack. This was caused by a bug when we tried to run the HELLO word that doesn`t exist. Normally, when an error occurs in Forth the stack is cleared. This doesn`t work properly in the Plus/4 version of BForth and instead an extra number is pushed onto the stack. The stack can be cleared manually with the following word, enter this now:

```
SP!
```

What if we try to pop a value when the stack is empty? Let`s see. C64 owners only enter the line:

```
.
```

Different versions of Forth give different messages. BForth responds with ‘stack empty`. Again, Plus/4 owners are hit with a particularly nasty bug here, if you try to pop a value off an empty stack BForth gets totally confused! You can recover by holding down Run/Stop while pressing the reset button and entering G 4018 in the monitor. This will restart BForth.

You may be wondering, at this point, what use all this is. Couldn`t we just use variables to store numbers in? Indeed, we could, but how many times have you needed to use a variable just to hold a temporary value during a calculation or to pass a parameter to a subroutine? Most BASIC programmers find themselves re-using these temporary variables. This opens up the possibility of bugs caused by values from one part of a program interfering with another part.

The great thing about the stack is its temporary nature. It is perfect for storing intermediate results of calculations and parameters to other routines. When you`re done with a value, the space it occupied is automatically re-used for something else. By using the stack, your programs are kept compact and tidy because they don`t need to keep track of all those temporary values.

Arithmetic

Okay, let`s expand our list of words and learn how to do calculations. Forth doesn`t understand expressions the same way as other languages do, it only understands words. Each word will take any parameters it needs from the stack and will also deposit any result it creates back onto the stack. An example will demonstrate this. Enter the following line:

```
1 2 + .
```

That`s 1, space, 2, space, plus, space, dot, return. Forth responds with the answer 3. Let`s break this down to see what`s happening. The first two words, 1 and 2 push the numbers onto the stack in that order. The + word takes two numbers off the stack and adds them together. At this point the stack is now empty. Once it has performed the addition, the plus word pushes the result onto the stack. The stack now contains a single value 3. Finally, the dot word takes the value from the stack and displays it. The sequence of words we entered has the same effect as PRINT 1+2 in BASIC.

Many people find this re-arrangement of numbers and operators confusing at first. It takes time and practice to be able to read the expressions in Forth fluently.

Now, consider a sum such as 1 + 2 * 3. The answer to this should be 7 since the multiplication has a higher precedence than the addition. Most languages need to support operator precedence rules and allow the use of brackets to override them. A big advantage of Forth using the stack to perform all arithmetic is that there`s no need for any of this. You simply perform the parts of the calculation in the order you need them. To perform the equivalent of 1 + (2 * 3) use the following:

```
1 2 3 * + .
```

This will display a result of 7. Let`s just take a moment to trace through how this works. The following table shows the contents of the stack after each step of this sequence of words. The rightmost value in the list here is the ‘top` of the stack:

```
Word  Stack     Effect
      --empty--
1     1         1 pushed on stack
2     1 2       2 pushed on stack
3     1 2 3     3 pushed on stack
*     1 6       2*3 = 6 pushed
+     7         1+6 = 7 pushed
.     --empty-- 7 displayed
```

Notice that each operator takes two items from the stack. For the multiplication, this means it takes the 3 and the 2 leaving the 1 still on the stack. This number 1 is taken, along with the result of the multiplication, as parameters to the addition.

To perform the equivalent of (1 + 2) * 3 and get a result of 9, the sequence of words would be:

```
1 2 + 3 * .
```

For addition and multiplication the order of the two parameters on the stack doesn`t matter. For subtraction and division, however, it does matter. To perform the equivalent of 5 - 3 enter this line:

```
5 3 - .
```

The result 2 is displayed as expected. To perform the equivalent of 7 / 2 enter this line:

```
7 2 / .
```

Forth displays the result 3. Where`s the half gone? Well, one of reasons for Forth`s speed is the fact it uses integer arithmetic. That is, it only deals with whole numbers. In the calculation above the remainder has simply been discarded. Fortunately, there`s a way to obtain the remainder, the MOD word. Enter this line:

```
7 2 MOD .
```

This displays the result 1, the remainder after dividing 7 by 2. One of the unique aspects of Forth is that a word can return more than one result at a time. We can see this in action with the word /MOD. This performs the division and leaves both the result and the remainder on the stack. Enter this line:

```
11 4 /MOD . .
```

Forth displays 2 and 3, the result of the division and the remainder, respectively.

Try playing around entering different expressions to get a feel for the way Forth uses the stack and how more complicated expressions can be built up from simpler components. Remember you can use the .S word to inspect the contents of the stack at any time without destroying its contents.

Making your own words

Now let`s introduce another new word, EMIT. This word will output a character whose ASCII code is on the stack. Let`s try it. Enter this line:

```
147 EMIT
```

The screen will clear and Forth displays the OK prompt as usual. 147 is the Commodore ASCII code for ‘clear screen`. So this command is the equivalent of doing PRINT CHR$(147); in BASIC.

Having to look up the ASCII code for a particular character is a chore, something the computer should be able to do for us. Fortunately, there is a word to do just that. Enter the following line (clr means press the Shift and Clr/Home keys):

```
CONTROL "clr" EMIT
```

The screen will clear as before. This is all very useful but wouldn`t it be nice if we could just say something like CLS to clear the screen instead of having to use EMIT every time? Well, you can, because Forth lets you define new words. Here`s how to do it. Enter this line:

```
: CLS 147 EMIT ;
```

That`s a colon, space, CLS, space, 147, space, EMIT, space, semi-colon, return. The colon means ‘define a new word` and must be followed by the name of the word to be defined. Then everything up to the semi-colon is taken to be the definition of that word. If the definition is accepted Forth will print OK. Let`s see if our new word is now recognised. To clear the screen, enter:

```
CLS
```

So, how does this work? Forth has what is called a ‘dictionary` of known words. When you enter a line of text Forth will look up the word in the dictionary and match it to a pre-compiled sequence of instructions. By using the colon word, we have added an entry to the dictionary. For all intents and purposes, we have extended the language.

You can get a list of the words in the dictionary at any time using this command:

```
WORDS
```

The Forth dictionary contains many words. The control key (or Commodore key on the Plus/4) may be used to slow down the listing, pressing a key will pause the output until another key is pressed. Run/Stop will stop the listing.

New words are added to the start of the list so our newly created CLS word is shown first.

We can now create any new word we like so how about making a word that prints out ‘hello world`. Enter this line:

```
: HELLO .( HELLO WORLD) ;
```

Forth responds with HELLO WORLD OK. Huh? This isn`t what we expected. What`s going on?

Here we have come across one of Forth`s little quirks. It isn`t important at this point to explain why this has happened, just be aware that the dot-bracket word will always display the text immediately, even inside a word definition.

The new HELLO word still compiled so what will it do? Try entering it to find out. Hmm, nothing happened. Let`s have a look at how the command has been defined. Enter this line:

```
SEE HELLO
```

The SEE word will ‘decompile` any word in the dictionary. In our case it will show:

```
: HELLO ;
```

So our hello word is empty. No wonder it`s doing absolutely nothing!

Forth uses a different word, called dot-quote, for printing within a colon definition. Enter this revised version of our hello word:

```
: HELLO ." HELLO WORLD" ;
```

Forth warns us that the word HELLO already exists then says OK. Enter the word HELLO and sure enough, our message is now printed out.

Forth has some other useful words for controlling the layout of the text you print. The CR word prints a ‘carriage return` so the following text will start on a new line. The SPACES word takes a parameter on the stack and outputs that number of spaces. To see these words in action, enter these lines:

```
: MARGIN 5 SPACES ;
: HELLO CR MARGIN ." HELLO WORLD" ;
HELLO
```

Once again Forth warns us that the HELLO word already exists. Running the new definition of HELLO prints the greeting on a new line, indented by 5 spaces.

Notice how we are using small, simple word definitions like CR and MARGIN to build up a word that performs a more complicated sequence of actions. This is typical of the way Forth programs are constructed.

You may be wondering what has happened to the old definitions of HELLO? Actually, they are still there! Enter the command

```
WORDS
```

Press Run/Stop after a couple of lines have been printed. If you look at the first line you will see three entries for HELLO, one before MARGIN and two after. Creating a word only adds it to the start of the dictionary. It doesn`t remove any previous entry with the same name. When Forth is looking up a word in the dictionary it uses the first matching word it finds, the latest version. You can delete dictionary entries using the word FORGET. This word will delete all dictionary entries from the start of the list up to and including the first matching word. Enter these lines:

```
FORGET CLS
WORDS
```

Press Run/Stop after a couple of lines have been displayed. If you look at the start of the list you will see that the CLS word, MARGIN and all versions of HELLO have now gone.

Peeking and Poking

C64 BASIC is particularly lacking, especially in the areas of graphics and sound handling. Programs often consist of little more than a series of POKE commands. Looking on the bright side of this, however, it does make porting programs to another language a fair bit easier. All you have to learn is the equivalent of POKE and PEEK. Enter the appropriate line for your machine to turn the border black:

```
0 53280 C!    (C64)
0 65305 C!    (Plus/4)
```

The word for POKE is C-exclamation-mark, more commonly said as C-store (the C stands for character, Forth`s name for a single byte value). Some of these numbers should look familiar to C64 users. This command is the equivalent of POKE 53280,0 in BASIC. The colour value is put onto the stack first, followed by the address. We can take advantage of this arrangement and create a new word to set the border colour that takes the colour value as a parameter on the stack. Enter this:

```
: BORDER 53280 C! ;     (C64)
: BORDER 65305 C! ;     (Plus/4)
```

Notice that we only include the memory address parameter to the C! word. The value to be stored must be pushed on the stack before we call the BORDER word. Let`s try it:

```
6 BORDER
```

The border colour should change to blue. This method of using the stack to pass parameters to other words is used a lot in Forth and is an important concept to grasp. Here is a trace through each step of the above command:

```
Word    Stack     Effect
        --empty--
6       6         6 pushed on stack
BORDER  6         call word BORDER
53280   6 53280   53280 pushed
C!      --empty-- blue border
```

We now have a word for changing the border colour, so let`s complete the set by adding two more words to change the background and cursor colours too:

```
: PAPER 53281 C! ;     (C64)
: INK 646 C! ;         (C64)
: PAPER 65301 C! ;     (Plus/4)
: INK 1339 C! ;        (Plus/4)
```

Now we can change the colours on the screen easily:

```
6 PAPER 14 BORDER 14 INK   (C64)
38 PAPER 70 BORDER 70 INK  (Plus/4)
```

Plus/4 users have the advantage of a larger palette of colours, 16 primaries each with 8 shades. The luminance values are numbered 0 (darkest) through 7 (brightest) and should be multiplied by 16 then added to the primary colour number to get the value to use with our new words. Since this is Forth, we can make our lives easier by simply creating a new word to do this for us:

```
: LUM 16 * + ;
1 2 LUM PAPER 7 6 LUM INK
```

The background should change to a darkish grey and the cursor to yellow. The first number pushed on the stack is the primary colour (1=white), the second number is the luminance (2). The LUM word calculates the colour number (16*2+1=33) which is then passed to the PAPER word. The cursor is similarly set to brightish (6) yellow (7).

You may also be aware that the Plus/4 supports flashing colours. To use these, add 128 to the colour number. Again we can use a word to make it easier:

```
: FLASHING 128 + ;
1 7 LUM FLASHING INK
```

The cursor changes to flashing white. This could get annoying so use 103 INK to get the cursor back to yellow. C64 users can wake up again now!

We can go a step further and define names for the colours by using the CONSTANT word:

```
(C64)              (Plus/4)
0 CONSTANT BLACK   0 CONSTANT BLACK
1 CONSTANT WHITE   113 CONSTANT WHITE
2 CONSTANT RED     50 CONSTANT RED
3 CONSTANT CYAN    99 CONSTANT CYAN
4 CONSTANT PURPLE  68 CONSTANT PURPLE
5 CONSTANT GREEN   85 CONSTANT GREEN
6 CONSTANT BLUE    38 CONSTANT BLUE
7 CONSTANT YELLOW  103 CONSTANT YELLOW
```

You get the idea. Now try this line:

```
CYAN BORDER WHITE PAPER BLACK INK
```

As you can see, just a few simple word definitions have turned an arcane series of POKEs into much more readable instructions.

The equivalent word for PEEK is the C@ word. This takes the address as a parameter on the stack and leaves the contents of that location on the stack. Try this:

```
147 EMIT 42 EMIT
1024 C@ .     (C64)
3072 C@ .     (Plus/4)
```

The first line puts an asterisk at the top left of the screen. The second line reads the memory location corresponding to the top left of the screen and displays its value. Forth should respond with 42.

Repetition and Looping

Most programs need to repeat a sequence of instructions multiple times. Forth has a variety of words for looping although they operate quite differently from what you may be used to. As an example we will create the equivalent of this BASIC loop:

```
FOR I=1 TO 5:PRINT I:NEXT
```

This loop simply prints the value of the loop index each time through. In Forth we would write this as:

```
: COUNTING 5 0 DO CR I . LOOP ;
COUNTING
```

Notice that we have created a new word called COUNTING here as well. If you tried to enter the DO/LOOP instructions directly, Forth would issue a warning that DO can only be used in a word definition.

Breaking the instructions down, we get the following components. The first number pushed on the stack is the end value. The loop will end when the counter reaches this value. The second number pushed on the stack is the start value. The index counter will be set to this value before entering the loop. The DO word marks the start of the loop. The I word puts the current index counter value on the stack ready for the dot word to print out. Finally the LOOP word increments the counter and goes back to the start of the loop if it is less than the end value. You will notice that this results in a loop that runs 5 times but the values of the index are 0 through 4.

Conditions

The other staple part of any language is decision making. In Forth decisions are made using this sequence of words:

```
condition IF action ELSE action THEN
```

The ‘condition` part of that statement is any sequence of words that leaves a ‘true` or ‘false` value on the stack. This might be comparing two numbers for equality or checking if a key is pressed for example. The IF word takes the result of this condition as a parameter and if the result was ‘true` performs the action words after the IF. The words after the ELSE are performed if the condition was ‘false`. In both cases, the word THEN marks the end of the statement. The ELSE section is optional, it is not required if you only want to perform an action when some condition is true.

The order of the words in a Forth IF statement will look odd to users familiar with other languages. In BASIC, the word THEN is used to separate the condition from the action. In Forth, the word IF serves this purpose so THEN is used to mark the end of the conditional statement. Don`t worry if it takes some time to adjust, it`s easy to be thrown by this when first learning Forth. Some examples should help make things clearer.

Condition statements, like loops can only be used inside word definitions in Forth. Enter this word definition:

```
: YESORNO KEY ASCII Y =
IF ." YES" ELSE ." NO" THEN ;
```

The KEY word will wait for a key to be pressed and then put its ASCII code on the stack. The words ASCII Y will put the ASCII code for the letter Y on the stack. The = word will then compare those two values and put 'true` on the stack if they are equal, 'false` if they are not. The IF word will take this value and continue to print YES if it was true (Y was pressed) or jump to the ELSE part that prints NO if it was false (any other key).

Enter the word YESORNO and press Return. Then press the Y key. The response should be YES. Enter the command again and press a different key. The response should be NO.

Forth has several comparison words for creating conditions: = you have seen, compares two values for equality, there is also < (less than), > (greater than) and <> (not equal). Forth doesn`t have <= and >= but they can be defined as follows if you would like to use them:

```
: <= > NOT ;
: >= < NOT ;
```

Conditions can also be combined by using the words AND and OR. These words take two true/false values on the stack and return one true/false value depending on whether both parameters are true (AND) or either are true (OR). Try entering this:

```
: ISSPACE DUP 32 = SWAP 160 = OR ;
```

This word takes an ASCII code on the stack and returns true only if that number is 32 (space) or 160 (shift+space). There are a couple of new words here. DUP creates a duplicate copy of the top number on the stack. We need to do this because that number will be gone after the first comparison with 32. We need a copy for the comparison with 160 later. After the first comparison, the stack will contain two numbers, a copy of the ASCII code and a true/false value in that order. For the next comparison we need the ASCII code to be on the top of the stack so we use the SWAP word to achieve this. It takes the top two values on the stack and swaps them around, now the ASCII code is on top we can perform the other comparison after which there will be two true/false values on the stack. The OR word takes these two values and returns true if either of them were true. Run the new word using this line:

```
KEY ISSPACE .
```

Press the space bar. The result -1 is displayed, the numeric value of true. Enter the same command again and press some other key such as Return. The result 0 is displayed, the numeric value of false. Enter the command one more time and press shift and space together. The result as expected is true.

Conditional looping

In addition to the DO loop which repeats a section of code a specified number of times, Forth also provides looping words that use a condition to end the loop. To repeat a section of code until some condition is satisfied, Forth provides the following construct:

```
BEGIN action condition UNTIL
```

Let`s try using this along with our ISSPACE word to make a word that waits for the space bar to be pressed:

```
: WAIT BEGIN KEY ISSPACE UNTIL ;
```

Enter the command WAIT and press a few keys on the keyboard. The loop keeps running until you press either the space or shifted space key. Do that now and the loop terminates and Forth responds with OK.

The condition is placed at the end of an UNTIL loop which means that the action part of the loop will always run at least once. Forth has another type of loop where the condition is at the beginning and so the actions may never run at all. It looks like this:

```
BEGIN condition WHILE action REPEAT
```

Here is an example that echoes each character typed onto the screen until the Return key is pressed:

```
: EMITKEYS BEGIN KEY DUP 13 <>
WHILE EMIT REPEAT DROP ;
```

Run the word by entering the command EMITKEYS. Now, type something in. The characters will appear on screen as you type. When you`re done, press Return to end. The BEGIN word marks the start of the loop. We use the KEY word to wait for a key to be typed and then DUP to make a copy of it. The comparison will use up one of these copies, we need the other to be able to print it out. The key is compared against ASCII code 13 (the Return key) using the <> word. This will keep going round the loop while the key is anything other than Return. The loop action is simply to EMIT the character typed to the screen. The REPEAT word marks the end of the loop. The final word DROP discards the topmost value on the stack. This is necessary because when the loop terminates (on pressing the Return key) the duplicate character has not been processed in the loop actions so it is still on the stack.

These stack manipulations such as DUP, SWAP and DROP are a fundamental part of Forth programming and it is important to understand how and when to use them. There isn`t enough space in this article to go into much detail. More information can be found in chapter two of the 'Starting Forth` online book.

Memory movement and filling

Forth has some built-in words for quickly filling and moving memory. These are typically written in machine code and are extremely quick. On the C64, the screen starts at location 1024, on the Plus/4 it is at 3072 so we will use this address to demonstrate. First, to avoid having to list two different sets of commands each time, let`s create a variable to store the address of the text screen. Enter these lines:

```
VARIABLE SCRN
1024 SCRN !    (C64)
3072 SCRN !    (Plus/4)
```

The first line defines the variable name and allocates some memory for it. The second line stores a value in the variable. Using the variable name SCRN as a word causes the address of that variable to be pushed on the stack. The ! word is used to store a value in that address. This is how Forth assigns values to variables. Now try this:

```
SCRN @ 1000 42 FILL
```

The screen instantly fills with asterisks. The name of the variable puts the address of that variable onto the stack, the @ word reads the contents of this memory location to get the value we put in earlier, the address of the screen. This becomes the first parameter to the FILL word, the second parameter is the number of bytes to fill (the screen has 1000 characters), and the third parameter is the value to fill with (42 is the screen code for an asterisk). A variation of this command is ERASE which is equivalent to FILL with a value of 0. Try this:

```
SCRN @ 1000 ERASE
```

The screen fills with @ signs (screen code 0). The ERASE word is of more use for clearing bitmap graphics and sprites.

Press Shift and Clr/Home to clear the screen. Move the cursor to the right a couple of spaces and type something on the top line of the screen but don`t press Return. Move the cursor down a few lines, press Return then enter this line:

```
SCRN @ DUP 40 + 40 MOVE
```

The text you typed on the top line of the screen has been copied to the second line. We use the variable to get the address of the screen, this is the location to copy from. Then we create a duplicate of this value and add 40 to it (there are 40 characters per screen line), this is the location to copy to. The third parameter, 40, is the number of bytes to move. Now try each of these lines:

```
SCRN @ DUP 1+ 39 MOVE
SCRN @ DUP 1+ SWAP 39 MOVE
```

The first instruction scrolls characters on the top line of the screen to the right, the second scrolls to the left. The 1+ word is a quick way to add 1 to the number on top of the stack.

Turtle graphics

Now for something a bit more fun. One of the more unusual features of Blazin` Forth is its graphics support which includes an implementation of Turtle Graphics. This is a system created as part of the Logo programming language for educational purposes. It originally used a mechanical robot ‘turtle` with a pen in its belly to draw geometric shapes on a piece of paper on the floor. By giving the turtle simple instructions like ‘move forward 10 steps`, ‘turn left 90 degrees` and so on, it is possible to create intricate patterns without having to understand the mathematics behind it.

To initialise the graphics screen enter this command

```
DRAW
```

The screen is split into two areas, a drawing area at the top and a command area at the bottom. The ‘turtle` is represented by an arrow head pointing upwards. This shows the direction in which the turtle is currently heading. Now enter this line:

```
100 FORWARD
```

The turtle moves 100 steps up the screen drawing a line behind it. To make it turn, try this:

```
90 RIGHT
```

The turtle turns 90 degrees to the right. To draw another line, enter this:

```
50 FORWARD
```

A shorter line is drawn perpendicular to the first. Now type:

```
HOME
```

The turtle is returned to the ‘home` position at the centre of the screen facing north. Another line is drawn as it does so and the end result is a triangle.

To move the turtle without drawing we have to ‘raise its pen`. This is achieved with the PENUP word. When we are ready to draw again we can use the PENDOWN word. Most turtle graphics words can be abbreviated to save typing. PENUP is PU, FORWARD is FD, LEFT is LT and so on. Enter this line:

```
PU 90 LT 100 FD 90 RT PD
```

The turtle should now be positioned some distance to the left of the triangle and facing upwards. Many geometric shapes can be created by repeatedly drawing a line and then turning a bit. For example a square is 4 repetitions of ‘draw a line` and ‘turn 90 degrees`. Let`s try creating a word to draw a square of any size:

```
: SQUARE 4 0 DO DUP FD 90 RT
LOOP DROP ;
50 SQUARE 30 SQUARE 10 SQUARE
```

This will create our new SQUARE word and draw three successively smaller squares. We can use a loop to repeatedly draw squares at different rotations to create a pattern:

```
: SQUARES 36 0 DO DUP SQUARE
10 RT LOOP DROP ;
DRAW
80 SQUARES 50 SQUARES 30 SQUARES
```

As you can see, impressive results can be achieved with very little effort. You can see the whole screen by entering the command NOSPLIT and get back to a split screen using the word SPLITSCREEN (or SP).

Since many different shapes can be created by simple repetition of line drawing and turning, let`s create a new word to allow us to draw any shape:

```
: SHAPE 0 DO OVER FD DUP RT
LOOP 2DROP ;
```

The word takes three parameters: line length, angle and line count. Another couple of new words have sneaked in here. OVER makes a copy of the second number on the stack (remember DUP makes a copy of the number on top of the stack). 2DROP is a quick way of discarding two numbers off the stack. Try these examples:

```
DRAW 100 90 4 SHAPE 100 120 3 SHAPE
120 LT 60 72 5 SHAPE 60 60 6 SHAPE
```

For regular geometric shapes like these, the angle to turn each time is 360 divided by the number of sides. We can make a new word to do this calculation for us:

```
: POLY 360 OVER / SWAP SHAPE ;
DRAW 100 4 POLY 100 3 POLY
120 LT 60 5 POLY 60 6 POLY
```

By drawing a polygon with many sides, we can approximate a circle like this:

```
DRAW 10 40 POLY
```

The SHAPE word can be used to create other effects too, try entering these lines:

```
: STAR 144 5 SHAPE ;
DRAW 100 STAR
```

We can approximate a spiral by drawing consecutively longer line segments after each turn. Enter this definition:

```
: SPIRAL 2 PICK DO OVER I SWAP /
FD DUP RT LOOP 2DROP ;
```

The first parameter to SPIRAL controls the line length. The larger the number, the slower the length increases. The second parameter is the angle to turn each time. The third is the number of line segments to draw. The new word this time is PICK. This will make a copy of the number on the stack at a specified position from the top. 0 PICK is the same as DUP, 1 PICK is the same as OVER. Here, 2 PICK makes a copy of the third value, the line length. To draw an example spiral, enter this line:

```
DRAW 10 10 210 SPIRAL
```

Now let`s try building up our shapes to create a simple picture. Enter this definition, pressing Return after typing each line:

```
: SNAIL 10 10 106 SPIRAL
20 FD 30 RT 120 FD
3 10 18 SHAPE
100 LT 20 FD 20 BK 100 RT
10 FD 80 LT 30 FD 30 BK
88 RT 90 FD ;
DRAW SNAIL
```

By using recursion we can create even more impressive effects. Try this, again press Return after each line:

```
: TREE-BRANCH DUP 4 > IF DUP 3 4 */
25 LT OVER FD DUP RECURSE
OVER BK 50 RT OVER FD RECURSE
DUP BK 25 LT THEN DROP ;
: TREE DUP FD DUP TREE-BRANCH BK ;
DRAW 30 TREE
```

There are a couple of new words introduced here. The first is */ which performs a multiplication and division at the same time (in this example multiply by 3, divide by 4). This is Forth`s way of dealing with fractions. The net effect is the same as multiplying by three quarters. This is how the tree branches are made successively shorter. The second new word is RECURSE. This word allows TREE-BRANCH to call itself with a shorter branch size. The IF in the first line prevents the branches from getting too small.

Finally, let`s combine some of these shapes to make another picture (Plus/4 users enter BLACK BG CS WHITE INK first):

```
: SCENE GREEN PENCOLOR
PU 100 60 SETXY PD 40 TREE
PU 160 60 SETXY PD 15 TREE
PU 260 60 SETXY PD 25 TREE
YELLOW PENCOLOR
PU 290 190 SETXY PD
50 170 36 SHAPE ;
HIDETURTLE DRAW SCENE
```

Some new words are used to help draw the scene. PENCOLOR changes the colour of the lines drawn. It can be abbreviated to PC. SETXY sets the turtle position in absolute screen co-ordinates rather than relative distances. HIDETURTLE (or HT) removes the image of the turtle from the display. It can be made visible again with SHOWTURTLE (or ST). Hiding the turtle will speed up line drawing considerably.

We`re done with graphics for now so enter this line to get back to text mode:

```
NODRAW
```

Saving words to disk

Up to now, every new word we have defined has been typed directly into the console. The words we created are stored in the Forth dictionary in the computer`s memory but they would be lost if we turned the computer off. Unfortunately, Blazin` Forth doesn`t support saving words that are already compiled into the dictionary. So, how do we keep a permanent copy of our work?

Forth uses a concept of source code ‘screens` that map directly onto blocks on the disk. A screen contains 1024 bytes arranged in 16 rows of 64 columns each. While modern Forths have abandoned this system in favour of plain text files, it does have an advantage on the Commodore. The poor performance of the Commodore disk drives is legendary and saving even a modestly sized BASIC program takes an age. Forth`s block system allows you to save and load small pieces of code at any time rather than having to save and load the entire program. So if you have a large program, say 50 screens worth, and want to make a change to one screen, you only need to load that one screen, make the change, then save it again. Since a screen is only 1K in size, this is a relatively speedy process, even on a 1541!

Let`s have a go at making a screen. Make sure the tutorial disk is still in drive 8. The first thing we need to do in Blazin` Forth is enter the following word:

```
MOUNT
```

This prepares the disk system for use. It only needs to be entered once at the start of the session or if you change disks, or restart Forth. The activity light on the disk drive will stay on. This is normal.

There are a few screens pre-written to the tutorial disk. To show the contents of a screen enter this line:

```
51 LIST
```

This reads block 51 from the disk into memory and displays it. This screen contains the definitions of some convenience words such as CLS. Screens 52 through 56 contain some of the graphics words we created earlier. Try listing them to see their contents.

Notice the first line of each screen contains a comment prefixed with //. It`s conventional in Forth to list the names of the words defined on that screen. We can get a list of these comment lines for a range of screens using the INDEX word. To index the tutorial screens try this:

```
50 56 INDEX
```

Now let`s make a new screen. Enter this line:

```
1 LIST
```

The block is currently empty so the listing will just show line numbers. It`s very important when creating a new screen for the first time to clear it. Enter this command:

```
WIPE
```

This clears the current screen to spaces. You may think that this is unnecessary since the screen was already blank but new disk blocks are full of null (0) characters, not spaces. These cannot be seen but they would interfere with loading the screen back in. So always remember to WIPE a new screen before you use it.

Now, there are several ways to edit a screen in Forth. There isn`t space in this article to cover them all so I`ll show you the most useful, XEDIT. This is an extension to the usual Forth editor that takes advantage of the screen editor already built into the Commodore. Enter this line:

```
1 XEDIT
```

The empty screen is listed with line numbers followed by > signs. Now we can change the contents of the screen by simply overtyping the lines shown and pressing Return on each changed line.

Let`s put some code into this screen. We will use this screen to load in all the words defined in the tutorial screens 51 through 56. Move the cursor over the displayed lines and change them as shown below. Make sure to leave a space between the line number and its contents and to press Return on each changed line. Plus/4 users replace the 52 on line 2 with 53.

```
0> // LOAD TUTORIAL SCREENS
1> 51 LOAD
2> 52 LOAD
3> 54 56 THRU
```

After you have made the changes and pressed Return over each line, let`s just check the contents of the screen. Press Shift and Clr/Home and enter the command

```
L
```

This re-lists the current screen. If the lines don`t match those shown above, enter the command 1 XEDIT again and make the changes. We are now ready to save the screen. To force any changed screens to be written back to disk, use this word:

```
FLUSH
```

The screen we just edited is written to disk immediately. As a general rule of thumb, if you are editing several screens at once just let Forth deal with the disk access between screens. Forth will automatically write changed screens back to disk when it needs to make room for loading in new ones. Once you are done editing, always remember to issue the FLUSH command to make sure all the changes are written to disk.

Retrieving words from disk

Before we continue let`s reset Forth back to its initial state. To do this, issue the command:

```
RESTART
```

After a few seconds you will see the original start-up screen. Now let`s try loading in the screen we created. To do this, enter these lines:

```
MOUNT
1 LOAD
```

We need to use the MOUNT command again because we have restarted Forth, but we only need to do this once at the start of the session.

The LOAD word reads the contents of a screen from disk and runs through its contents as if the words had been entered in the console. After a few seconds of disk activity, Forth responds with OK. Now try entering the CLS word again. Success! The screen clears as expected.

The word THRU is used to load in several consecutive screens at the same time. In our tutorial screen, the command 54 56 THRU will load the contents of screens 54, 55 and 56 all in one go, displaying the screen numbers as it reads them.

If you want to experiment and save your work to the tutorial disk, please only use screens below number 50. Screens 50-80 have been reserved for tutorial content, and screens above that are used to store the Blazin` Forth programs so they are unavailable. You may also use a separate disk of your own but if you do, be careful not to use it for anything other than Forth. Mixing normal Commodore files and Forth screens on a disk will cause corruption.

We have finished with Forth for the time being. To leave the Forth environment enter:

```
BYE
```

The computer will reset back to BASIC.

Final words

That brings us to the end of this little insight into the Forth language. I hope you found it interesting and that it encourages you to explore the language further. If you would like to see a regular Forth feature in the magazine, write in and let us know.

The tutorial disk can be downloaded here:
http://www.commodorefree.com/tools/forth/bforth.zip

The online edition of Starting Forth can be found at
http://www.forth.com/starting-forth/

The original Blazin` Forth archives for the C64 and Plus/4 can be found on the Forth Interest Group FTP site at
ftp://ftp.forth.org/pub/Forth/Compilers/native/misc/commodore64/
(be aware that this site can take a long time to respond, but it will connect eventually!)

COMMODORE FREE
Paul Davis has said he may continue this introductory course if there is enough interest from users, (I certainly would like to know a little more about this language) feel free to email me and let me know what you think, I will pass on you comments (with email addresses removed if you prefer) to Paul for him to "think about it"

```
** * * * * * * * * * * * * **
*                             *
* Blazin' Forth for CBM-64    *
*                             *
*   System Documentation      *
*                             *
** * * * * * * * * * * * * * **
```

This documentation and the software it describes are Copyright (C) 1985 by Scott Ballantyne

Distribution on a not for profit basis is encouraged. Sale or Resale of this manual or software is not allowed.

I would like to acknowledge the following people, who contributed time, support or knowledge to this effort:

Glen Haydon, author of MVP-FORTH, whose book ALL ABOUT FORTH was my constant companion during the early stages of this project.

Henry Laxen, Forth Wizard and writer. Many of the best features of this compiler are due to his ideas, and his columns in Forth Dimensions are among the best and most creative writing on Forth I have ever seen. In particular, Laxen is the creator of the DEFER IS concept.

To RMS and WRG, wherever you are, thanks for everything you taught me.

Thanks to Chris M. A friend indeed.

Special thanks to BJ, who knows what she did.

Thanks to all those who put up with various versions of this compiler, and suffered almost as much as I did while debugging it.

CBM C64 C128 and Commodore-64 are trademarks of Commodore Business Machines.

This software is dedicated to the memory of Leonard Rose.

Commodore Computer Club U.K. Membership form

Membership forms for the Commodore Computer Club (UK) (CCC(UK)).If you are filling in this form electronically (using a word processor), then please send it to shop@CommodoreComputerClub.co.uk stating that the subject is CCC (UK) membership. Call the file CCC-XXXXXXXXXXXXX when saving - replace the Xs with your name. We will assign you with a membership number. For those people who have printed out this application form to fill in, please send it with a cheque payable to 'Commodore Computer Club', to: Commodore Computer Club Treasurer, 2 Willis Road, Blackburn, Lancashire, BB2 2UA - United Kingdom. Fees can be arranged electronically (via PayPal), or by personal cheque or postal order. Once you have filled in and submitted this membership form, an invoice will follow with relevant payment details, so please don't forget to fill in your contact details. Please sign me up for a membership to the CCC (UK)

I wish to be a member for:
[ ] 6 months* at 3 GBP;
[ ] 1 year* at 5 GBP
[ ] life member at 30 GBP

My personal details:
**Name:

**Address:

**Postal code:

Country (if outside of the United Kingdom)**:

**Date of application (Please use dd/mm/yy format):
[FOR INTERNAL USE: Date membership fee received:]

**PayPal account (for those paying by this method): If you have already registered on the CCC (UK) forums (http://www.commodorecomputerclub.co.uk/forums), please tell us your username here:
Email address: Please tick or cross the box below once you have read and understood the club's rules and regulations:**[ ] I declare that I, the named applicant above, have read and understood the CCC (UK) rules and regulations, and agree to abide by them fully and co-operatively. I understand that I am joining this club on a personal level, and not as a representative of any group, developer, publisher or vendor that I belong to. Please tick which
computers you own:

[ ] CBM/PET 40 columns	[ ] CBM/PET 80 columns	[ ] VIC/VC 20	[ ] C64/64c	[ ] C64GS
[ ] SX-64&	[ ] C16/116	[ ] Plus/4&c16	[ ] C128/128D	[ ] C64DTV
[ ] Other Commodore 8-bit	[ ] Other Commodore 16-bit			

If you have ticked 'Other Commodore 8-bit' or 'Other Commodore 16-bit', please list these machines below:
Please tick from the list below your interests from the following:

[ ] Gaming	[ ] BASIC programming	[ ] Machine language coding	[ ] GEOS
[ ] JOS/WiNGs	[ ] Tech/scene demos	[ ] Collecting	[ ] Archiving/preservation
[ ] Other applications			

If you have ticked 'Other applications', please give details below:
Please tick if you use any of the following peripherals below:

[ ] Datasette	[ ] 1541 compatible disk drive	[ ] 1581 compatible disk drive	[ ] FD2000/4000
[ ] CMD HD or RAMLink	[ ] 1351 mouse or compatible		
[ ] Commodore REU	[ ] Other RAM expansion		
[ ] SuperCPU 64/128	[ ] Other accelerator		
[ ] MMC/Retro Replay	[ ] SwiftLINK/Turbo232	[ ] RR-Net or FB-Net	[ ] Other networking device
[ ] 1541Ultimate/+	[ ] Action Replay	[ ] Trilogic Expert	[ ] Other cartridge upgrade
[ ] Other speed loader			

[ ] Commodore VDU
[ ] Other hardware Please list any items not mentioned above which you have (especially for other Commodore 8-bit machines):

COMMODORE COMPUTER CLUB (UK) RULES AND REGULATIONS

These rules were agreed on Saturday 26th July 2008 at our first meeting held at Blackburn, Lancashire. They cover all rules of the running of the Commodore Computer Club (UK), herein referred to as 'the club', and also regarding complaints made to the club, and membership of the club.

(1) Membership subscriptions, raising funds and re-selling items.
(a) All members will pay a membership fee as follows: 3 GBP for six months, 5 GBP per year and life membership at 30 GBP. This fee will entitle the holder to free entry to the meetings, and special limited areas of the website, such as private forums and exclusive downloads, should we get anything exclusive to download. Membership will always be back-dated to the start of the month in which the member took out the subscription, so that everyone joining in the month of April for one year will see their membership expire on the 31st of March the following year.

(b) We should have a 'Commodore Computer Club Shop', which will stock all of the latest hardware mods and sods for Commodore computers where possible. To stop the 'Maurice Randall' effect, in which the club will have to repay people for not receiving their goods because they haven't been delivered but have been paid for, items will only be on sale if they are in stock.

(c) There will be two prices, one for members (cost of item + postage and packing + 10%), and one for none members (cost + postage and packing + 20%).

(d) Any members that do work for the club, organising events, donating items for auction, coding, or are otherwise active, with exception to posting on forums and turning up to meetings, will be considered for free membership and/or lifetime membership on merit based only on work they have done for the club. In certain instances, will include what they have contributed overall to Commodore computing or gaming during their life-time, should any 'Commodore legends' show sufficient interest to join the club.

(2) Events, software and other developments.
(a) Any money that is raised by the club should be used primarily for setting up events, or bolting onto other events as appropriate This is to go towards, or cover costs of van hire, hotels, and food and drink, so the person or people who are willing to travelling to these events, man stalls and generally promote the club and its work are not be out of pocket as far as possible.

(b) Profits made from items sold at events should contribute to cover the costs of attending, or hosting, and/or expenses accrued during the event. This will not include monies raised from membership subscriptions paid for during the event.

(c) The club should also seek to raise money for the purpose of developing hardware and/or software that will benefit Commodore users in the UK and world-wide, and such items could therefore be sold through the club Shop.

(d) Payments to developers who are commissioned to work on behalf of the club should not be made in advanced or up front unless otherwise agreed by the treasurer and chair-person, and any other two members. This should be openly discussed with all members either in private members areas of the site, or at an organised meeting as appropriate.

(e) Hardware that is commissioned on behalf of the club which reaches production should be sold at a small profit, and monies raised to put back into the clubs funds.

(f) If it is agreed that the club should commission entertainment software, the productions should be available to download for free from the site for members only. Real-media versions should also be sold through the shop with non-members able to buy copies, though at a higher price than members.

(g) Any software commissioned by the club will either be purchased outright, paying the programmer an agreed fee on completion, or paying a lesser fee and splitting the profits at an agreed rate. This should be discussed on a case-by-case basis. The chair-person and treasurer, and two other members, must agree which method should be implemented.

(3) Meetings and monies.
(a) The club should hold an annual general meeting in which members have a say in its running, and are able to make suggestions and table official club business for the year ahead. Membership subscriptions should be reviewed at the annual general meeting, and any price increases must be agreed by the chair-person, treasurer and at least two other members.

(b) There should be an annual audit of the clubs finances, with a news-letter at least every three months. The audit should be published before the annual general meeting, and this and the news-letter should be available to current members online in the private member areas. Former members may request this information, which will be granted on a case-by-case basis.

(c) Members will be able to attend any events that organised and run by the club for free when ever possible, whilst non-members will pay a small signing in fee of at least £2.50. With agreement with other event organisers, and at events that the club is attending in an official capacity, we will work towards getting members a discounted entry fee.

(d) All monies raised will go back into club funds.

(4) End of line.
(a) If it is apparent that the club is not running within its means to the extent that it is likely to fold, or that legal action against it will lead to the club being dissolved, all club assets should be sold or auctioned off, the monies raised pooled and members will be refunded their current subscriptions based on the length of time they have been members. The longest-serving paying members will be refunded first as appropriate, either partially or fully depending on the financial circumstances at the time. The newer members will be dealt with last.

(b) Personal donations to the club's funds can never be fully refunded, and are not guaranteed to be paid back at all depending on the circumstances.

Club complaints procedure:
Phase 1: Where a complaint is made against the club, or one of the club members, there should initially be a private apology between the club or individual and the plaintiff. This apology should be for 'any undue harm or upset caused', and will not amount to an admission of guilt or a retraction in any way. The club will not be able to force any of its members to make this initial apology except in the instance that the individual has clearly and admittedly worked on the clubs behalf in the matter specifically relating to the complaint that has been lodged.

Phase 2: The matter should then be investigated to establish the facts. If it is deemed that an individual club member has not been acting on the clubs behalf with regards to the specifics of the complaint, then this becomes a personal matter between the two parties. The club should therefore stop any further investigations or involvement in the matter.

Phase 3: If the complaint lacks any real evidence, or it is felt that the findings are not conclusive, then the matter should be closed. Neither the club, nor any of its members, should therefore discuss the matter publicly. All findings should be reported to the plaintiff, and the matter should be considered closed from the club's point of view.

Phase 4: Where a complaint is upheld, a public apology and/or retraction should be published through the official website, and in the newsletter. The club should also give the plaintiff the opportunity to give his or her point of view through the website and/or newsletter as appropriate In this instance, the case will be considered closed from the club's point of view unless the plaintiff wants to take the matter further through due legal process.

Emergency phase: If at any point during this process the plaintiff feels aggrieved to the extent that he or she instructs a solicitor to take the matter up against the club or club members who have clearly being acting on behalf of the club in this instance, the club should then consider its legal position on the matter, and a meeting should be set up with the principle members of the club within two weeks of receiving legal notice to discuss the matter, and what to do next. Obviously, one would hope that any complaint would ever get to this stage.

Membership:
People who join the club will have a personal membership to it. They may not join the club as a company, publisher or software distributor or hardware vendor.

Newsletter and reviews:
The Commodore Computer Club (UK) is an independent user group which will review and stock all appropriate wares. We will do so on merit only, and invite all members to have their say about any literature published through the newsletter or any reviews written on behalf of the club. We will invite hardware and software vendors and publishers to have their say on reviews written, and we will publish their comments through the newsletter.