Euro 5.33 USD 6.17

8/1999

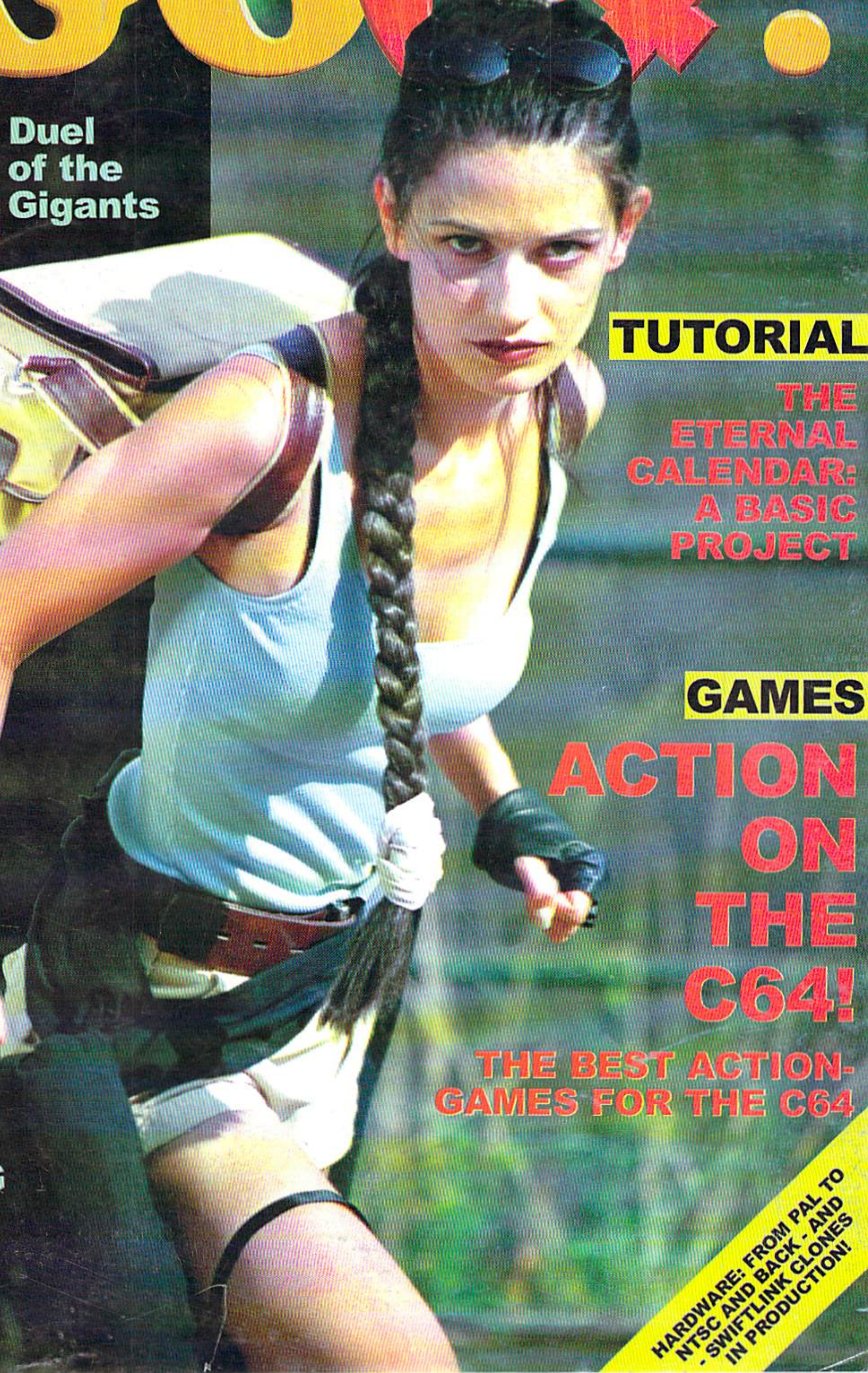
The Magazine for true Computerfans

GO64!

CW

GO64! commodore WORLD

Duel of the Gigants



MP3 versus Wheels

Which GEOS-Update is better? faster? most compatible?

TUTORIAL

THE ETERNAL CALENDAR: A BASIC PROJECT

GAMES

ACTION ON THE C64!

THE BEST ACTION-GAMES FOR THE C64

NEW SCPU OPERATING SYSTEM

What can the Unix-like JOS do?

HARDWARE: FROM PAL TO NTSC AND BACK - AND - SWIFTLINK CLONES IN PRODUCTION!

CMD • HARDWARE • ORDERS: 1-800-638-3263 • SOFTWARE • CMD

FLOPPY DISK DRIVES

CBM 1541C Disk Drive (New w/JD)	$119.00
CBM 1541-II Disk Drive (New w/JD)	$129.00
CBM 1571 Disk Drive (Remanufactured w/JD)	CALL
CMD FD-2000 (800K and 1.6 MB)	$179.00
CMD FD Real-Time-Clock Option	$39.00
Box of 10, High Density Disks (1.6MB)	$14.95
Box of 10, Enhanced Density Disks (3.2 MB)	$29.00

HARD DRIVES

HD-KIT (Controller, Case, PS, Manual, Serial Cbl.)	$249.00
HD-1000, 1 GB (Special Edition)	$449.00
HD-2000, 2 GB (Special Edition)	$549.00
HD-3000, 3 GB (Special Edition)	$669.00
HD-4000, 4 GB (Special Edition)	$749.00

MISCELLANEOUS HARDWARE

Monitors (40 column and 40/80 column)	CALL
Aprotek 3-Way User Port Expander	$30.00
Aprotek User Port Extension Cable	$19.00
C-64 Computer (Remanufactured, w/JD & PS)	$99.00
C-64C Computer (Remanufactured, w/JD & PS)	$109.00
C-64C Computer (New)	$139.00
SX-64 Computer (Remanufactured w/JD)	$299.00
C-128 Computers	CALL
C128-D Computers (Remanufactured w/JD)	CALL
C-64/64C PowerSupply (New, 1.7 Amp.)	$29.00
C-64/64C RepairablePS (New, 1.7 Amp.)	$39.00
C-64/64C Heavy Duty Repairable PS (New, CMD)	$64.95
C-128 RepairablePower Supply (New, CMD)	$64.95
C-128D Internal Power Supply (New)	$39.00
1541-II/1581/RLv1 Power Supply (New, 1 Amp.)	$25.00
CMD EX2+1 3-Port Cartridge Port Expander	$39.00
CMD EX3 3-Port Cartridge Port Expander	$34.00
CMD GeoCable II Print Cable 6ft./15ft.	$34.96/$39.95
CMD Sega-style Gamepad for C-64/128	$24.95
CMD SmartMouse (1351 Compatible Mouse)	$49.95
Flip'n'File 25 Disk Holder (5.25-inch.)	$5.95
Floppy Disks (25 pack of DSDD 5.25-inch.)	$10.00
Inkwell Light Pen Model 170C	$75.00
Monitor Cables	CALL
Cannon BJC-4400 Color Inkjet Printer (Refurb)	$179.00
Panasonic KXP-1150 9-pin Printer	$199.00
Panasonic KXP-2130 24-pin B/W Printer (Refurb)	$199.00
Panasonic KXP-2130 24-pin Color (Refurb)	$299.00
Panasonic KXP-2130 24-pin Printer (B/W)	$329.00
Panasonic KXP-2130 24-pin Color Printer	$429.00
Smartline C-600 Printer Interface	$10.00

UTILITIES

Big Blue Reader V4.10 (SOGWAP)	$39.00
CMD Utilities	$24.95
JiffyMON-64 (ML Monitor)	$19.95

LANGUAGES & COMPILERS

BASIC 64 Compiler (Abacus)	$17.00
BASIC 128 Compiler (Abacus)	$25.00
Blitz! 64 Compiler (Skyles)	$30.00
Buddy 64/128 Assembler	$39.00
Cobol 64 (Abacus)	$17.00
Pascal 64 (Abacus)	$17.00
Power C (Spinnaker)	$19.00

GEOS

Collette Utilities (Handy Geos Utilities)	$19.95
Desk Pack Plus	$29.00
Dweezils Greatest Hits (NewTools2,Stamp,Label128)	$30.00
FONTPACK Plus	$25.00
gateWay 64 or 128 (Specify Version)	$29.95
geoBASIC	$20.00
geoCalc 64/128	$40.00/$45.00
geoChart	$29.00
geoFAX	$39.95
geoFile 64/128	$40.00/$45.00
geoMakeBoot (Makes Bootable copies)	$12.95
geoProgrammer	$45.00
geoPublish	$40.00
GEOS 64 v2.0	$44.00
GEOS 128 v2.0	$49.00
geoSHELL V2.2 (CLI for GEOS)	$24.95
International FONTPACK	$25.00
Perfect Print LQ for GEOS (Laser-like output)	$49.95
RUN GEOS Companion	$20.00
RUN GEOS Power Pak I or II (Specify)	$20.00
Wheels 64 (GEOS Upgrade, req. GEOS 64 v2.0)	$36.00

MIDI & SOUND

Digimaster	$34.95
SID Symphony Stereo Cartridge	$44.95
Dr.T's Caged Artist or AlgorythmicComposer	$15.00

EDUCATIONAL

Adventures w/ Fractions (ages 10-14)	$15.00
Expeditions (ages 8-15)	$15.00
Pre-Reading (ages 3-7)	$15.00
Spelling Bee (ages 8-13)	$15.00
The Glass Computer (ages 12+)	$15.00
The Market Place (ages 8-14)	$15.00
Where in Europe is Carmen SanDiego?	$29.00
Where in the World is Carmen SanDiego?	$29.00

ACCELERATORS

SuperCPU 64	$179.00
SuperCPU 64 w/SuperRAM (no RAM installed)	$219.00
SuperCPU 64 w/SuperRAM (4 MB)	$254.00
SuperCPU 64 w/SuperRAM (8 MB)	$278.00
SuperCPU 64 w/SuperRAM (16 MB)	$316.00
SuperCPU 128	$239.00
SuperCPU 128 w/SuperRAM (no RAM installed)	$279.00
SuperCPU 128 w/SuperRAM (4 MB)	$314.00
SuperCPU 128 w/SuperRAM (8 MB)	$338.00
SuperCPU 128 w/SuperRAM (16 MB)	$378.00
SuperRAM Card (no RAM installed)	$79.00
SuperRAM Card (4 MB)	$114.00
SuperRAM Card (8 MB)	$139.00
SuperRAM Card (16 MB)	$179.00

RAM DEVICES

RAMLink Base Model (ØMB, No RAMCard)	$169.00
RAMLink w/1MB RAMCard (Special!)	$209.00
RAMLink w/4MB RAMCard (Special!)	$249.00
RAMLink w/16MB RAMCard (Special!)	$369.00
RAMCard RTC Option/After Market Kit	$10.00/$19.00
RAMLink Battery Back-up (Optional)	$24.95
Parallel Cable (RAMLink to HD)	$19.95

PRODUCTIVITY

Bank Street Writer	$12.00
Cadpak 64 (Abacus)	$22.00
Cadpak 128 (Abacus)	$25.00
Chartpak 64 (Abacus)	$17.00
Chartpak 128 (Abacus)	$25.00
Data Manager 64 (Timeworks)	$16.00
GoDot	$34.95
I Paint v1.5 (128, 80-col., 64K VDC)	$39.00
I Port v1.54 (128, 80-col., 64K VDC))	$29.00
Personal Portfolio Manager (Abacus)	$16.00
RUN Productivity Pak I, II, or III (Specify)	$15.00
RUN Super Starter Pak 1541 or 1581	$20.00
RUN Works	$20.00
SEC Check Register 128	$50.00
SuperScript 128 (Precision)	$20.00
Suberbase 64 Version 3.01 (Precision)	$35.00
Superbase 128 Version 3.01 (Precision)	$35.00
SwiftCalc 64 (Timeworks)	$16.00
TWS 64 w/Speller (Busy Bee)	$29.00
TWS 128 w/Speller (Busy Bee)	$39.00
TWS Modules (HD/RL/Illustrator)	each $5.00

TELECOMMUNICATIONS

Aprotek Com-Modem Adapter (for ext. modem)	$19.00
Aprotek Com-Modem Adapter Cable	$9.95
Commodore 1670 Modem (1200 baud)	$19.95
Modem (9-25) or Null-Modem Cable (9-9 or 9-25)	$9.95
Novaterm 9.6 (specify 3.5" or 5.25")	$29.95
Turbo232 Cartridge (Up to 115Kbps)	$39.95
USR Sportster 33.6Kbps FaxModem	$169.00
USR Sportster Faxmodem w/Turbo232 & Cable	$199.00

GAMES

Atomino	$17.00
Ballistix	$13.00
Chomp!	$10.00
Cut Throats!	$10.00
Day in the Life of Prehistoric Man	$19.00
Escape Route	$19.00
Felony	$10.00
Frogger	$10.00
Grand Prix Circuit	$10.00
Guerilla	$10.00
Hardball	$10.00
Heavenbound	$19.95
Island of the Dragon	$19.00
Jordan vs. Bird: One on One	$10.00
Kings of the Beach	$10.00
Lions of the Universe	$19.95
Mainframe	$13.00
Mean Streets	$10.00
Menace	$15.00
RUN C64 Gamepak or C128 Funpak (Specify)	$10.00
Skate or Die	$13.00
The President Is Missing!	$10.00
The Three Stooges	$10.00
Tie Break Tennis	$16.00
Total Eclipse	$10.00
Walkerz	$19.00
War in Middle Earth	$10.00
Wings of Circe	$19.00

BOOKS

Anatomy of the 1541	$8.00
C-64 Science & Engineering	$8.00
C128 Computer Aided Design	$8.00
C128 BASIC Training Guide	$8.00
Commodore 64 Tricks and Tips	$8.00
Compiler Design & Implementation for the C64	$8.00
GEOS Programmers Reference Guide	$28.00
Graphics Book for the C-64	$8.00
Hitchhikers Guide to GEOS	$28.00
Ideas for Use on Your C-64	$8.00
Printer Book for the C64	$8.00
Mapping the C64	$8.00

••• All Major Credit Cards Accepted •••

Mastercard • Visa • Discover • American Express

Creative Micro Designs, Inc.

P.O. Box 646
East Longmeadow, MA 01028
Visit our Web Site at http://www.cmdweb.com/

Info: (413) 525-0023
Fax: (413) 525-0147

Wheels

THE HOTTEST UPGRADE EVER CREATED FOR GEOS

If you've been waiting for GEOS to get better, faster and more efficient, your wait is over! Click Here Software's *Wheels* takes GEOS to the next level, with full integrated support for all the latest hardware. *Wheels* also adds many features not previously available in GEOS, as well as some that could only be had by buying or adding other utilities and patches. GEOS is really going someplace, and it has the *Wheels* to get there!

Wheels64 **$36.00** *Wheels128* **$40.00**

Minimum requirements for Wheels: GEOS 64 v2.0 or GEOS 128 v2.0, 1541 or 1571 disk drive, mouse or joystick, and 128K or larger RAM expander.

Novaterm

version 9.6

Join the Telecommunications Revolution... get on the Information Super Highway TODAY!

- **Easy-to-use — novice user configuration**
- Now supports **UUencode and UUdecode** and converts files on disk or in a buffer. Also ASCII <> PETSCII and PETSCII to Unix.
- Protocal support: **Zmodem up/download, resume (crash recovery), streaming to buffer**; Ymodem batch; **Ymodem-g**; Xmodem-1k; **Xmodem-1k-g**(to buffer); Xmodem-CRC; Punter; Kermit; WXmodem.
- Supports High Speed Interfaces like Turbo232 for speeds up to 230Kbps.
- Supports ANSI color/graphics, VT102, and VT52 in 80 col. Plus Commodore color graphics and ANSI color. **All emulation modules now load from one window.**

Just $29.95
(Specify 5.25" or 3.5" disk)

- New faster 80 column modes: **Soft-80 on C-64 (enhanced scrolling w/ REU); and 25- or 28-line C-128 VDC 80-col. mode**.
- **Use any memory device as a buffer** including: 17xx REU, RAMLink partition, GeoRAM, C128 VDC, BBGRAM, and internal C64 memory.
- New text capture features:Capture text on-line and store in any device; High-speed transfers directly into memory and now **retain buffer contents with battery/power backed RAM devices such as RAMLink**.
- Enhanced full featured **Text editor can now load/save files from the buffer** and has an integrated script compiler.
- Script language for automatic operations
- Simple BBS mode - dial-in downloading
- Supports **real-time clocks in CMD devices**
- **All new user's manual**

JiffyDOS

Increase Disk Access Speed by up to 1500% While Retaining 100% Compatibility!

JiffyDOS 64 System **$39.00**
Includes JiffyDOS C64 Computer ROM and one JiffyDOS drive ROM.

JiffyDOS 64c System **$39.00**
Includes JiffyDOS C64c Computer ROM and one JiffyDOS drive ROM.

JiffyDOS SX64 System **$49.00**
Includes JiffyDOS SX64 Computer ROM and SX1541 drive ROM.

JiffyDOS 128 System **$49.00**
Includes JiffyDOS C128 Computer ROMs and one JiffyDOS drive ROM.

JiffyDOS 128D System **$49.00**
Includes JiffyDOS C128D Computer ROM and 1571D drive ROM.

JiffyDOS Drive ROMs **$19.00**
Available for Commodore 1541, 1571, 1581 and most compatibles.

Note: Computer Serial Number Required for C-64/64C Orders

Shipping and Handling Charges

	Continental US		AK,HI,PR	Canada	Others
Order Subtotal	UPS Ground	2nd Day	UPS 2nd Day Air	Air Parcel Post	AirMail or UPS
$0.01 to $19.99	$4.75	CALL	$16.00*	$5.00	CALL
$20.00 to $29.99	$5.75	CALL	$17.50*	$7.00	CALL
$30.00 to $59.99	$6.75	CALL	$21.00*	$9.00	CALL
$60.00 to $149.99	$7.75	CALL	$29.00*	$12.00	CALL
$150.00 to $299.99	$11.00	CALL	$38.00*	$20.00	CALL
$300.00 to $799.99	$16.00	CALL	$42.00*	$25.00	CALL
$800.00 and up	$21.00	CALL	$55.00*	$38.50	CALL

UPS C.O.D. add $5.00 (US only)

* Other shipping methods available. Call for details.

POLICIES: CMD accepts: Major Credit Cards, Money Orders, COD and Personal Checks. Personal Checks are held 3 weeks. Customer is responsible for shipping charges on refused packages. Most items are stock, contact CMD for firm delivery. Returns for merchandise credit only within 30 days with prior authorization. Credits are less shipping, handling, taxes, customs, duties, and a 15% restocking charge. No refunds or credits on opened software. All prices and specifications are subject to change without notice. CMD Business Hours: (Orders) M-F, 9AM-5:30PM EST (Tech Support) M-F, 1PM-5:30 PM EST.

CONTENTS

- August 1999 -

6
JOS

```
Welcome to J-Shell V0.1
Please type one of the following
commands:
[illegible]
Welcome to J-Shell V0.1
Please type one of the following
commands:
[illegible]
```

We have another new operating system for the Super-CPU, but this one's different. It uses the same basic concept as another big time OS, Unix. More about this on page 6!

Picture: Stefan Kluge

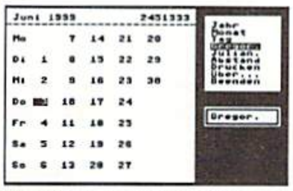
9
System Comparison

Almost simultaneously, the two new GEOS patches, Wheels and MegaPatch 3, have been released. But which one is better? We'll compare the strengths and weaknesses of both programs and let you know. Page 9

18
Action on the C64!

Picture: Bernhard Pehall (www.lichtbogen.at)

Because the C64 has become so popular because of games, we'll focus heavily on these. The historical development from Galaga to Katakis as well as the technical milestones of the C64 can be found further explained on page 18!



```
Juni 1999                 2451333
-----------------------------------
Mo       7   14   21   28
Di   1   8   15   22   29
Mi   2   9   16   23   30
Do   3  10   17   24
Fr   4  11   18   25
Sa   5  12   19   26
So   6  13   20   27
```

23
Eternal Calendar

How do you program an eternal calendar in BASIC? The answer, together with necessary mathematical knowhow is given to you by Arndt Dettke - BASIC pro through and through. See our continuing column "A BASIC Project" on page 22!

[news]

[geos]

[coverdisk]

[hardware]

[games]

[tutorials]

[tips & tricks]

[departments]

A

Datapump
Gigacad Patch
Mouse-Anatomy Files
Zynax

B

MP64 Patch
MP128 Patch
GEOS Speed Comparison
Traffic

Have you noticed?

Commodore World magazine has a new look and it is with pleasure and excitement that we announce these now changes which we feel will better serve you.

As most of you are aware, the publisher of Commodore World magazine is Creative Micro Designs, Inc., the devoloper and manufacturer of many of the finest hardware accessories ever developed for the Commodore 64/128. Although publishing is not CMD's area of expertise, Commodore World was undertaken as a result of our desire to provide a Commodore specific 64/128 magazine after the demise of other such publications. Despite our best intentions, we have not been able to deliver the type of service that we felt you deserve. It is therefore our pleasure to announce that we have found a solution to address these shortcomings after many months of searching.

Effective with this issue, Commodore World will be produced by CSW Verlag, publishers of GO64! and will become known as GO64!/Commodore World. The magazine will be produced in the English language and all outstanding issues due to Commodore World magazine subscribers will be provided in this new format. CSW Verlag has shown us that they have the ability and expertise to provide you with a quality magazine dedicated to the Commodore 64/128. With their proven track record and staff dedicated to the full time publishing of a Commodore specific magazine, Commodore World subscribers will benefit greatly. The magazine will now be received with greater frequency (once a month) and will have larger pages, a glossy cover, spot color on inside pages and will also be provided with a cover disk.

While all of us here at CMD will miss the opportunity to bring our personal touch to Commodore World magazine, we are convinced, that this is in the best interests of our loyal readers. High quality, better frequency and a broader base upon which to report on Commodore news and interests, coupled with our continued resolve to provide reliable Commodore hardware and software products, should benefit the Commodore user. We wish to thank you for your loyal following over the past five years and again assure you that the changes that have taken place have been done so with our readers best interests in mind.

From all of us at CMD, we sincerely believe that you will find the GO64!/Commodore World magazine to your liking and trust that you will continue to support these efforts as you have in the past. CMD wishes to assure each and every one of you that we are not disappearing from the scene and that we will be continuing our efforts to provide quality hardware, software and repair services well into the new millennium.

Truly yours,
All of us at Creative Micro Designs, Inc.

Dear Readers,

this GO64! issue is quite special. It's not only our 30th issue since our premier in March of 1997, it's also the first issue which will be published in English as well as German. A dream of the GO64! team has been fullfilled, because GO64! has gone from a German-only magazine to an international medium.

We'd especially like to greet our new English-speaking customers. Many Commodore World subscribers were somewhat astounded by the appearance of this magazine, as it doesn't look like a typical issue of Commodore World. Through cooperation with Creative Micro Designs, Inc., CSW-Verlag has been given the commission to keep Commodore World alive within GO64!. So this means that Commodore World subscribers will now get the English version of GO64!/Commodore World, which has the same contents as the German version.

Surely you must all have an infinite number of questions. Firstly, I'd like to let our German speaking customers know that nothing will change for them. For the English speaking customers who have Commodore World subscriptions, the following applies: All remaining issues of Commodore World which you have paid for will be replaced by issues of GO64!/Commodore World. If you have three issues remaining on your subscription, you will receive three issues of GO64!/Commodore World. If we don't hear anything from you when your subscription runs out (you can respond by email, fax, letter or phone call) your subscription will be renewed for five months (five issues). If you like GO64!/ Commodore World magazine and it's the end of your subscription, (that is, after the five month extension) and if you still haven't replied to us, your subscription will be automatically renewed for one year (twelve issues), and the same conditions will apply to you as to our German-speaking subscribers.

GO64! itself is made with more expenditures than Commodore World. GO64! appears monthly, is partly in color, has more pages and a coverdisk is included, which contains contemporary software. If you calculate the shipping cost from Germany to other contries, you can see that the original price of 8 issues of Commodore World is no longer valid. Our subscription prices are as follows:

3 month trial subscription: US $19.95 (US $6.65 per issue)
5 month subscription (only available for Commodore World subscribers): US $29.95 (US $5.99 per issue)
12 month subscription: US $74 (US $6.16 per issue)

Additionally, we're offering customers with soon-to-expire Commodore World subscriptions a reduced yearly price of:

12 month privilege subscription: US $69 (US $5.75 per issue)

This price is only valid for Commodore World subscribers who subscribe for a year subscription before the October 31 cutoff date. As an incentive, the first ten subscribers to this option will receive a FREE diskette, full of the best software from the last two years of GO64! coverdisks.

All subscription prices include shipping costs, no matter where you are.

Through our international expansion we hope to bring the English-speaking and German-speaking communities together. We'd like to invite all our subscribers, old and new, German- and English-speaking alike to let us know your opinions, give us your feedback, submit articles as guest authors, write programs, contact other users, work on projects together and other activities to keep the C64 communitiy alive into the next millennium.

We're always looking for additional people to help us out with the English language version of GO64!. If you'd like to apply, write to the editorial department (see the address in the editorial section) or email us at: **go64@c64.org**

Günther Bauer,
for the GO64! team

News Flash

The Error Gremlin

The Error Gremlin has struck again. The Amica Paint color drives on disk 4/99 were incomplete. Additionally many copies of the 7/1999 disk are incomplete, as they have identical first and second sides. The GEOS-Adventure was not included as the full version. CSW is offering free replacement and rectification, Just ask. The game "Traffic", which was on disk 5/99 was defective, and can be found on this disk

UK8 Get-Together (England)

On Sunday, September 26, the UK8 Get Together will take place in Walsall, West Midlands, UK for all users of all 8-bit system platforms. Entrance fee is 5 pounds. For further information please contact via e-mail:
uk8@spheroid.demon.co.uk
or
Brian Watson, UK8 Publicity
Email: brian@spheroid.demon.co.uk

Chrome Mag for GEOS

Tom Gosser announces a new disk magazine for GEOS and Wheels called Chrome Mag. In it there will be articles by Maurice Randall, Bo Zimmermann, Cindy Simms, and other important people in GEOS. Each issue is supposed to come with new and well-chosen demos of the latest GEOS programs, clip arts, p.d., shareware, and even original versions of programs. The Chrome Mag Engine was designed by Bo Zimmermann (geoBrowser and geoBEAP). There will be four issues each year beginning on May 1, 1999. For the time being, the price will be $25 US ($30 US for international dispatch). Send cheques or money orders to
T. Gosser Publications, 955 Washington, St.
Douglas, WY 82633, USA
Email: combug@delphi.com

C64/128 Hard & Software

GRÄF Hard & Soft - Im Osterfeld 13 - 56462 Höhn-Neuhochstein

Telefon/Fax : 02661-40338 von 10:00 Uhr bis 20:00 Uhr

Internet : www.graef-hard-soft.de (Komplettliste & Bilder/Infos zu Hardware/Software)

Für unser Gesamtangebot : Kostenlose Gesamtliste anfordern oder Homepage im Internet besuchen

Wir bieten ständig weit über 700 verschiedene C64 Programme auf Diskette , Kassette oder Cartridge - darunter eine grosse Anzahl klassischer Spiele , Oldies aus den ersten Jahren , Spiele von Parker , Activision, Electronic Arts,SSI,Infocom,Magnetic Scrolls u.a.Auch Software für C16/Plus 4 bzw. manchmal VC20 vorhanden ! Hardware für C64 : Wir bieten Rechner , Floppies , Bastlerware und Einzelteile und diverses Zubehör zu guten Preisen.
Versand : Vorkasse (+7 DM) oder Nachnahme (+15 DM) ; Ausland nur Vorkasse (+20 DM)

Staffelpreise bei Mehrabnahme möglich ! GRATIS ab 20 DM Bestellwert : Tape für 1.- nach Wahl , ab 100 DM Tapes/Disk bis insgesamt 5 DM

Steckmodule für C64 : (Neuware)

Je 40 DM : (Activision , originalverschweißt)
Pitfall , Pitfall 2 , H.E.R.O. , Decathlon,
Beamrider , Zenji

Je 35 DM : Toy Bizarre , Zone Ranger ,
Solar Fox ,Spitball,Terminator 2

Je 30 DM : Q-Bert,Blueprint

Je 20 DM : Ms Pacman, Centipede,
Galaxian,Jungle Hunt

Viele weitere Module auf Anfrage !!!

C64 Spiele/Programme auf Diskette :

Rampart (Burgenschlacht) : 20 DM
Thai Boxing (Kampfsport),C128 !! : 15 DM
Tai Pan (Action-Adventure) : 15 DM
Strider 2 (Action) : 15 DM
Zork II (Adventure),Commodore : 20 DM
Zork III (Adventure),Commodore : 20 DM
Cosmic Causeway (Action) : 15 DM
Hunt for Red Octobre (Action) : 15 DM
Highway Encounter (Action) : 15 DM
Trivial Pursuit (Frage/Antwortspiel) : 20 DM
Breakdance (Action) : 15 DM
Superman vs. Darkseid (Action) : 15 DM
Super Pacman (Action) : 20 DM
Turtles - The Coin Op (Action) : 20 DM
Bad Cat (Action) : 15 DM
Nuclear Embargo : 10 DM
Let´s make Greeting Cards : 15 DM
Let´s make Signs & Banners : 15 DM
Let´s make Calendars : 20 DM
Art Library 1 & 2 (zu Let´s make ...) : 5 DM
50 Great Games : 15 DM
Big 4 Vol. II : 10 DM
(Sigma 7,Deep Strike,Thanatos,Saboteur 2)
Five Star Games : 15 DM
(Spindizzy,Batalyx,Zoids,Equinox,Scarabäus)
Alien Destruction Set : (Death or Glory,
Thundercross,Vengeance,Discovery) : 20 DM
Avenger (Action) : 15 DM
Corporation (Strategie) : 15 DM
Dark Castle (Action) : 15 DM
Double Take (Action) : 15 DM
Darkside (Action-Adventure , 3D) : 25 DM
Game Over (Action) : 15 DM
Grand Monster Slam (Sport) : 10 DM
Grand Prix Master (Autorennen) : 20 DM
Gyropod (Action) : 15 DM
Hollywood Poker Pro (Strip Poker) : 20 DM
Implosion (Action) : 15 DM
Indy Heat (Autorennen) : 20 DM
Invest,Winzer (Wirtschaftssim.) : je 30 DM
Iron Lord (Ritterspiel) : 15 DM
James Bond Collection : 20 DM
Kinetik (Geschicklichkeit) : 20 DM
Mag Max (Action) : 15 DM
Quill-Adventure Writer : 20 DM
Zig Zag (Action) : 15 DM
Trailblazer (Action) : 15 DM
Domark Spielepack : 15 DM
(Toobin,Licence to kill,Hard Drivin´)
19 - Boot Camp (Action) : 15 DM
Suspended (Adventure),Commodore :20 DM
Samurai Trilogy (Action) : 15 DM
Sidearms (Action) : 15 DM
Target Renegade (Action) : 10 DM

Sonderangebote - solange Vorrat reicht :

Disks je 5.- :
Warlock the Avenger,Loopz,Lone Wolf,Impact, Garrison,Plasmatron,I Alien,Discovery,Troll, Cauldron II,Paperboy,Quest for Tires & Zip, Inner Space,Jet Boys,Madness,Rimrunner

Disks je 10.-
Arkanoid 2,Pascal 64,Assembler/Monitor 64, Logo,First Word,Oink,Fifth Quadrant,Vera Cruz,Saboteur 2,C128 Musik Maker,Exile, Ace,Moonshadow,Volley Ball Simulator, Security Alert,Batman,Last Mission, After Burner, Action Fighter,Stunt Car Racer, Ballistix,Blasteroids,Kokotoni Wilf,Mission Elevator ,Snap Dragon,Scorpion

Tapes je 1.00 DM :

Vera Cruz,Lifeforce,Traxxion,ISS,Bangkok Knights,Der Fall Sydney,Inner Space, Mercenary Kompendium (deutsch !!)

Tapes je 1.50 DM

X-Out,Steve Davis Snooker,Dragon Spirit , Tracksuit Manager, Ice Palace ,Cuddly Cuburt , Rupert and the Ice Castle , Mystery of the Nile,Mega Apokalypse, Killed until dead , Pi-r-squared

Tapes je 2.50 DM

Firetrack,Thundercats,Championship Sprint, China Miner,Implosion,Defender of the Crown, Traz,Sidewize,Blood Money,Krystals of Zong, Die Erbschaft,Die Erbschaft 2,Oink,Critical Mass,Hades Nebula,Crazy Comets,Fruity, Sanxion,Galvan,Pile Up,Last Mission,Dandy, Sidearms,Moby Dick,Frogger 64,Starray, Deactivators,Time Scanner,Bounces, Ace 2088,Space Doubt,Equinox,Panic 64, Mothership,Dominator,Toobin,Metro Blitz, Challenge of the Gobots,Skramble 64, Dancing Feats,Hotshot,Starfox,Aquanaut, Star Raiders 2,IQ,4 Great Games,Starion, Ranarama,Velocipede,P. Beardsley´s Football

Hardware : Computer & Floppies & Zubehör

C64-I ("Brotkasten") : 50 DM
C64-C (neuer SID 8580,neue CPU) : 50 DM
C64-G (weißer "Brotkasten") : 50 DM
C128 : 50 DM
Floppy 1541-I oder II, weiß o. grau : 50 DM
Floppy 1571 : 70 DM
Floppy 1581 : a.A.
Oceanic u.a. Fremdlaufwerke : a.A.
C128D Blechgehäuse : ab 120 DM
C128D Plastikgehäuse : ab 80 DM
SX-64 : a.A.
C16/116/Plus 4 : a.A.
VC20 : a.A.
Final Cartridge III (Freezer) : a.A.
Action Replay u.a. Freezer : a.A.
Joystick-Mouse : 20 DM
Interpod RS 232 Interface : 20 DM
Disklocher : 5 DM
Staubschutzhaube C64-I : 8 DM
Staubschutzhaube 1541-I : 8 DM
Monitore 1702,1802 : ab 50 DM
Monitore 1084 , 1084S : ab 80 DM

C64 Klassiker & Oldies (zum Teil gebraucht) :

SSI :
Pool of Radiance , klassisches Rollenspiel : 30 DM
Buck Rogers , Science Fiction RPG : 30 DM
Curse of the Azure Bonds , Rollenspiel : 35 DM
Gateway to the Savage Frontier , Rollenspiel : 35 DM
Heroes of the Lance , Action-Rollenspiel : 25 DM

Infocom :
Plundered Hearts , Piratenabenteuer , neu : 60 DM
Zork I , Fantasy C16/Plus 4 Disk , neu : 30 DM
Zork II , Fantasy C16/Plus 4 Disk , neu : 30 DM
Zork III , Fantasy C16/Plus 4 Disk , neu : 30 DM
Suspended , Science Fiction C16/Plus 4 Disk , neu : 30 DM
Starcross , Science Fiction C16/Plus 4 Disk , neu : 30 DM
Stationfall , Science Fiction : 30 DM
Wishbringer , Fantasy (Goodies fehlen) : 20 DM

Electronic Arts (US Versionen) :
Strike Fleet , Flottensimulator : 30 DM
PHM Pegasus , Simulation : 30 DM
Patton vs Rommel (neu) : 50 DM
Hacker , Adventure : 30 DM
Arcticfox , Simulation : 25 DM
Chuck Yeager´s AFT , Simulation : 30 DM
One on One , Basketball : 25 DM
Marble Madness , Arcade : 20 DM
Dan Dare , Action : 20 DM
Axis Assassin , Action : 20 DM
Bard´s Tale 1-3 : a.A.
Music Construction Set , Anwender : 20 DM

Andere Hersteller :
Rommel battles for North Africa , (SSG) (neu) : 50 DM
Star Fleet 1 ,Strategie (Interstel) : 35 DM
Sorcerer Lord , Fantasy-Strategie (PSS) : 25 DM
War in Middle Earth , Fantasy Strategie : 30 DM
Barbarian , Action (Psygnosis) : 25 DM
Deathbringer , Rollenspiel (Spotlight) : 25 DM
Pirates , Simulation (Microprose) : 35 DM
Infiltrator 1 oder 2 (Mindscape) : je 20 DM
Deja Vu , Adventure (Ariola) : 40 DM
Napoleon in Russia,Strategie (Datasoft) (neu) : 50 DM
Dragon Wars, Rollenspiel (Interplay) : 35 DM
Oil Imperium , Wirtschaftssimulation (Reline) : 20 DM
Pegasus Bridge , Strategie (PSS) : 30 DM
Countdown to Shutdown (Activision) : 30 DM
Starglider , Action (Rainbird) - Tape+Disk-Kopie : 20 DM
Konflikte 1 , Strategie (PSS) Tape : 15 DM
Ace of Aces , Flugsimulation : 30 DM
Blue Max , Flugzeug Ballerspiel : 25 DM
F15 - Strike Eagle , Flugsimulation (Microprose) : 30 DM
Decision in the Desert , Strategie (Microprose) : 30 DM
Ghostbusters , Action (Activision) : 30 DM
Jumpman , Plattform (Epyx) Tape : 15 DM
Swiss Familiy Robinson , Adventure (Windham) : 30 DM
Zeppelin , Action (Synapse) : 20 DM

Hardware Restposten/Schrott :

ICs (ungeprüft) je 1.- :
318005-04,6510,901227-03,906114-01,901225-01
Defekte C64/C64C/128/1541/1541-2 Platinen je 10.-
Defekte C64/C128 (komplett) : 15 - 20 DM
Leergehäuse C64/C128/1541 je 5.-
Tastatur C64 weiß oder schwarz , ok : je 10.-
C64 oder C128 Netzteil : 20 DM
TV- , Floppy- , Monitorkabel : a.A.

JOS - Get the most out of your SuperCPU

CMD SuperCPU Corner

Why would I want a new OS?

Some of the appeal of the C64 was the fact that it didn't really have an OS, you just turn it on and quick as a flash your greated by BASIC, with the Kernel doing the background work. Alot of programs (especially games), when they had booted, switched out the Kernel, installed there own disk/tape routines, and proceeded to load the rest of the program without bothering with the Kernel at all, and this was done because the standard Kernel routines for disk I/O are incredibly slow, and also weren't interruptible so doing something else at the same time wasn't possible.

Anyone who knows about the SuperCPU knows that it runs at 20MHz and can have access to 16MB directly. With simple logic you can work out that the SuperCPU should perform 20 times better, and run applications that are much more cpu intensive or require much more RAM. But that currently is not happening. Most people are using SuperCPU's purely to make stock c64 programs go faster, and there are few people developing specifically for SuperCPU. This is not how it should be! With a bit of luck, and a bit of support, this will change. First things first the SuperCPU needs a new OS. But why? Read on...

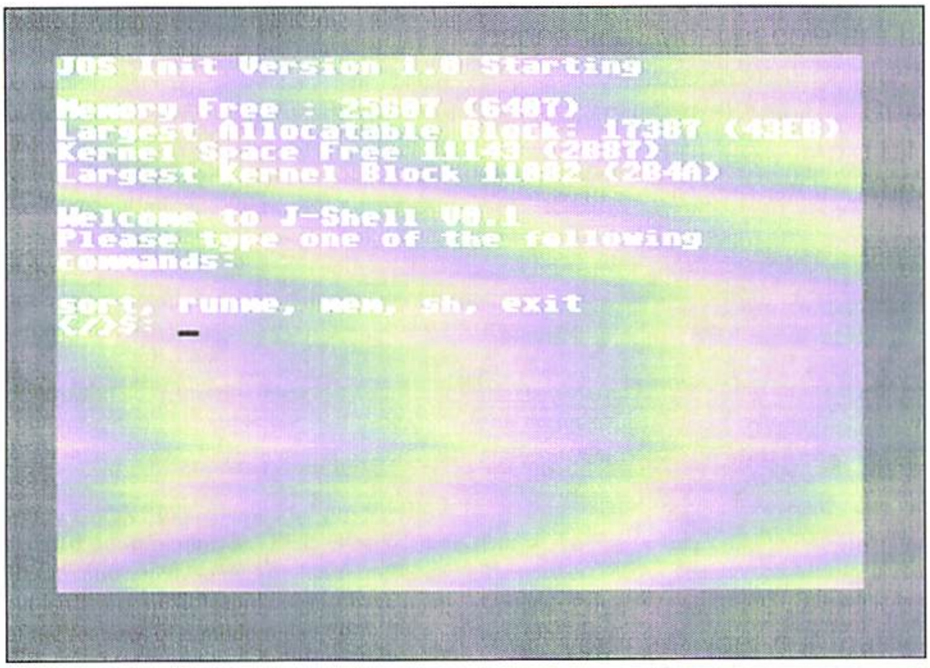
by Jolse Maginnis

Bypassing the OS, is a sure way to make your program incompatible with new or different hardware. Anyone who wanted to use their brand new REU as a ramdisk weren't able to unless the programmers decided to program an REU system too. GEOS alleviated some of these problems, but isn't very flexible and doesn't have many of the features of a modern day OS.

Bypassing the OS, and having control over the whole computer was the only way to get any performance out of C64, and the lack of memory was also another good reason to switch the Kernel and BASIC out.

But we're not talking 64k, 1MHz anymore. The SuperCPU is capable of having a real OS and it deserves a real OS. But what is a real OS?

A real OS should provide atleast some of the following.



```
JOS Init Version 1.0 Starting

Memory Free : 25807 (6487)
Largest Allocatable Block: 17387 (43EB)
Kernel Space Free 11143 (2B87)
Largest Kernel Block 11082 (2B4A)

Welcome to J-Shell V0.1
Please type one of the following
commands:

sort, runme, mem, sh, exit
</>$: _
```

Multitasking - Every should know what this means by now...

Memory Management - When multitasking, the memory has to be shared amongst the programs running.

Device independence - The application doesn't need to know that it's writing to a ramdisk rather than a real disk.

Networking - Nowadays TCP/IP is the default choice, since it allows you to use the internet.

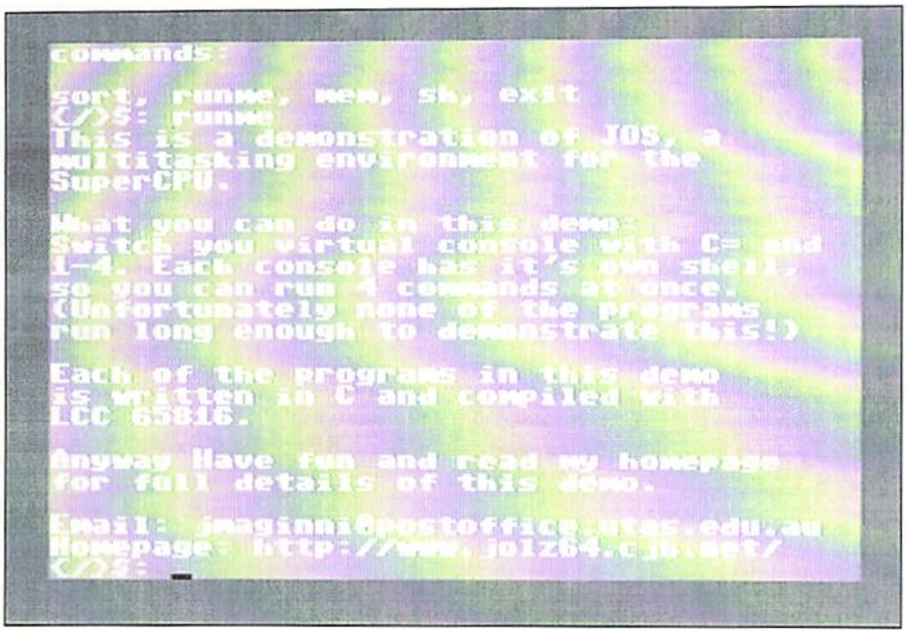
Shared libraries - Program size and memory usage is greatly minimised by using shared libraries.

Shared code - If the same program is running twice it should be able to share the code.

Disk cache - A big speedup can be had by storing frequently used disk blocks.

Effective IPC - Programs and device drivers have to be able to communicate with other programs quickly and effectively.

Virtual memory - Not really possible with a



```
commands:

sort, runme, mem, sh, exit
</>$: runme
This is a demonstration of JOS, a
multitasking environment for the
SuperCPU.

What you can do in this demo:
Switch you virtual console with C= and
1-4. Each console has it's own shell,
so you can run 4 commands at once.
(Unfortunately none of the programs
run long enough to demonstrate this!)

Each of the programs in this demo
is written in C and compiled with
LCC 65816.

Anyway Have fun and read my homepage
for full details of this demo.

Email: jmaginni@postoffice.utas.edu.au
Homepage: http://www.jolz64.cjb.net/
</>$:
```

65816, but isn't really needed since the C64 way is anti-bloat.

All of the things in that list (aside from virtual memory) are/will be implemented in JOS. At this stage only Shared Libraries, Networking and the GUI aren't nearly finished.

Still not convinced? Well let's say you wanted to be able to check your email with your SuperCPU...

A non OS email program would need it's own TCP/IP and SLIP/PPP networking built in, and then once you have checked your email, you can't do anything else. You'd have to turn off or restart the computer just to load another program, and if you needed to use another internet program you'd be forced to reconnect!

This would not occur with JOS, you could load your email program and check your email and even talk on IRC all at the same time.

And what about anything that wants to have a GUI? Everyone is forced to write there own GUI and that in itself is enough for someone not to bother. But if it's part of the operating system, it's alot easier to . This is why GEOS is so popular and is still being used.

Well that's all very well but will I be able to use it easily?

JOS without a GUI will look and feel basically like unix. Alot of people are becoming familiar with unix because of the wide availability of linux and other free unixes. It isn't the most user friendly of OSes because of it's command line nature, but it gives you alot of power. Things such as shell redirection and piping will be fully implemented.

An example would be:

"cat readme.txt | more"

which would display the file readme.txt and have prompt when the screen is full.

Eventually though, JOS will have a GUI. I haven't put much thought into the coding of it as yet, but I do have some experience programming a GUI, as I tried porting some of Borlands Object Oriented Turbo Vision, to C64, and it was quite succesful. So I already have some ideas as to how the GUI's going to be implemented.

Another interesting point is that running a Java Virtual Machine (JVM) under JOS is a realistic goal. I recently received an email pointing me to Sun's web pages on their new cut-down and optimised JVM called KVM. Sun claim that it's code is only about 40k, and it is certainly within the realms of possibility that it can be ported. Java is a platform independent object-oriented language that has a large support base, and this JVM would allow java programs to run in JOS.

What hardware will JOS be compatible with?

For starters, JOS doesn't require any extra RAM to run. SuperCPU's actually contain 128k of RAM, but normally only 64k is usable. Since JOS doesn't use any of CMD's code or anything at all from the Kernel, the extra 64k can be used, leaving plenty of RAM left.

Initially, JOS will only be compatible with the hardware which I have access to. But it really is a simple task to write new drivers for other hardware. So far all I have access to is 1541 disk drives. I should be able to code device drivers for some devices without ever seeing them, such as an REU.

Techical Details

Technically and structurally JOS is very advanced. Alot of the ideas have come from reading technical papers on the web. Most notably the technical papers about the QNX Realtime Operating System. They produce a demo-disk which contains the OS, a GUI, TCP/IP networking, a web browser and a PPP dialer, all of which is on the one disk! None of the bloat seen in Microsoft OSes and even Linux. This inspired me to make something similar for the SuperCPU.

JOS is a microkernel based OS, rather than a monolithic kernel OS. A microkernel is a system where several co-operating processes, work together to provide the services of an OS. So for example, in JOS, all the memory management is done by a memory manager process, and all console (the screen) I/O is controlled by the console process (con.drv). The same goes for other drivers like the serial bus driver (iec.drv) and the commodore filesystem.

There has been much debate about whether a monolithic or microkernel approach is best when creating an OS, and I must admit that I used to think that a microkernel would be incredibly slow with all the task switching it needs to do. It's only when you start coding one that you realise just how great they are. This system is divided into easily managable chunks of code, which you can work on seperately much easier than with a monolithic kernel.

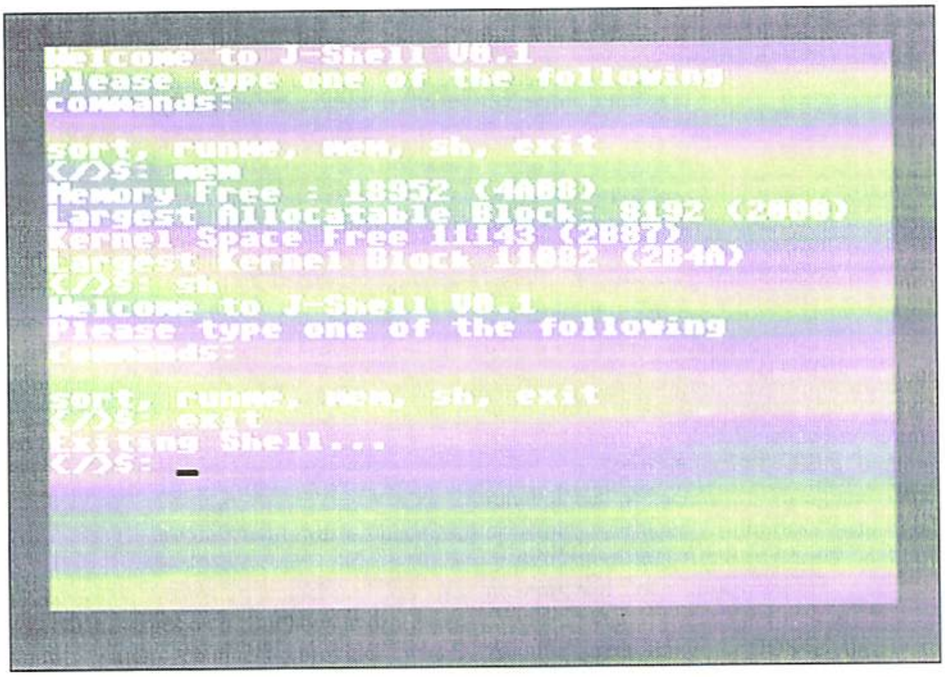
As for speed, the key is to have very fast and very effective IPC (Inter Process Communication). Much is said about the slowness of Microkernel's because they have to copy messages between processes. JOS doesn't suffer from this because it takes advantage from the fact that the 65816 has no MMU and all memory is shared. If you add this together with the fact that the IPC is syncronous (sender waits for the reply), no message copying is necessary. This provides a significant speed increase over message copying, but it does mean that before a process can be terminated, it must wait until no other processes are using it's memory, but that's a small price to pay.

All messages are directed towards channels. So if a process needs to talk to other processes, it creates a channel and waits to receive messages on it. But how do processes know which channels provide which services? There are 2 system processes which have known channel numbers, the memory manager owns channel 0, and the process manager owns channel 1. But what if you wanted to access the console? The console process registers a name with the process manager, say "/dev/con", and then any process that wants to use the console sends a message to the process manager asking to find the channel associated with "/dev/con" and the process manager returns the result. Simple.

Not only is JOS multitasking but it is also multithreading. They are essentially the same thing except threads run together within a process, allowing processes to do two things at once. Probably the best use for this is when writing a program for a GUI, since the code to update the screen can run at the same time as the code doing the real work.

Alot of programs end up using common code, such as routines to print formatted text, and string manipulation routines, so it's a good idea to be have this code loaded once and have all programs share it. Thanks to the 65816's larger stack and it's data bank register, this can be achieved in JOS without too much fuss. It already shares code segments between already loaded programs (you can load about 25 shells without running out of memory!). Using the same principles, shared libraries can be implemented and used to save alot of memory and alot of disk space.

Another great thing for JOS is the LCC compiler. I created a 65816 backend for this free compiler, and it produces quite good code. Although all of the microkernel code and most drivers will be written in 65816 Assembler, utilities and applications can be written in C. Not only that but since JOS is quite similar to Unix in many ways, programs can be easily ported. Unix carries with it alot of historical baggage that just isn't relevant for JOS, so i don't plan on making it so that everything can just be compiled straight off, but rather just easy to port.



```
Welcome to J-Shell V0.1
Please type one of the following
commands:

sort, runme, mem, sh, exit
</>$: mem
Memory Free : 18952 (4A08)
Largest Allocatable Block: 8192 (2000)
Kernel Space Free 11143 (2B87)
Largest Kernel Block 11082 (2B4A)
</>$: sh
Welcome to J-Shell V0.1
Please type one of the following
commands:

sort, runme, mem, sh, exit
</>$: exit
Exiting Shell...
</>$: _
```

If you take that all into consideration you can see that JOS is a pretty advanced and effective design, much better than many other operating systems, and even boasts features that several operating systems didn't have until recently, or don't even have at all.

Applications planned for JOS

These are the first things I'll be coding/porting once there is enough of the operating system to work with.

PPP dialer (not just SLIP)
Gzip/PKUnzip/something similar..
Telnet.
IRC.
Email Client.
Web Server. Simple Web browser.

I know that all sounds optimistic, but trust me, they will all be done in time.

What would it be like in a dream world?

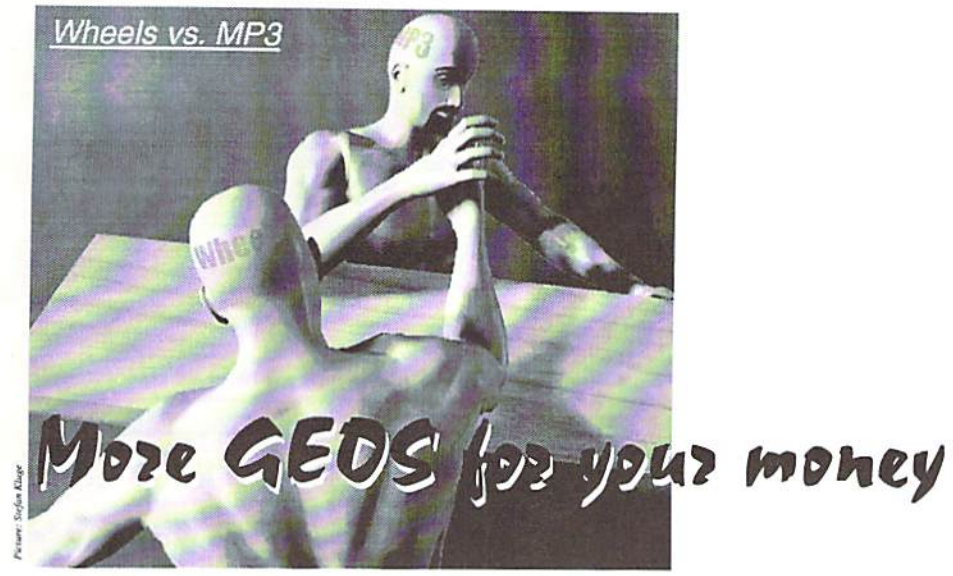
We'd all be connecting to online game servers playing c-64 games against each other. Hey it's a possibility!

Where is JOS upto at the moment?

At this stage I need to do several things before I can release anything that'll make you want to throw away your PC, but it's not far off. I suggest you check
http://www.jolz64.cjb.net/jos/
every now and then to see if anything has been released.

If you'd like to help with this project in any way, or you'd just like more information, send me an email at
jmaginni@postoffice.utas.edu.au

Thankyou for reading this article and I hope it inspires any non-SuperCPU users to take the plunge and get themselves a SuperCPU!



Wheels vs. MP3

More GEOS for your money

Picture: Stefan Kluge

GEOS, the graphical user interface for the C64/128 has been existing since 1986 - but it's already 10 years since the last update to version 2.0. In 1993, a new version 2.5 was released in Germany, but it didn't contain any really new features. Everything in V2.5 was already available - you just didn't have to buy it seperately if you upgraded to 2.5 - and this was the last big event concerning GEOS, until now.

by Nikolaus Metz

Almost six years later, us GEOS users finally have reason to ask ourselves what we might still be missing - more or less at the same time, but independent from each other, two updates to the now 13 years old GUI were developed: "Wheels" by Maurice Randall (in America), and "MegaPatch 3" (in short: MP3) by Wolfgang Grimm (in Germany). This GUI has become a hit, as can easily be seen from the variety of applications and utilities available on the commercial market as well as in the Public Domain pool. My personal archive likely contains about 35 to 40 Megs of GEOS stuff - and that's just an estimate, I think it might be even more.

Big brother and little brother

New developments on the GEOS market always originated in America - which isn't a miracle, as GEOS was first developed there. That's certainly not to say time stood still in Germany and people there just looked across the pond and used the products coming from the US. In the meanwhile, Europe has become as active as America; TopDesk, which was released 6 years ago and included in the German GEOS version 2.5, replaced the original DeskTop. Another development was RAMProcess, which allowed the user to integrate a variety of utilities directly into GEOS. And last, but not least, the German GEOS users were the first to use the SCPU128 at a full 20 MHz with GEOS 128 V2.01.

By now, Maurice Randall (the developer of Wheels) is working on 'Config128 2.1s' which will allow you to use a RAMdisk in a SuperCPU RAMCard. No need to say, we will follow this CMD development and report any new results.

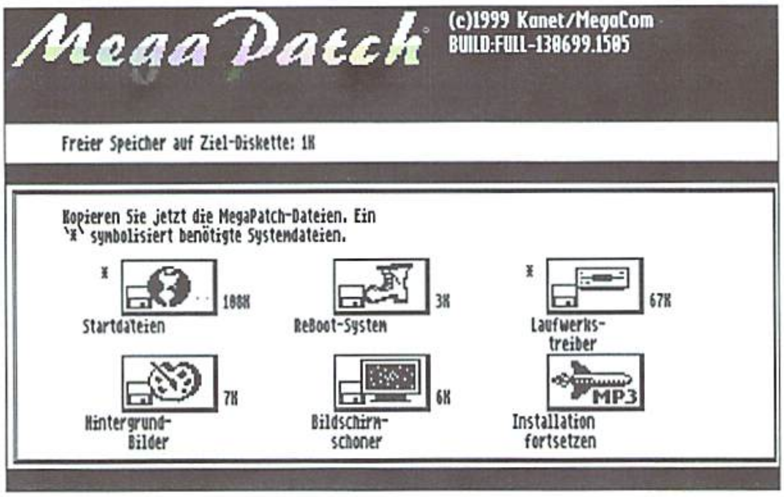
The first time

The first differences between the two GEOS updates turn up during installation: while the MP3 setup runs only under GEOS (and under Wheels as well), Wheels can be installed directly from its master disk, without starting GEOS first. This was not the case with the first versions of Wheels; these had to be installed from GEOS. This is a huge plus, as the annoying preconfiguration process, which, in earlier versions, could cause problems with GeoMakeBoot, isn't needed anymore. You simply connect all drives and expansions, power up the computer and the drives and start the installation on this configuration; the components are automatically recognized and installed. The GEOS system disk is only needed to read the serial number and to test if the GEOS system is present - this is necessary because Wheels, as well as MP3, is merely GEOS update.

The installer of MegaPatch3 must be started from GEOS, Wheels or MP3, and, if all drives should be immediately usable, they have to be configured already. Users with a 16 meg RAMCard might encounter problems under Wheels if they have preset a 16 meg RAMDisk, as MegaPatch can only use 4 megs of the expansion memory. In this case, the RAM drive has to be reconfigured later, before finishing the installation. On the other hand, if you've configured a RAM81, it's taken over without any problems.

One positive thing you notice at once is that MP3 can be installed from any drive to any other drive, even onto a RAMDisk. From there it can be copied onto the final boot partition or disk which can then be made bootable using 'GEOS64(128).MakeBoot'. Unfortunately, Wheels' 'MakeSysDisk' will only run if started from a real disk; at this point we have to divide between the master disk and a boot disk/partition. A boot disk or boot partition can be created on any drive, while a MasterDisk with the 'MakeSysDisk' program necessary for the installation can only be created on a physical (real) drive. Therefore you'll have to find a master disk each time, and the installation process might take longer.

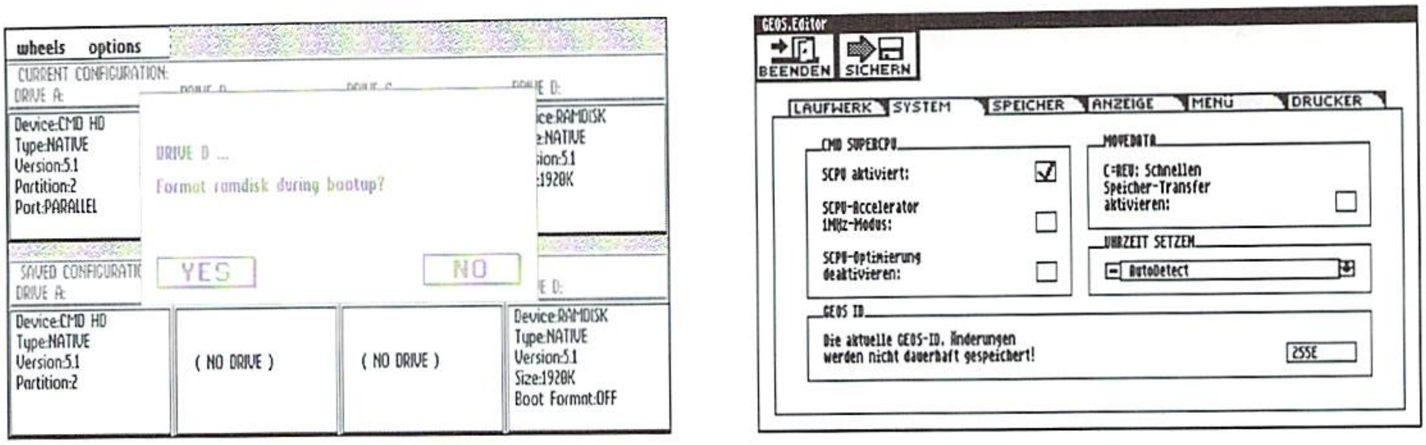
MP3 during installation



MegaPatch (c)1999 Kanet/MegaCom
BUILD:FULL-130699.1505

Freier Speicher auf Ziel-Diskette: 1K

Kopieren Sie jetzt die MegaPatch-Dateien. Ein
'*' symbolisiert benötigte Systemdateien.



CURRENT CONFIGURATION:
DRIVE A:
Device:CMD HD
Type:NATIVE
Version:5.1
Partition:2
Port:PARALLEL

DRIVE D ...
Format ramdisk during bootup?

SAVED CONFIGURATIC
DRIVE A:
Device:CMD HD
Type:NATIVE
Version:5.1
Partition:2

(NO DRIVE)

(NO DRIVE)

Device:RAMDISK
Type:NATIVE
Version:5.1
Size:1920K
Boot Format:OFF

GEOS.Editor

CMD SUPERCPU
SCPU aktiviert: ☑
SCPU-Accelerator 1MHz-Modus: □
SCPU-Optimierung deaktivieren: □

MOVEDATA
C=REU: Schnellen Speicher-Transfer aktivieren: □

UHRZEIT SETZEN
AutoDetect

GEOS ID
Die aktuelle GEOS-ID. Änderungen werden nicht dauerhaft gespeichert! 255E

Toolbox (Wheels) and GEOS.Editor (MegaPatch 3)

Editor/Toolbox

Both updates brought a huge improvement to GEOS users by replacing the old 'Configure' with entirely new systems. With this, you can now configure all four drives, without having to look for the appropriate 'Configure'. Also, the constraint that the fourth drive has to be a real 1541/71/81 drive is now gone, as you may have any drive type on all positions. MP3 is a step ahead at this point, since it allows 1541 and 1571 partitions on all CMD drives, in addition to the 1581 and native partitions offered by Wheels. However, whenever you change the drive configuration, you have to activate the appropriate driver by hand, contrary to Wheels, whose Dashboard automatically handles the changes between 1581 and native partitions. It is planned to include this feature in GeoDesk, too.

Additionally, MP3 offers an MS-DOS driver for 1581 and FD floppies which enables you to copy directly between MS-DOS formatted disks and another drive using Commodore disks. Unfortunalely, this will reduce the file name to 8 plus 3 characters (for the suffix); strangely, the dot between them is omitted. Because of the low transfer speed, this feature's no competition to Big Blue Reader or GeoDos (a real 128 version of GeoDos is planned for the near future).

One thing that hit my eye is the difference in the way you select the RAM expansion for the system. With Wheels, you have to repeat the whole install procedure after adding or removing an expansion, including selection of mouse and printer drivers and so on, even though none of these has changed and you don't want to change them. Somehow, this reminds me of plug'n'play - and I'm not fascinated of it (or maybe in spite of it?) - it just makes it all slower and more complicated. In contrast, MegaPatch installs its RAM area as usual, and only complains if the selected expansion is missing or out of order. If you want to (or have to) use another expansion, you simply run 'GEOS64(128).Reset' and select it, and the system reboots automatically - I find that much easier.

With either system, 64Net users can't get direct access yet, however, it's possible under MP3 using the little trick shown in the previous issue of GO64! (7/99, p26); but there'll have to be drivers for both updates in the future.

Task manager

It's always nice to enrich articles like this one with screen shots illustrating things to the readers. MP3's unique task manager makes this very easy, while the competitors don't see any necessity for this (to express it politily). Changing to another application or into another document without going through TopDesk is just as easy, as MP3's task manager is available at nearly all times, and has but few limitations.

The task manager also contains the printer spooler that allows you to manage your print jobs. One of the advantages this gives you is that before a print job is executed, you can change the driver if necessary, and can also access other drives for this. Another is you can keep print jobs and print them later, so you'll have to take just one coffee break instead of several ones. Also, you can select the numbers of copies, as users of TextPrint V3 know it.

Dashboard vs. TopDesk

Although we'll still have to wait some time until GeoDesk arrives, TopDesk V4.1 has much to offer and can compete with DashBoard. On the one hand, its (mostly German) users who've already used the earlier versions of the DeskTop alternative won't have to get used to another graphic interface, particularly as its screen design has proven its qualities. On the other hand, new features have been added to the known ones, such as calling the partition change routine by clicking on the lower window bar - in earlier times, I often got my fingers in a knot using the C=J key combination. The dash board doesn't require such a window bar function, as the C=S combination it uses is far more practical.

The newest TopDesk version even has a particular feature that has been available in Dashboard in a rudimentary form since its beginning. Until now, like the Dashboard, TopDesk always reactivated an open, inactive window when you clicked on it, which I found quite annoying during the testing phase as this could take quite some time. Wolfgang Grimm paid serious attention to this problem and found an elegant solution: now, if you click on a function in an inactive window, TopDesk switches to the window and immediately executes the function, as it can be done in Dashboard. Also, MegaPatch will mark a file in an inactive window when it's clicked upon, and invert the file name.

Another feature I really liked in TopDesk is its modern date&time display, in contrary to Dashboard, whose German installation still displays the time in American format and will show a '00' next year. Unfortunately, this is not going to change in the near future, as Maurice Randall told me upon request: "The Y2K problem in GEOS is limited to applications like geoWrite or geoPublish." But, as he assured me, the translation of the menues in Wheels into German language is under way. One of the future updates to Wheels will also contain a trashcan which is still missing in the current version, and a request for additional screen savers as offered by MegaPatch is on its way.

But Wheels has its own advantages, like the system directory, which holds, for instance, the applications and fonts in native partitions, thus separating them from the documents. An application can access the system directory and its home directory as if they were one, but in the Dashboard they appear fully seperated, providing a new quality in viewing disk contents.

Another thing that I miss desperately in TopDesk, is the ability to create subdirectories, a huge deficit that has to be filled in sooner or later. You can, however, create subdirectories in BASIC mode, but I usually find myself starting Wheels for this purpose as I don't know yet if creating subdirectories from BASIC has any effects on existing GEOS files.

There are some other disadvantages to TopDesk, which aren't likely to be amended before the release of GeoDesk: for instance, there are but four windows with one common mode (either icon or text display), compared to 16 available under Wheels, each of whom has its own mode. Another flaw of TopDesk is its lack of a format routine for RAMDisks and RAMLink partitions. When it comes, it should include a security query like the one in Dashboard, that asks the user to type the partition number or name in addition to the drive, which is the only way to avoid formatting the wrong partition by mistake, as I did some time ago.

We Could use some more speed

Both systems make use of the speed provided by modern hardware, particularly when used in conjunction with an SCPU and/or a RAMLink. Normal accesses to a RAMLink are executed five to eight times faster, and in addition, Wheels supports the parallel connection to a hard disk via the RAMLink. This way, the Dashboard executes HD operations about twice as fast on average. A validate on a 16 meg native HD partition shows an extreme speed difference: while TopDesk needed 2000 seconds, Wheels took a mere 560 seconds until it was done. No wonder support for a parallel connection is on one of the top positions on Wolfgang Grimm/ Markus Kanet's list of missing features.

Reloading the Dashboard and TopDesk are realized in different, but efficient ways: it's still possible to have a RAM-TopDesk, which should be used, since it uses only one bank of the RAM expansion which is required anyway. Wheels searches the available drives for the Dashboard, beginning with the fastest one, therefore it's an advantage to have a boot partition on a RAMLink.

The results of my measurements up to now can be found in a geoWrite document on the coverdisk, rather than in a boring table that people who aren't into GEOS wouldn't want to read anyway. This table is not complete yet; any user who'd like to contribute some information can fill it in and send it back to me - as a geoWrite file, if possible - via email to Nikolaus.Metz@SachsenGEOS.fido.de . If you do, please, don't forget to include information on your system configuration, since comparison to other people's data doesn't make sense without this information.

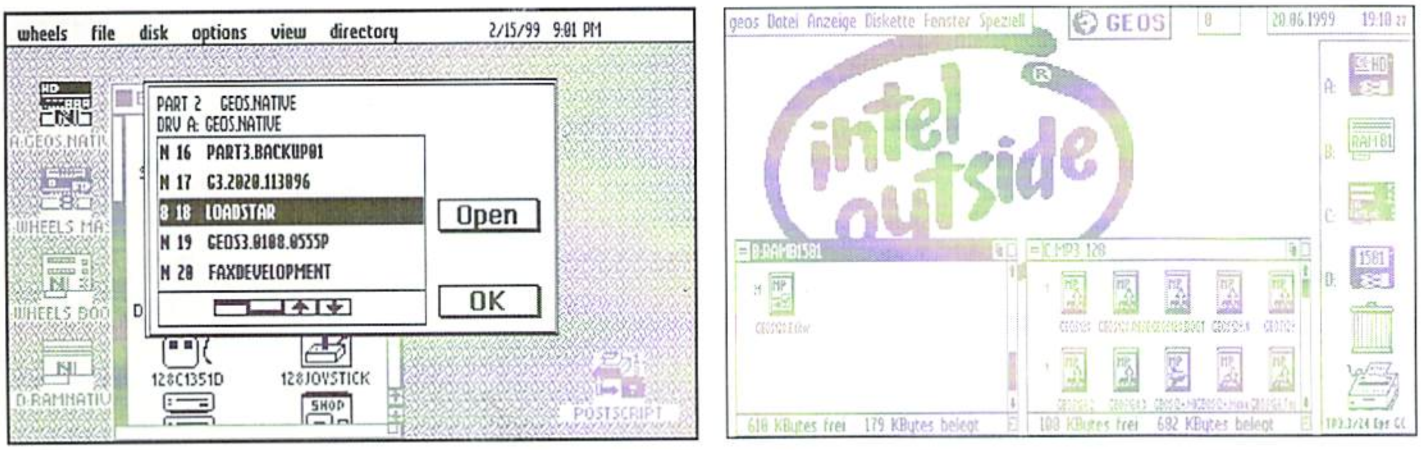
Old and new

What if you want to use old applications under a new system? You need patches for the applications - the more, the better. Werner Weicht supplies patches for geoWrite, geoPaint, geoCalc, geoChart, geoFile, geoPublish, geoMerge, and even for geoSpell (written upon request of a single Mr. from England - and no, this time it wasn't me!). There are different versions of the patches for the US and German versions of the programs. The patches change the file request routines in such a way that documents can be created on any exisiting drive (A to D). Also, documents can be started from any drive, if you're using a DeskTop with this capability, like TopDesk V3.x and up.

There's also a patch for fans of BootTrans V2.x to make the utility run under MegaPatch3. For those who don't know it: BootTrans allows the user to specify a list of files which are automatically copied to another drive, like a RAMDisk, at boot time.

The patches are shareware and can be found on the coverdisk and Werner Weicht's homepage (http://home.t-online.de/home/wweicht/index.htm), as well as on BBS's that offer GEOS software. If you use them, please, don't forget to pay the shareware fee, to make the huge efforts pay back, and enable further improvements. Those who notice malfunctions in the programs can drop a message to Werner in order to help finding the bugs as soon as possible.

I found it interesting to observe how old utilities perform under the new systems. With MegaPatch, several tools have become obsolete, like RAMProcess with FileBrowser and similar ones; Screen shots can be made with the system itself, and BootTrans can be patched. However, Wheels doesn't like the above tools and refuses to run them. Maurice reasons with 'Safety first', meaning all normal applications and utilities run as well as before, but as soon as a program wants to manipulate the system, the command is blocked. This attitude may be right, after all we want to avoid imitating Bill G. (full name known to the editors and most readers) and his bluescreen during a presentation. For the future, however, I hope for better cooperation with the programmers who want to adapt old,



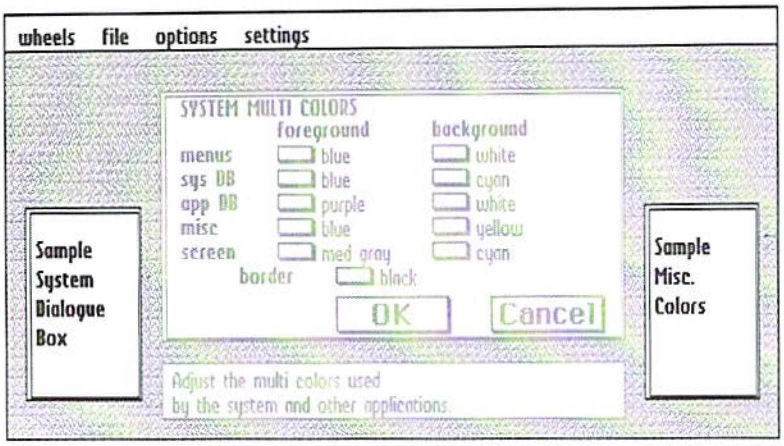
PART 2 GEOS.NATIVE
DRV A: GEOS.NATIVE

N 16 PART3.BACKUP01
N 17 G3.2020.113096
8 18 LOADSTAR
N 19 GEOS3.0100.0555P
N 20 FAXDEVELOPMENT

618 KBytes frei 179 KBytes belegt

108 KBytes frei 682 KBytes belegt

Dashboard (Wheels) and TopDesk V4.1 (MegaPatch 3)



SYSTEM MULTI COLORS

	foreground	background
menus	blue	white
sys DB	blue	cyan
app DB	purple	white
misc	blue	yellow
screen	med gray	cyan
border	black	

Adjust the multi colors used by the system and other applications.

Detailshop (Wheels)

but necessary programs for the new system.

Why update?

It should be clear by now, that with either of the updates, you'll need some modern hardware to fully exploit their potential. After all, the reason these updates have been made is that the original GEOS can't handle such hardware without additional software. And as soon as you've integrated a new device into GEOS, it'll fail on the next.

Those with only a bare C64/128, and drives like the 1541/71/81 and a REU 1750 certainly won't get the idea to buy Wheels or MegaPatch. But with only one other component from more recent times, things begin to look different: It's pure pleasure working with a large RAMDisk without having to watch the remaining space as you'd have to with a C=1750 that offers just enough capacity for a RAM71. Now, you can use the full capacity of a 16 Meg RAMCard under Wheels, and a quarter of it if you use MP3, while CMD's GEOS64 patch allows no more than 2 Megs and a RAM81; and under GEOS128, there isn't any way yet to use any of the expansion memory. Plus, the speed of a SCPU(128) has been risen a little by both patches.

Which update is the better?

German fans of GEOS, who kept using the system during the recent years, are likely to tend towards MP3, as with its TopDesk 4.1, there won't be too many changes to adapt to, besides the fact that it's all in German language. However, MP3, when compared to Wheels, shows some weaknesses concerning its speed, which will hopefully be settled in the near future. Equipped with a new GeoDesk, parallel cable support and an improved HD-native-driver, MP3 could well get ahead of its American competition.

I can't, and I won't take the decision for either of the programs off the user; everybody has his own opinion of what he requires of a graphical interface and will choose accordingly. I only hope I've given an objective review and fair comparison of both systems that will make your decision less difficult.

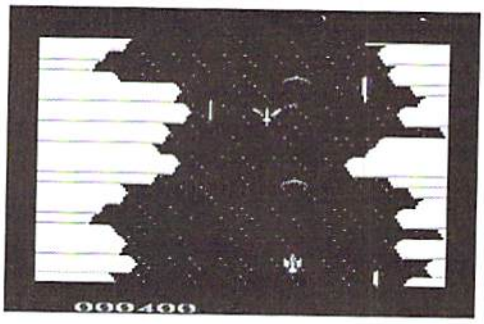
I have to remark, however, that both programmers have made a great effort to include everything in these updates they considered important. In my opinion, they have admirably succeeded in this, so us GEOS users can look ahead into the new millenium, knowing the good old breadbox will make it there.

Cover-disk

GigaCad plus for your SuperCPU

by Wanja Gayk

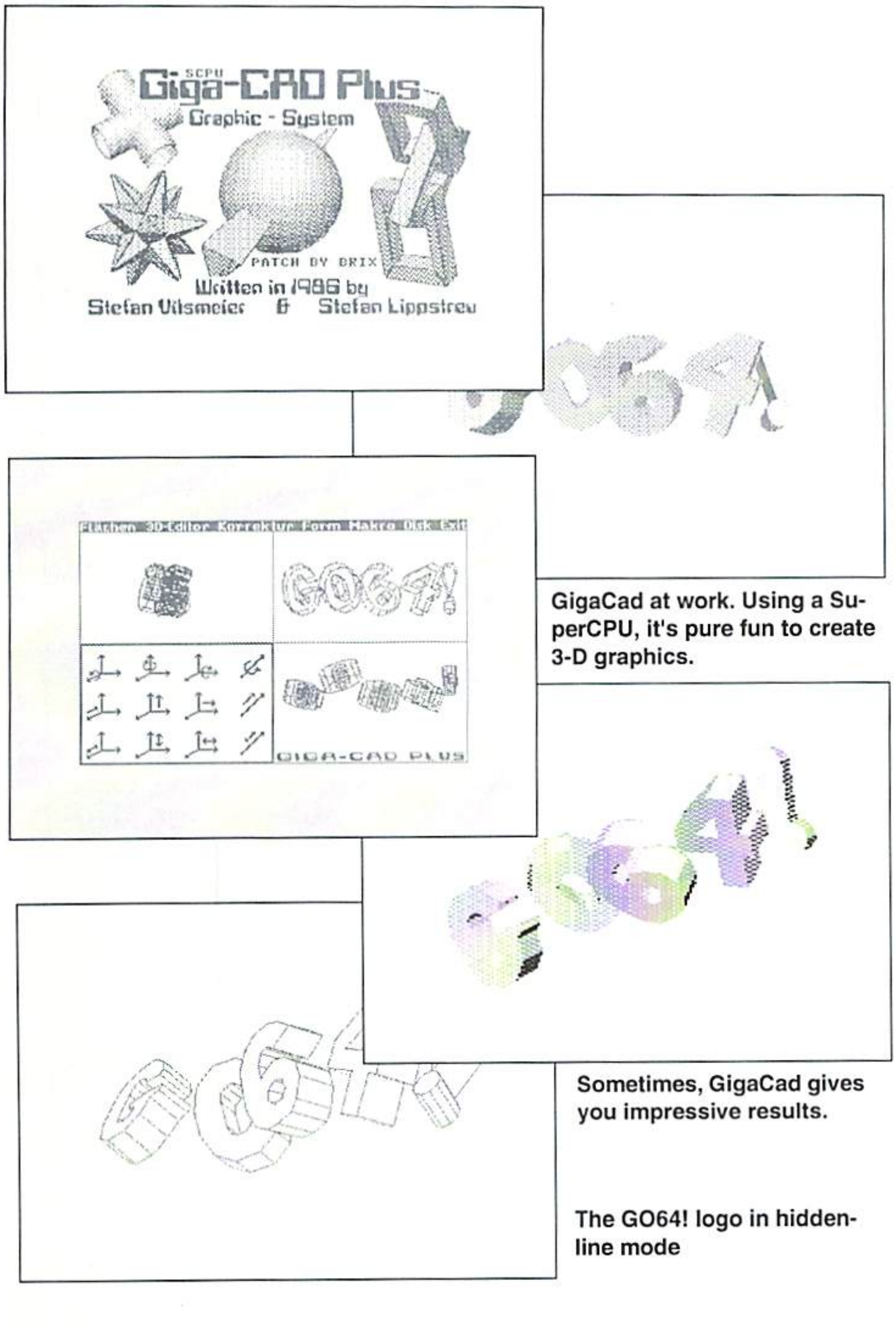
Once again, we have good news for users of a SuperCPU. The 3-D graphics program called "GigaCad+" was written by Stefan Vilsmeier and Stefan Lippstreu way back in 1986, and it's still without any competition on the C64. Unfortunately, it's not only known for its enormous capabilities but also for its slow working speed as well. However, it gets wings when running on a SuperCPU combined with JiffyDos - if only it weren't for the speed of the graphics cursor, which increases as well, practically rendering the program unusable. That was just the impulse I needed to give the program a try and write a small SCPU patch to fix this problem. However, the joystick button still reacts a bit too fast, so you should only give it a quick hit. The graphics cursor is only slowed down with the turbo switched on; if you disable the turbo, it will react as normal, which means the patched version of GigaCad can still be used without a SCPU and not cause troubles.



Zynax

The patch can be found on the coverdisk. To adapt GigaCad plus to the SuperCPU, you'll have to replace the files called "GIGA.G" and "GIGA.H" with the new versions from our disk. Please note that the patch works ONLY with the program "GigaCad plus", but not with its predecessor, "GigaCad". To make sure any malfunctions caused by the patch won't affect your original, I'd also like to recommend using the patch on backups only. Anyway, no bugs have been found yet.

GigaCad plus was published back in 1986 by the Markt&Technik publishing house as a "book + software" in German (ISBN 3-89090-409-2). Unfortunately, the publisher states that it's no longer available, so we recommend looking for it on the secondhand market.


SCPU
Giga-CAD Plus
Graphic - System
PATCH BY DRIX
Written in 1986 by
Stefan Vilsmeier & Stefan Lippstreu
Flächen 3D-Editor Korrektur Form Makro Disk Exit
GIGA-CAD PLUS
G064!

GigaCad at work. Using a SuperCPU, it's pure fun to create 3-D graphics.

Sometimes, GigaCad gives you impressive results.

The GO64! logo in hidden-line mode

Zynax

With this month's main subject, we certainly can't go without an arcade style game on our disk. Zynax is a classical space shoot'em'up, its motto being "don't ask, blast!". Please note that some cartridges aren't compatible with the game and should be disabled before loading Zynax. By the way, this game's the first in a trilogy, so your mission will continue...

MP3 Patches

Those patches are important to enable GEOS applications to work with MegaPatch 3. The patches are available for the 64 and the 128 versions of MP3, in English as well as in German language.

Unfortunately the patches had to be stuffed on disk in the form of zip archives, due to their number and size. This means that you first have to unpack the files with UNZIP (Either Errol Smith's Unzip64 or Pasi's Gunzip64) , then use CONVERT to convert them into GEOS format. After that, you should always read the "Readme" file ("Bitte lesen"), as it contains further important information.

Hasty Hens...

may be able to lay eggs quickly, but as far as remote data transmission is concerned, we'd better stick to our computer. But still, haste was the reason for the development of the "Datapump" project in the year 1991 by Perry M. Grodzinski.

by Niko Malecki

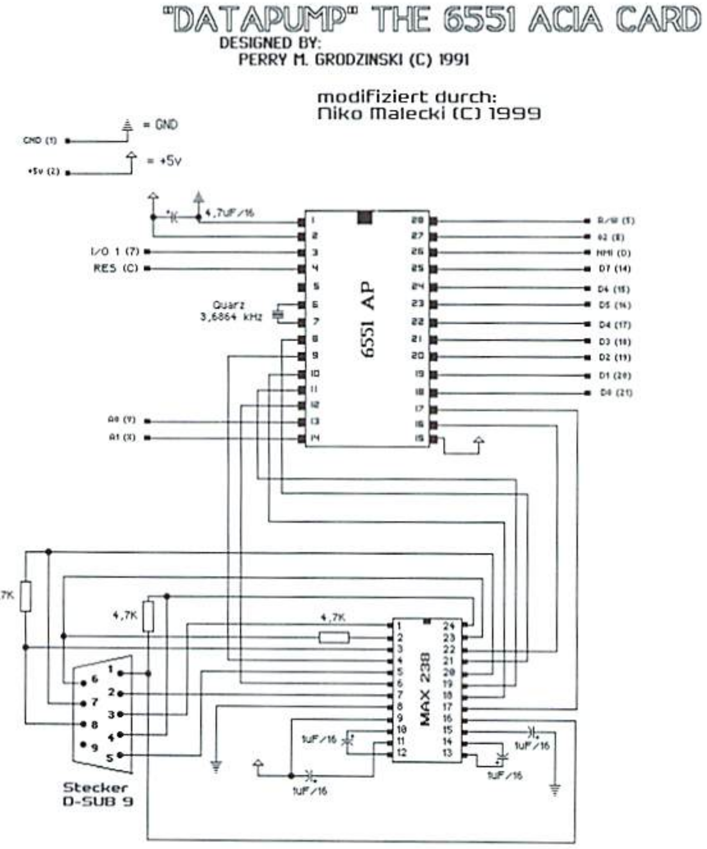
In an excellent way, Datapump does indeed makes sure that fast modems can be connected to our computers. Transfer rates of up to 38,400 bps are quite realistic here and allow for the phone bill to remain bearable. The construction instructions, stored at miscellaneous mailboxes and p.d. traders, are based on the 6551 ACIA-chip and therefore just about correspond to the renown interfaces, Swiftlink and Datablast.

Well, that was 1991, and the instructions are therefore already eight years old. In the computer age, eight years is nearly forever, and time hasn't stood still for the chips industry either. While Perry still needed four ICs in order to realize this thing, only two are needed today. I have implemented this modernization for you, and the construction is quite simple as well. The wiring diagram makes the required components quite obvious, the cost for them being about 18 Euros. Circuits ending with a number or letter in brackets are those leading to the cartridge port, but certainly you have already guessed this. One cheap source for all the material is:

Reichelt-Elektronik, tel: +49 (0) 4422-9550 or fax: +49 (0) 4422-955111

So that all things are in order, I must point out that the main construction manual by P. Grodzinski retains full validity and that my modernization is merely an addition or an option when constructing it. The wiring diagrams and the original instructions are found on our readers' disk (in GEOS format).

Yours, Niko



USEFUL TIP FOR THE C128

I've written about the voltage requirements of our computer a few times already. This subject came up especially after the introduction of the SuperCPU, which draws more than than the normal power requirements from the power supply. First, it's important to check to see if your power supply can deliver the minimum voltage of 5V. Pin 2 of the userport can be used to test the voltage; you should first test the machine without any expansion equipment attached, that is, either user port or expansion port devices. The voltage should read exactly 5 Volts, and an exact measurement could only be read with a digital multimeter. If just a tenth of the power is not being supplied, it means you have a weak power supply. The C128, regardless of whether it's flat or the D model, uses a power supply with a device known as a regulator. This device adjusts the power flow, and is what is used to get the desired 5 Volts. There is only one regulator, so it's not possible to switch it. It is advantageous in every case to set the regulator from 5.15 to 5.20. With this little trick you can reserve enough power from the power supply for your power hungry expansion devices, REU, RAMLink, SCPU, etc. So you can strengthen your power supply without having to build a whole new power supply.

Hardware

PAL NTSC Change the video standard of your C64/128

During the last few years, the 64 scene has become a lot more international than it was earlier in the 80's. At that time, people hardly looked across the big pond, but today, everyone's quite aware of the existence of two rather different C64 systems.

by Nicolas Welte

We're talking about the different video standards used in the U.S. and Europe - PAL-B versus NTSC-M. Problems occur, particularly when using games or demos written for one system and then used on the other system. The development of the GEOS update called "Wheels" was delayed as well, due to the differences between the two systems. First, there are different screen formats, a problem mostly games and demos struggle with. Secondly, the computers have slightly different clock rates, while those of the floppy drives are always the same. Some fast loaders can get confused with this. Furthermore, the realtime TOD clocks in the CIA's have different timings, although their timing does not depend on the video timings but on the frequency of the mains power. However, due to historic reasons, the frequency of the mains power is equal to the video frame rate. Unfortunately, the clock source of the CIA TOD clocks can't be adjusted without massive hardware changes.

If you're a developer and want to make sure your software is compatible, or if you want to run software written for one system with the hardware of the other, you might like to have both variants in your collection. It's simple for PC users; the latest CCS64 beta emulator lets you choose between three (!) different video timings. However, those who don't own a PC or don't want to use one will either have to import a C64 or convert one of their own. This article gives a detailed description of how to do such a conversion.

No matter what kind of 64 you're converting, you always need two parts - a VIC-II of the other video system, along with a quartz crystal of the other clock rate. The exact type numbers can be found in the table and in the list of parts at the end of this article. Luckily, the VIC is socketed in all 64's; however, the crystal has to be desoldered. Also, you'll have to manipulate a jumper.

Some theory

Just what is it that we're converting? To understand that, we have to understand how the different frequencies are generated in a C64. The crystal is the primary clock source, supplying a 17.734472 MHz clock on PAL systems and 14.31818 MHz on NTSC systems. On the one hand, the clock circuitry serves as the color clock, which is directly fed into the VIC-II, which in turn uses this signal to generate the color subcarrier frequency by dividing the color clock by four, and thus results in a 4.43 MHz signal on PAL systems and 3.58 MHz on NTSC. On the other hand, the signal is effectively divided by 18/8 (14/8), resulting in the dot clock (7.88 or 8.18 MHz). The dot clock is also fed into the VIC, where it's used to clock the internal pixel shift register, and divided by 8, which finally provides us with the system clock, phi0 (0.985 MHz on PAL, 1.02 MHz on NTSC). It's this system clock from which the VIC derives the whole video timing; namely the duration of each raster line and of each full frame. On PAL systems, a raster line ends after 63 cycles of phi0, after 65 on NTSC. A new frame begins every 312 raster lines on PAL or 263 on NTSC systems. There's even an old variant of the NTSC chip that uses 64 cycles per line and 262 lines per frame.

The conversion is the same for all models of the C64 and C128; however, there are some differences to note. First of all, you've got to find out the exact kind of computer you have. There are 8 different C64 boards, seven of which have an ID number listed in the table. 128 users are more lucky, as there are no more than two different types; the flat one and the D version in the plastic casing have the same board, while the C128DCR (the one in the metal housing) has the floppy board integrated on the system board (besides some other changes). I know of the existence of a C128CR prototype (like a C128DCR, just without the disk drive), but I don't know if it was ever produced. Also, I don't own any circuit plans of this machine. However, it should be possible to draw conclusions from the information available on the two other types of the 128.

Next, you look up the type number of the required VIC-II chip, based on the type of machine you own and the kind of conversion you want to make. Unfortunately, the 65xx and 85xx versions of the VIC-II require different voltages, making it impossible to exchange a 65xx with a 85xx. Also, the chips of the 65xx series are hard to get nowadays, because they were only made by Commodore until 1986; from 1987 they used the 85xx versions. On the other hand, the 65xx series offer a better picture quality, which makes them worth their cost. However, you should avoid two versions: the 6567 R5 (also called 6567 R5 6A) and the 6569 R1. These early versions don't run as stable as the later ones; they require some special adjustments on the board, and their picture looks worse. Even more, the video timing of the early 6567 deviates from its successors! Given these facts, I don't recommend converting a C64 which is equipped with one of these old VIC versions. The result won't satisfy you. C128 users don't have to make such choices, because there's only the 85xx variant of this video chip, whose pinout is incompatible with the C64 version, of course.

Where's the chip?

Now you'll have to locate the VIC-II inside your computer. If your 64 has a simple piece of cardboard with an aluminium layer as a shielding, you can simply fold it back, and you should find a somewhat larger metal box on the board. In very old models, this box has a cover that can be opened using a flat-blade screwdriver; alas, on other models, the box is made of a single part and even soldered into the board... So, your only rescue will be a strong soldering iron. Instead of the aluminium-covered cardboard, some 64's have a massive metal shielding which is held by some screws. If so, the VIC will have no extra casing. The newer 64s (after 1987) have no casing around the VIC, no matter what kind of shielding [illegible] contain. There are also a few 64's with a simple aluminium angle glued to the VIC as a heatsink. With a C128, the situation is similar. After removing the metal shielding, you'll find a metal box whose cover you have to open. Unfortunately, owners of a plastic C128D will have to work their way to the main board. First, remove the floppy drive's board, the plastic front, and then the drive and the power supply. After doing this, go on as with a 128DCR, which has the metal box without any further shielding, lying directly below the power supply. Remember to handle the power supply carefully. Hold it at its metal frame only; never touch any of its parts. Some models of the 128 have a switching power supply with no protection against touching. The high voltage warning on the back of these machines is there for good reason!

The changes

The next step is to take a look at the numbers printed on the 64 board in order to identify which kind of board you have; 128 owners will likely know it on first sight. With this information, you can look up the part number and location of the VIC-II chip in the table and carefully remove it from its socket. Now we come to the crystal. The old one is desoldered (which should pose no problem because it has only two connectors) and replaced with the new one. However, the jumper or solder point might be a little more difficult. On real old 64's there's a piece of wire that has to be removed and put in at another position. On newer models, it has to be either removed or put in, and on the C64E and all models of the C128, there's just a jumper point which is either open or closed. In order to close it, apply a little drop of solder on it; to open it, you either have to remove the solder on its surface or cut the narrow line between the two contact pads.

Now you're ready to install the new VIC-II - watch the alignment of the notch - and do the first test. Of course, you must use the right kind of monitor. The TV output likely won't function properly anymore. Modern TV sets should offer the best chances, because their video inputs mostly have multi-standard capabilities. Last, you have to fine-tune the frequency of the "color subcarrier" by adjusting the little trimmer until the picture looks good to you; it should be colorful, and adjoining lines in the power-up screen should have exactly the same color. Those who have the test screen program from 64er magazine can use it, of course. Please note that in the newer 64's and the 128, it's not recommended to use a metal screwdriver to adjust the trim capacitor; the adjustments would be difficult to make. You can use a metal screwdriver if need be, but you'll have to lift it off each time you change the adjustment to see what the screen looks like.

Finally, when the picture looks acceptable, you can put the machine together again, in the reverse order as you took it apart, and have fun with a C64 from a different world!

List of parts:

NTSC to PAL: quartz 17.734472 MHz, VIC-II 6569, 8565 or 8566, depending on model
PAL to NTSC: quartz 14.31818 MHz, VIC-II 6567, 8562 or 8564, depending on model

Sources:

1. Commodore C64 Service Manual, March 1992, 314001-03

2. Commodore C128/C128D Service Manual, November 1987, 314001-08

Model	PCB-ASSY	VIC-II Position	Trimmer Position	VIC-II PAL type no.	Jumper PAL	VIC-II NTSC type no.	Jumper NTSC
C64	326298	U19	R27	6569	E1-E2	6567	E1-E3
C64A	250407	U19	R27	6569	E1-E2	6567	E1-E3
C64B	250425	U19	CT1	6569	J2 closed	6567	J2 open
C64B-2	250441	U19	CT1	6569	J2 closed	6567	J2 open
C64B-3	250466	U19	CT1	6569	J2 closed	6567	J2 open
C64E	250469	U7	CT1	8565	J3 closed	8562	J3 open
C128/C128D	310379	U21	C20	8566	J1 closed	8564	J1 open
C128DCR	250477	U21	CT1	8566	J1 closed	8564	J1 open

Table: List of the technical details of the particular models

GO64! Bazar Form

Please insert the following advertisement in the next issue of GO64!

My advertisement is ☐ private/non-commercial (four (4) lines, 40 characters wide.)
US Dollar 3 (Euro 2.50) are ☐ included in cash ☐ included as cheque ☐ chargeable from my credit card (VISA, AMEX, Master Card)

☐ commercial (any reasonably sized ad, US Dollar 7 (Euro 6) plus taxes

The item(s) I am advertising about are related to: ☐ C64 ☐ C128 ☐ Software ☐ Accesories ☐ Misc

Address
CSW Verlag
Goethestraße 22
D-71364 Winnenden
Germany

I assure that I own all rights concerning my offers.

From:

Last Name/First Name

Street

ZIP/Town/State

Credit Card no./expire date/holder

Date/Signature

Masthead

Publishing Manager and Chief Editor: Enno Coners
Editorial Staff: Sebastian Bach, Günther Bauer, Rainer Buchty, Arndt Dettke, Wanja Gayk, Alexander Klock, Nikolaus Malecki, Nikolaus Metz, Volker Rust, Wolfram Sang, Sven Tewes
Editorial Assistance: Marco Baye, Helmut Bieg, Jolse Maginnis, Christoph Oehm, Errol Smith, Nicolas Welte
German Supervisor: Arndt Dettke
International Supervisor: Robert Bernardo, Nate Dannenberg
Translations: Bettina Beck, Matt Bodnarchuk, Stephan Lesch, Ranjith Ekanayake Mudiyanselage, Kendra Thiemann

Submitting Manuscripts: The editorial staff accepts most original Commodore-related manuscripts and programs. You must inform GO64!/Commodore World if any such programs or writings have been offered to be published or used commercially by a company other than CSW or one of it's subsidiaries. By sending in any manuscripts, programs or construction plans, the author agrees to allow the CSW Publishing House to print them. The author also agrees to allow any and all programs to be copied using whatever means is necessary (disk, online service, etc). Furthermore, CSW reserves the right to build and/or distribute any projects submitted to GO64!/Commodore World. The author shall assume no liability for damages or losses caused by the use of his/her programs or projects. Use any projects or programs contained herein at your own risk.

Layout: Günther
Cover Layout: ([illegible]r; photo by Bernhard Peball featuring Lilly 'Croft' Kretschmer
Graphics: Christoph [illegible], Sebastian Bach, Günther Bauer, Daniel Rac

Price for Advertisments: Price List No. 1 shall be considered valid unless specifically noted.

Publication: Monthly (12 issues per year).

Print: Berufsbildungswerk Waiblingen, Postfach 1129, 71301 Waiblingen, Germany

Copyright: All articles published in GO64!/Commodore World are protected by copyright. All rights, including translation and second commercialization, reserved. This publication may not be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the publishers. The original authors shall retain any and all rights with regard to specific terms, passwords, and the like, unless otherwise noted.

Liability: If GO64!/Commodore World gives false information or one of the published programs or plots contains errors, liability can only be considered in case of culpable negligence of one of the assistants.

Circulation Manager: Enno Coners

1999 CSW Verlag

Address of the Publisher: CSW Verlag, Goethestr. 22, D-71364 Winnenden, Germany
Telephone/FAX: +49/(0)7195 61120, E-Mail: go64@c64.org, cosowi@t-online.de

GO64! Bazar

Where can one buy or sell second-hand hardware, accessories, literature and software? Is there a convenient solution? Yes, there is! Check out our Classified Advertisements section, the GO64! Bazar! Above, you'll find a small registration form. Just clip it out, fill in the blanks, and send it to the publisher. For every advertisement you submit. For personal/private (non-commercial) advertisements, you may have up to four lines of text, each up to 40 characters in length. Please see the coupon for pricing and other details. We will make every effort to publish your ad in within the next issue of GO64!. If your ad doesn't make it by the next issue, it will appear in the issue following that.

For Free - Contact Corner

We're going to offer a free service to people who are looking for other C64 enthusiasts. Just send a contact advertisement in the form of a letter, FAX, or email to our editorial staff, care of: Contact Corner. The editorial staff reserves the right to shorten parts of letters which are not immediately important, in order to save space.

Commodore 64

For sale: SX64 + 2x C128D, 2 1541old + accesories. SX64: DM 800 C128D Tel. +49 (0) 2156/80160

Do you want to exchange data between your C64 and PC? It's simple with the X1541-/XE1541-cable from the PC-printerport to the C64-drive: +49 (0) 234/332887 (Stefan)

For free - Contact Corner

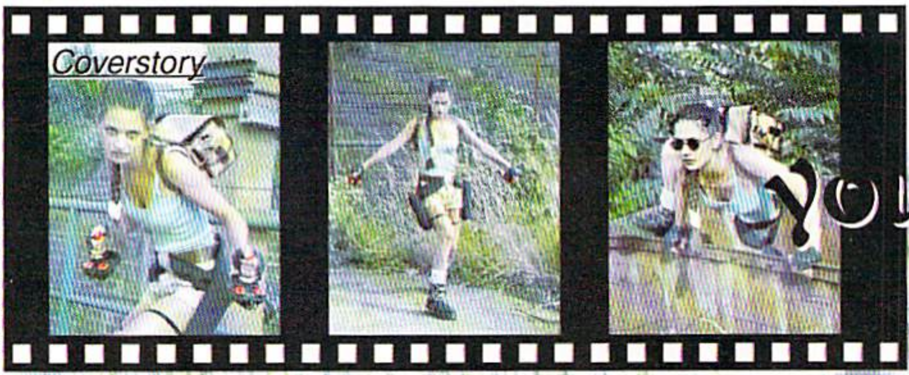
I am searching for contacts to swap C64 Games, Demos. 3.5 inch disks prefered! Tel.: +49 (0)202/504306

Searching for contact to other C64 users? I ca help you. Commodore Connection Line. Oliver Biasin, Schwalbenweg, 15 91220 Schnaittach, Germany. Tel/Fax +49 (0) 9153/1831. More than 75 entries.

I am offering information about traders and their products (and much more!). M. Kopetzky, Burgtreswitzer Str. 16, 92709 Moosbach, Germany, Tel./BTX: +49 (0) 9656-1772 (-0001), eMail: Martin.Kopetzky@t-online.de

The editors kindly want to draw your attention to the fact that offering, selling or distributing copyrighted software is only allowed for original programs and materials. For your own protection we ask our readers not to offer or to distribute pirated copies of any copyrighted program or materials.

The publishers reserve the right to refuse advertisements that appear illegal in nature, including advertisements dealing with pirated material. Original programs which carry a copyright will be conspicuously noted as such in their accompanying text. The label on the disk (or other media) will indicate any copyright as well.



Pictures: Bernhard Petschl (www.[illegible])

Shoot 'til your thumbs fall off

Action games - a timeless fascination. They have been popular ever since the beginning of the computer era. Starting with the forerunner of all action games, '**Space Invaders**', to the bestsellers of the 80s, like '**Katakis**' and later on '**Crush**', they share a never-changing and straightforward game concept - shoot 'til your thumbs fall off.

by Helmut "Katakis" Bieg and Volker Rust
Technical Support:
Christoph "Catweazle 6510" Oehm

It is just this basic simplicity that makes 'shoot'em up' games so popular. Who really wants to read endless manuals and bother with thousands of features all the time? Here we'd like to give you a short survey of some of the games without which playing would only be half the fun.

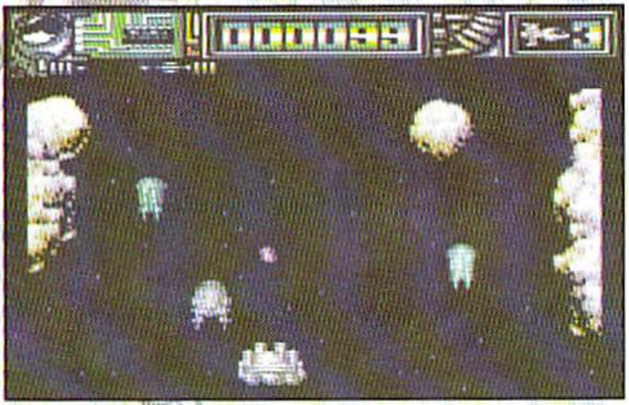
The forerunner of all action games - maybe even the first of all - was '**Space Invaders**'. As shown by its innumerable clones, like '**Galaga**', '**Bandits**' and later on '**Star Force**' and its successor '**Mega Star Force**', this game concept has proved to be of unsurpassed popularity; a spaceship at the bottom half of the screen has to destroy groups of enemies that occupy the upper half of the screen. New features, like extra weapons or barriers to shield the player's sprite, were refined over the years and added to the concept's possibilities. 'Starforce' stands out, due to the metallic look in its graphics and its great sound effects.

'Moon Patrol' and its successors

One of the first scrolling 'shoot'em up' games was '**Moon Patrol**' by Atari in 1983. Even this early, the game had a kind of pseudo-parallax scrolling, though its graphics showed a lot of single pixels. The game concept (a moon buggy drives from left to right and has to evade potholes, rocks, mines and attacking UFOs) is extremely simple but indisputably brilliant and exciting enough to be fun to play even after all these years. Another nice idea is the mountains you have to drive through. As early as 1983, 'Moon Patrol' had music in the background, and thus music became common in games from then on.

'**Battle Valley**' in a way is an evolution of this game concept. Players choose to direct either a tank or a helicopter through desert landscape, enemy fire, and the like. With the helicopter, you have to collect pontoons in order to repair bridges in the tank's way. Considering the graphic design, the game is rather average, and the music by 'Maniacs of Noise' gets on your nerves in no time. However, the game concept offers some good ideas and is good fun, though it is a little tricky.

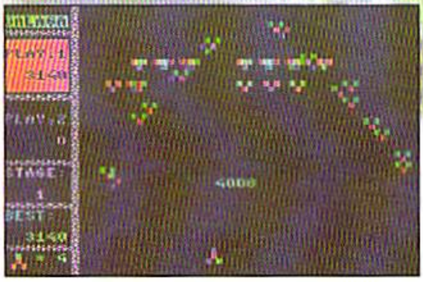
'**Silkworm**' by Sales Curve in 1988 has much less strategy but a lot more shooting. Missions have to be fulfilled either with a helicopter or an armored car and consist of fighting against attacking formations in levels that scroll horizontally. We especially like the two-player mode which is good for some entertaining side-by-side battles. Unfortunately, some of the enemies return a little too often, a fact that naturally diminishes the good impression the design left. 'Silkworm's successor, '**Swiv**', is not as convincing; it still keeps the two-player mode, but the perspective has changed to a bird's eye view, scrolls vertically, and strongly recalls games designed with the S.E.U.C.K. ('Shoot'Em Up Construction Kit').



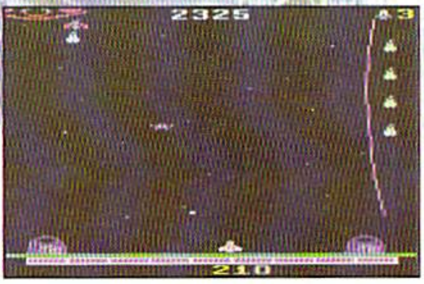
Dominator

Vertical Scrollers

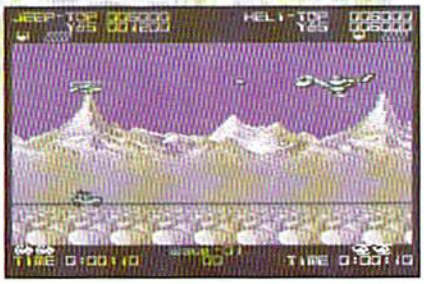
There are rather few vertically-scrolling games for the C64, at least as far as high quality is concerned. Apparently, the S.E.U.C.K. editor deterred many programmers from designing games with a similar game design, and thus the result is that there are very few pearls in this field. In 1986 '**Lightforce**' was released, in its day a very well-designed game with excellent music by Rob Hubbard. Three years later, System 3 proved that it is



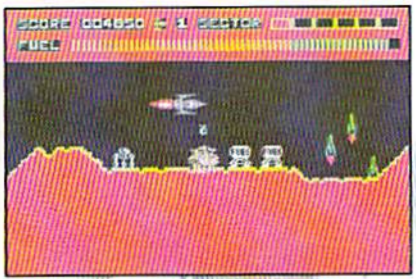
Galaga



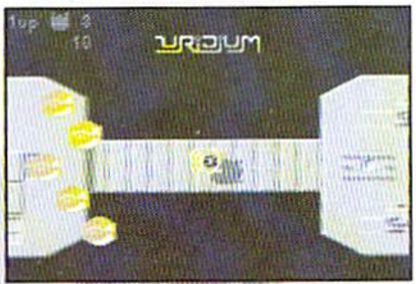
Bandits



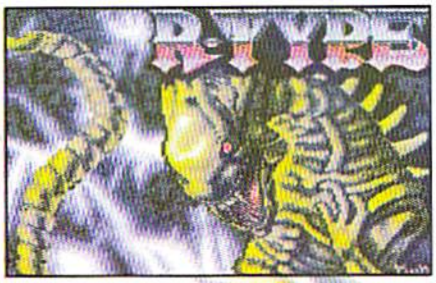
Silkworm



Skramble



Uridium



R-Type

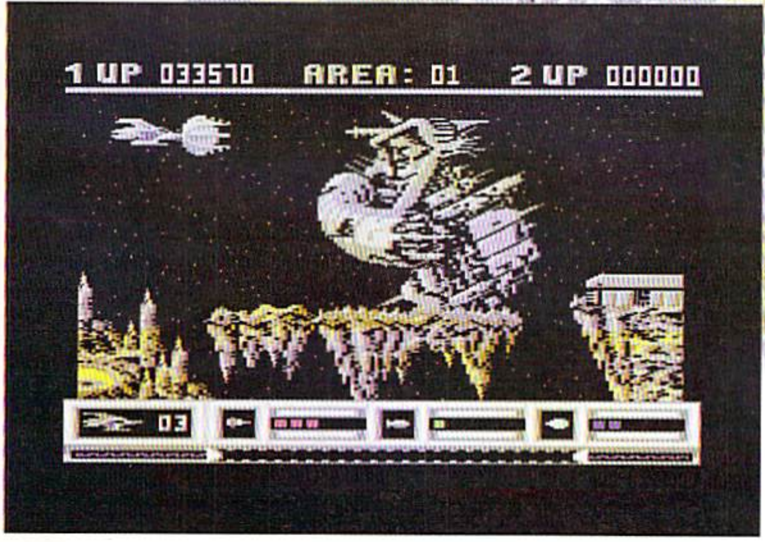
always possible to be better; **'Dominator'** matches the latest games in graphic artwork, but you have to be content with only sound effects instead of music during the game. Nevertheless, if you compare the various vertical scrollers with each other, 'Dominator' is still the creme de la creme. Hannes Sommer of Cosmos Designs released **'Moons'** in 1992, the first vertical scroller with 16-color scrolling and a game concept in between 'Space Invaders' and 'Lightforce', but regrettably there are some quite difficult parts, and the colored artwork sometimes seems a little out-of-place and oddly pieced together. For your ears, there's a mixture of sound effects, music, and short jingles which seem to be a bit arbitrarily chosen.

Horizontal Scrollers

The first serious horizontal scroller with a controllable spaceship appeared on the market in 1983. **'Scramble'** by Anirog Software does not scroll really smoothly, but instead it offers intelligent gameplay with a slowly rising degree of difficulty. Players have to fly over a moon landscape, destroy enemy rockets, and bomb fuel tanks. The game is a fascinating opportunity to examine the beginnings of 'shoot'em up' games, and the simple means with which a stimulating game can be designed.

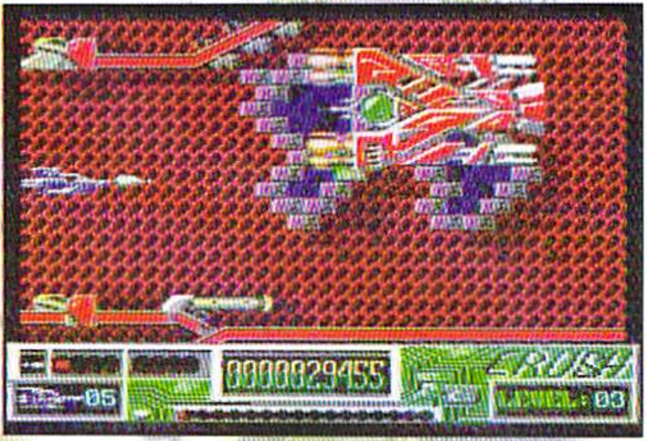
'Uridium' by Andrew Braybrook was released in 1986 by Hewson and has a most interesting story. Players have to leapfrog giant enemy spaceships, land on them, and finally destroy them. 'Uridium' was the first game in which flight directions - either left or right - were freely selectible. The idea, too, was completely new, and the game has a very high speed. The dissolving of the giant spaceships when destroyed is preprogrammed. There are lots of challenging levels, but overall, the game is a little too tricky. The negative sides are a lack of music (good sound effects instead) and sprites that are a little monotonous, but it should not be forgotten that the game has been around for some years and still is very exciting.

'R-Type' by Irem (1987) is the first action game Manfred Trenz and Andreas Escher designed for the C64. The first thing you'll notice is the great title track by Chris Huelsbeck. The in-game music by Ramiro Vaca fits very well with the mood, too; in comparison to the original arcade machine music, the tempo has gone up a little. Simultaneous sound effects are added, which enhance the feeling of taking part in a real shoot-out. Nevertheless, the gameplay could be improved. The spaceship moves too slowly and needs a speed-up extra to be real fun to fly. After a while, it becomes apparent that there is a deplorable lack of new ideas in the design of the different levels. Some good ideas that make playing less monotonous should not go unmentioned, e.g., a giant spaceship in level 3, though it moves a little too slowly. Again, the graphics are rather excellent, but at times, the compound sprites move unnaturally, especially when meeting the final adversaries. Overall, 'R-Type' is convincing as a good game, but from level 5 on, it becomes much too difficult.



Katakis

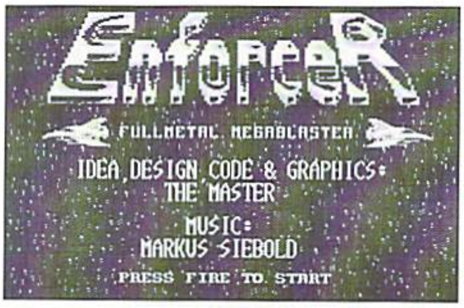
Another representative of the horizontal scrollers genre is **'Into Oblivion/IO'** by Firebird (1987). The metallic design looks great. There's no in-game music but very good sound effects. The concept of the game puts a strong emphasis on strategy. Thus, it's no use to continually fire, because the groups of enemies return in ever-changing formations, sometimes in circular form, sometimes in that of a sinus curve. Strikingly, the whole game consists of only one file. From an overall point of view, the degree of difficulty of the game is too high; there is every reason to count it among the hardest 'shoot'em up' games. Moreover, the extra weapons system is not the most reliable (extras change when ship is hit).



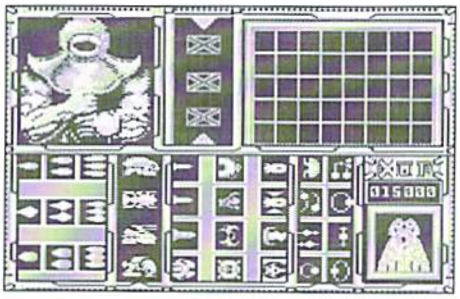
Crush

One of the most important games of all time seems to be **'Katakis'** by Manfred Trenz and Andreas Escher, released by Rainbow Arts in 1987. There has been no better design of levels ever since its release. It's just this very point in which so many of the next generation action games fail; even the brand-new 'Crush' cannot compete with 'Katakis'. There's rarely a game more sophisticated than 'Katakis', one of the most well-planned games of all time. The gameplay is just brilliant and offers lots of fun, even after the fiftieth play. Sprites and graphics look magnificent and have lots of variation. The space glider's weapons can be enhanced in various levels; different extra weapons and

Armalyte



Enforcer



X-Out

guided missiles can be added. Moreover, there's the famous super fire. The soundtrack definitely is one of the best ever composed by Chris Huelsbeck. It fits into the atmosphere of the different game scenes and is accompanied by sound effects. Even the fast load by Heureka Teachware leaves a positive impression; it's one of the fastest ever on the C64, and it takes only five seconds to load a level. Though the collision control is not all too accurate, a positive side effect is that exact maneuvering isn't essential in situations where your back is to the wall.

Yet, a game of such high quality as 'Katakis' has a few weak points, even if they're only technical flaws. The final adversaries are composed of more than one sprite and tend to flicker. The same problem can also be observed with other enemy sprites. If too many sprites are gathered on the screen, at times the multiplexer suffers from an overload, and the scrolling that normally was very smooth starts jolting for a few seconds. But in general, 'Katakis' is the one and only game all others are compared with, and its artwork still has not met its master.

'Nemesis', **'Salamander'**, and **'Delta'** (1987-88) are all based on the same system of extra weapons, that is, the 'collect-as-many-points-as-you-can-and-then-pick-an-extra-weapon-from-the-menu-at-the-bottom-the-more-points-you-have-the-cooler-the-weapon-you-get-when-moving-the-joystick' principle. Although this might not be the best solution, it certainly has produced a lot of imitations. However, these three games are typical representatives of their type, horizontally-scrolling games, with 'Nemesis' being the best, because it is very varied and has secret levels as well as the best artwork. 'Salamander' has good implementation but has some flaws in the gameplay, e.g. a level design that is not all too interesting. 'Delta' suffers from a lack of color but is fun to play anyway, although its successor 'Armalyte' is way better. One year after its predecessor, **'Armalyte'** was released in 1988 by Thalamus. The whole design recalls a metallic look and is completely acceptable, if not a bit boring after a while. The levels are designed in a fair and balanced way but require a certain strategic talent in some parts. They are never truly unfair, but the level of trickiness remains high throughout the whole game. One negative feature is a lack of music, though replaced by good sound effects that also have a metallic sound. Obviously, the shots were designed with less care, consisting of too many pixels. It is possible to improve the fire, not in the sense of 'Katakis' or '**Enforcer**' with varied systems of shots, but with a tail gun or guns on the upper or lower side of the ship. The game's sprite multiplexer has the positive effect of completely preventing any flickering or jolting.

1989 seemed to be the year of the dragon; dragons took the leading parts in two of the year's action games. First, **'Dragon Breed'** by Irem offers a good audio sample in the intro, but its gameplay is a little poor, and it clearly occupies a place behind games like 'Katakis' or 'R-Type'. What's completely new is the 5-way scrolling that makes moves up or down possible. **'Saint Dragon'** by Sales Curves returns to traditional values of the horizontal scroller genre and has a lot in stock for your eyes and ears. Unfortunately, the game has no sound effects and only one piece of music, though it's a good one. The design is absolutely new and independent from any of its predecessors, giving a new charm to the traditional landscapes of 'shoot'em up' games. Image Works also made a good start with **'Phobia'** from 1989; the graphics show a lot of care, and for the first time, players have the option to choose in which level to continue after the completion of another level. On the other hand, there's no music and the effects are a little rough. Moreover, the level design is by no means perfect.

From 1989 as well, **'X-Out'** by Rainbow Arts has a very nice pixeled intro that puts players on the bottom of the sea. Andreas Escher's in-game graphic artwork is of very high quality and never becomes boring. Another positive is the design of the levels, a well-balanced mixture with a little bit of everything. Something not yet seen before in any other action game is the shop where parts of ships can be purchased. The points in your score serve as currency. This new invention allows you to individually add to your glider's abilities and add satellites and extra fire. Overlay sprites were used very successfully to improve the look of the spaceships. Music and sound effects by Chris Huelsbeck were produced by FAME and again manage to keep their high quality.

But on a technical level, 'X-Out' belongs to the games of lower quality. The limited 5-way scrolling jolts a bit, and a large amount of sprites on the screen makes the scene look like it was filmed in slow motion. The final adversaries look good and surely represent a challenge, but similar to 'Katakis', they flicker from time to time. The compound sprites sometimes show cracks. At times, it is hard to fight the feeling that someone forgot the fast load in this game, because loading one level can take up to two minutes.

In 1991 CP released two real 'shoot'em up' free-for-all's at once. **'Lethal Zone'** on Golden Disc is a game that impresses with an independent design and nice artwork. It has sound effects and music playing at the same time during the game. Unfortunately, it is very hard to play and therefore not very exciting, not easy to use without the training mode. Released by the Game On 12/91 disk magazine, **'Plural'** by Cosmos Designs is at once recognizable as a high-quality clone of 'Armalyte'. Its graphics strongly resemble that of its model, though it is not as good. The option for two-player mode is an extra motivation. However, it has to be mentioned that the game is extremely difficult, and at times when many enemies are present, it is rather confusing to determine which was hit and which was not, because the explosions of enemies continue along their former trajectories. The extra weapons system has been improved in comparison to its predecessor; spheres open when hit and reveal certain extras. They are placed in a way that insures a constant improvement of fire. But again, there is no in-game music. 'Plural' is one of the least widely known games by Hannes Sommer but surely one of his best.

From Italy - 'til then known to be more or less one of the 'Third World' countries of programming - **'Catalypse'** by Genias appeared on the market in 1992. It offers a good intro which puts players in the right mood for a real 'High Noon', though the speech samples are nearly indecipherable. There are certain similarities with 'Armalyte', but they can easily be overlooked. Only the fourth level is a drawback in design, because the background consists solely of a simple grating. Instead, great final adversaries and the last sequence compensate for that.

In the same year, Manfred Trenz, alias 'The Master', was back on the scene. After virtually ending his programming career for C64s with **'Turrican II'**, he produced another action game classic called 'Enforcer' (CP). The graphics are in part recycled from earlier projects, but the game has enough new ideas to clearly distinguish it from its predecessors. For the first time, Manfred Trenz uses a concept of energy charge for his glider so that collisions with enemies do not necessarily result in a loss of life each time. Markus Siebold's music seems a little more neutral than the music in earlier games by Trenz but nevertheless captures the mood and is accompanied by excellent sound effects. Regrettably, the game has only a length of six levels.

'Parsec' by X-Ample was released in 1993 by CP and starts off with a nice intro. It, too, is a successor of 'Armalyte', and at times its graphic design is even better. One criticism would be that it is composed of three tough, relatively long, shooting levels only, followed each by a level that requires high reaction speed. We miss satellites to shield the player's ship, but the fantastic music by Thomas Detert makes up for a lot.

The latest game in the genre of the horizontal scrollers is **'Crush'** by AEG/Smash Designs (1999), one of the high-quality 'shoot'em up' games in the lavish Katakis-style. There's not much left to be said after the thorough test in our June issue, only that it is definitely not the best action game ever. It's 'just' a very good one.

Jump'n'Shoots

One of the forerunners of the Jump'n'Shoot genre undoubtedly is **'Hawkeye'** by Thalamus (1988). It has fine graphic artwork, big final adversaries, different types of weapons, and parallax scrolling along with great music by Jeroen Tel. The animation of the hero is a real success, but the game is not all too exciting to play for long time. The concept of the game (player has to run through levels and collect items) has been copied more than once and perfected afterwards by others. Unfortunately, the game is not available for the C128.

'Darksword' was released in 1990 by CP and is a kind of minimized variant of 'Hawkeye'. Players have to cover three levels where items have to be collected and enemies killed, all this without any sound effects, though replaced by music.

'Turrican I & II' (1990-91) by Manfred Trenz/Andreas Escher/Rainbow Arts are two of the most style-defining action games there are. **'Turrican'** is the first game ever with perfect 8-way scrolling. Its levels are next to endless; it takes weeks to discover their last depths and to collect the last diamonds, so that exploring the 'Turrican' world never loses its fascination, in particular because the design is brilliant and manages to create a very absorbing atmosphere. Graphics and sounds are excellent, even though there is no in-game music, which would most probably have exceeded the memory capacity anyway. There are a few pieces of music that are true masterpieces. In the middle of the game, in levels 3.1 and 3.3, the hero's path is predetermined (vertical scrolling, Turrican flies by means of the jetpack first up and then down), which provides variety and permits music.

'Turrican II' offers even more adversaries and great level design with three Katakis-style flying levels as extra worlds that have parallax scrolling. If you compare the flying levels on the C64 or the Amiga, you will find they have been implemented even better in the breadbox, and the game even has a multiplexer. The last level's structure strongly resembles that of level 3.1 in 'Turrican I' (jetpack), and again music and a giant final adversary are waiting for you.

At times the game suffers from graphics errors that in very few cases can even disturb the game's course. It doesn't have the fastest load, but in every other respect, the game is absolutely fantastic (the player's sprite looks especially great) and has lots of different extra weapons, enemies, and giant levels for the action game fan.

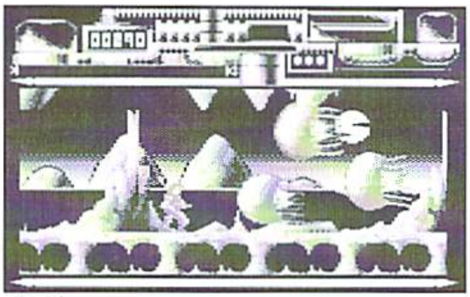
Another famous clone of the Hawkeye's game concept is **'Another World'**, which was released on the Golden Disc of 1991 and should not be confused with the game of the same name for Amiga and PCs. The gameplay has been amplified; players have to collect crystals which are needed to complete the construction of teleports that lead to other levels or to other regions of the same level. The graphics are appropriate, considering the techniques used at the time (overlay-sprites), and even the background design looks great.

What's strange is the frequency with which the game crashes, a flaw that the manual explains by 'high technical demands'. Doubt remains whether a better solution could not have been found.

'Rubicon' by 21st Century Entertainment in 1992 was seen as an unofficial successor of 'Hawkeye'. It does indeed have an intro with

Lethal Zone

Plural


Rubicon

excellent digital music by Jeroen Tel. The game itself offers numerous, very finely pixeled adversaries and landscapes with parallax scrolling. Nearly all of the monsters are animated very well and even the monsters in the middle of the different levels compare favorably with the final adversaries of other games. Unfortunately, the weapons system was designed with less care; only four variants of fire exist. Scrolling is only possible in one direction. But on the other hand, it has simultaneous music and sound effects during the game, and the design of the levels is very good.

It was only in the following year that two more games of a similar concept were released; **'Greystorm'** (CP) and **'Genloc'** (Markt & Technik) were exact copies of the Hawkeye's game concept, though they were technically of much higher quality. Unfortunately, 'Genloc' in particular is much too difficult so that playing without a cheat poke is nearly not possible. X-Ample should have paid as much attention to that as to the perfect technical implementation. In 'Greystorm', more thought was put into the gameplay (with cooperation of X-Ample). Its levels have a sensible level of difficulty and are not simply plastered with enemies.

Game	Producer	Released in	Genre	Difficulty	Two-player-mode	Music-SFX	Special features	Grade percentage
Another World	CP-Verlag	1991	Jump & Shoot	medium	no	only SFX	16-color scrolling; uses floppy drive RAM as graphics memory	72%
Armalyte	Thalamus	1988	Horizontal scroller	high	yes	only SFX	Metallic look and sound; good Multiplexer	82%
Catalypse	Genias	1992	Horizontal scroller	high	no	yes	giant final adversaries	78%
Crush	Smash Designs	1999	Horizontal scroller	high	no	yes	strong resemblances with 'Katakis; nice intro; minor bugs	79%
Dominator	System 3	1989	Vertical Scroller	high	no	only SFX	good graphics	75%
Enforcer	CP-Verlag	1992	Horizontal scroller	low	no	yes	typical representative of the Trenz-style	85%
Genloc	Markt&Technik	1993	Jump & Shoot	high	no	only music	fantastic parallax scrolling; extremely unfair gameplay	73%
Greystorm	CP-Verlag	1993	Jump & Shoot	high	no	yes	yes very nice intro, parallax scrolling, funky digital music	81%
Hawkeye	Thalamus	1988	Jump & Shoot	medium	no	yes	not available for C128, great title track by Jeroen Tel	78%
Into Oblivion	Firebird	1987	Horizontal scroller	high	no	only SFX	one file for four levels	79%
Katakis	Rainbow Arts	1987	Horizontal scroller	rising	yes	yes	great gameplay, graphics, and soundtrack	92%
Lethal Zone	CP-Verlag	1991	Horizontal scroller	high	no	yes	good graphic artwork; flaws in the gameplay	73%
Parsec	CP-Verlag	1993	Horizontal scroller	high	no	only music	alternating shooting and reaction levels	75%
R-Type	Irem	1987	Horizontal scroller	high	no	yes	great soundtrack by Ramiro Vaca	78%
Rubicon	21st Century Entertainment	1992	Jump & Shoot	medium	no	yes	parallax scrolling; lots of entertainment & giant monsters	82%
Saint Dragon	Sales Curve	1989	Horizontal scroller	medium	no	only music	true mood music	76%
Silkworm	Sales Curve	1988	Horizontal scroller	medium	yes	only SFX	good gameplay in the 2-player mode	70%
Turrican	Rainbow Arts	1990	Jump & Shoot	medium	no	partly	giant levels and final adversaries; perfect 8-way scrolling	92%
Turrican II	Rainbow Arts	1991	Jump & Shoot	medium	no	partly	parallax scrolling in 'Katakis'-style levels	95%
Uridium	Hewson	1986	Horizontal scroller	high	no	only SFX	a trueclassic of the 'shoot'em up' genre by Andrew Braybrook	74%
X-Out	Rainbow Arts	1989	Horizontal scroller	medium	no	yes	5-way scrolling; technical flaws	77%

A BASIC PROJECT

Dealing with tables, Boolean mathematics, and the division with and without remainder

You are cordially invited to GO64!'s second BASIC project! Welcome! Just like the first project (which you can read in the GO64! issues 10 to 12/98), this project was started in response to a reader's demand. In issue 4/99, Hansjoerg Meduna asked on page 31 for a calendar program, one of the kind that provides us with the day of the week of a given date. Your wish is our command!

by Arndt Dettke

A Look Ahead

At the same time, this issue is a wonderful opportunity to add a real application to the menu system of the first project. Thus, you will be shown how the mere menu handling of BP1, that served no other purpose, can be turned into a real application -- and what you have to be on the look-out for. In addition, the calendar theme makes it necessary for us to take a look at tables. In order to move in interdependent tables, a programmer often has to determine in which line and column a value resides, so that he may find the corresponding value in the next table. Here you will see how this problem is solved. To do that, we need the division with integer results (other computer languages have the operator DIV for that), and we need to determine the remainder of a division (this is called "modulo" or MOD). Very often, we will also have to deal with truth values (is the comparison correct, or is it not correct?) that are displayed in the computer as numbers and that can therefore be used for the computation of other numbers (Boolean mathematics). You see, once more we have made big plans, haven't we?

A Look Back

Unfortunately, C64 BASIC has the unpleasant characteristic that variables are valid everywhere in the program, even where it isn't useful at all, for instance in subroutines. Therefore, variable names have to be chosen in such a manner that they don't come into conflict with other parts in the program, where the values they currently hold may be dumped for some other reason. In BP1 I had proposed this reason -- to structure your programs modularly and to use just the few variables necessary. We then gathered a basic selection of program modules, which was supposed to give us all the freedom possible for further programs. In order to avoid wrecking our current plans for a calendar program, we have to know which variables are already used by the menu system and in what function. In the box "Variables of the Menu System", you will find all modules from BP1 together with their purpose and the variables used. With this, we have then taken care of the prerequisites for our BP2.

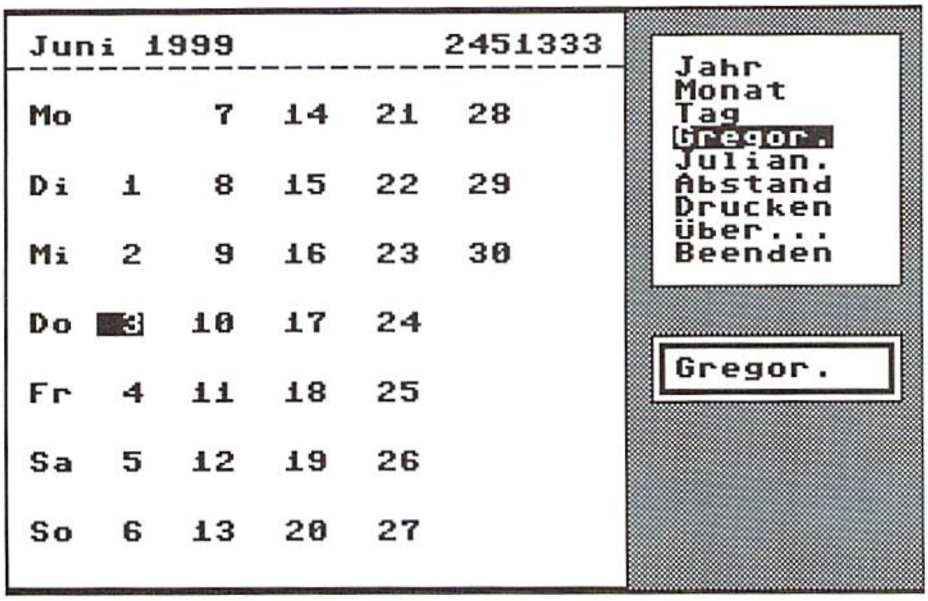
The Plan

Our calendar program won't be a perfect application; it won't supersede already existing calendar programs. First of all, we will forego a printing option which would seriously distract from the real subject and and which would be more suitable for an item in a future BASIC project. Then we will limit our display to a single month and not try to get an entire year on the screen all at once. We also leave out the determination of holidays, because otherwise, the display routine would have to be substantially more complex than I have in mind here. After all, our project is supposed to induce you to take up the programming pen yourself. We have to leave you something demanding to do, I think. Once finished, our program is supposed to look as shown in picture 1.

To the left there is a big box for the chosen month; to the right the only menu -- a status display; and beneath that, if required, an input box. We will work with the font designed by Ralph Schmieder, which can be used in a very simple way to simulate a graphical-looking environment like GEOS's. Detailed information on that (char table and so on) can be found in issue 12/98. The long number at the top and to the right in the month box is the so-called Julian date for June 3, 1999 -- something like the "stardate" in Star Trek. From that we will calculate the number of days between two dates.

A Little Bit of Calendar Trivia

It would be so wonderfully simple, if the earth exactly spun around itself 365 times during a year. Then the year and the day would perfectly fit, and there wouldn't be all that computing. But neither month nor year are integer multiples of the basic unit day -- just the other way around. There are manyfold ways to determine the length of a year exactly. Currently in astronomy, the so-called tropical year or solar year is



```
Juni 1999                2451333
--------------------------------
Mo        7   14   21   28

Di    1   8   15   22   29

Mi    2   9   16   23   30

Do    3   10  17   24

Fr    4   11  18   25

Sa    5   12  19   26

So    6   13  20   27
```

commonly used. It is defined by the point of time when the sun passes the middle spring point on her apparent path over the celestial sphere. From one earthly revolution to the next, one counts exactly 365 days, 5 hours, 48 minutes and 46.98 seconds; or (decimal) 365.2422 average solar days.

Ever since mankind has been able to do agriculture, it has also created calendars in order to begin farming at the best time possible. At first, the moon was used to measure time (in the area of oriental belief, it is this way even today). Then the ancient Greeks invented a mixture between the lunar and the solar year, which had the flaw of quite arbitrarily adding entire leap months. The Roman, Julius Caesar, ended this chaos and introduced a calendar that was made for 365.25 days, the Julian calendar. It was adopted by the Christian church and had been valid in Catholic areas until October 4, 1582. Protestant regions all over the earth changed only very much later to the next and currently most commonly used calendar, the Gregorian (named after the pope who had ordered its design and later introduced it).

The Gregorian calendar calculates the year as 365.2425 days, nearly exactly matching the tropical year. In the year 4915, it will be one day off only 3,333 years after its introduction. On the day of the Gregorian calendar's introduction, the people introducing it made 10 complete days officially disappear from that year. Thursday, October 5, 1582 was followed by Friday, October 15 -- the Julian calendar had already been off that many days. By the way, this is also the reason why we are talking about the "October Revolution"; Russia still used to use the Julian calendar then (here we already had November).

The Gregorian calendar works especially well, because it has a cleverly designed system of leap days built in. Inserted every fourth year into February, leap day comes in the years that can be divided by 4 without a remainder. The exception to this rule are the full centuries; for instance, the year 1900 is no leap year. This is known to few people. And very, very few have heard about the fact that all those full centuries which can be divided by 400 without a remainder are leap years, like the coming year 2000! You see, now you are a member of a small group of chosen wise ones, and this only because you are reading GO64!. Now, is this something, or what?

All right. Let's enter:

```
5500 sj=(j/400)=(INT(j/400)):IF sj THEN 5515
5505 sj=(j/100)=(INT(j/100)):IF sj THEN sj=NOT sj: GOTO 5515
5510 sj=(j/4)=(INT(j/4))
5515 IF sj THEN PRINT" leap year":END
5520 PRINT "not a leap year":END
```

And to check it out, let's try the following:

```
j=1999: GOTO 5500
```

The result is going to be the answer, "not a leap year". 1999 isn't one, is it? Change the number behind "j=" and toy around a bit with it. The answer is going to be always correct, as long as you don't go below the "magical" year of 1582. Before that, the Julian rules were valid, and the answers of our algorithm would be wrong. And how does that exactly work? Hehe, I have already tricked you into using some operations of Boolean mathematics and calculating with remainders. Just look at the value the variable sj has taken after every trial run. You will see that there will only be two values given: zero or minus one! If it's zero, then it has not been a leap year; whereas if it's minus one, it has been a leap year. Those two values have the same meaning, as if we said yes (-1) or no (0)! Thus, our algorithm provides for the question, "Is xxxx a leap year?" The answer -- "Yes, it is" or "No, it isn't". This answer is conveyed in the variable sj.

A Little Mathematical Trivia

Have you been marvelling about the two equals signs in the BASIC lines given above? I hope not, because the comparison between the two values is dealt with during variable allocation in the same way as if it stood behind the command IF. Therefore, we could also have written:

```
5500 IF (j/400)=(INT(j/400)) THEN 5515
```

According to the other lines, why should we do this way anyway? Hm, I do want to get into this a little bit further with you, and later on, it would be advantageous if the information "leap year yes or no" is available at any place in the program. Thus, these BASIC lines will end up in our finished program nearly just the way they currently are. In addition, you now know that during comparisons in BASIC, a numeric value is created, 0 or -1. These numbers are called truth values or Boolean values (named after their inventor), and you may calculate with them.

"Division with and without remainder" -- this has also been already used in the example above. Unfortunately, our BASIC has no modulo function, which would directly provide you with the remainder of a division, making everything much easier. Instead of 1999 MOD 4 (with a result of 3), we have to help ourselves with another function that has actually nothing to do with division at all, namely INT. You probably know that INT will turn a floating point number (e.g. "10.4") into an integer (yes, that's INTeger); INT(10.4) returns 10. In the process, everything behind the decimal point is cut off. The same also happens if the function is given a division as the argument; INT(1999/4) therefore returns 499 and not the more exact value of 499.75. We take advantage of this very fact in the example above. On the one hand, we divide the year correctly without manipulating the result; on the other hand, we turn the result into an integer value. Then we compare both results in order to find out if they differ. If the result is the same in both instances (which will only happen if the number can be divided by 4 without remainder), then we can say with

Table I and II Months						
10 1	7 4 1	11 3 2	8 2	5	6	12 9
6	5	2	1	7	3	4
7	6	3	2	1	4	5
1	7	4	3	2	5	6
2	1	5	4	3	6	7
3	2	6	5	4	7	1
4	3	7	6	5	1	2
5	4	1	7	6	2	3

Table III Years of a Century																	
0	6		17	23	28	34		45	51	56	62		73	79	84	90	
1	7	12	18		29	35	40	46		57	63	68	74		85	91	96
2		13	19	24	30		41	47	52	58		69	75	80	86		97
3	8	14		25	31	36	42		53	59	64	70		81	87	92	98
	9	15	20	26		37	43	48	54		65	71	76	82		93	99
4	10		21	27	32	38		49	55	60	66		77	83	88	94	
5	11	16	22		33	39	44	50		61	67	72	78		89	95	

Table V						
7	6	5	4	3	2	1
1	7	6	5	4	3	2
2	1	7	6	5	4	3
3	2	1	7	6	5	4
4	3	2	1	7	6	5
5	4	3	2	1	7	6
6	5	4	3	2	1	7

Table IV Centuries							
julian (until 1581)	0	1	2	3	4	5	6
	7	8	9	10	11	12	13
	14	15	16	17	18	19	20
	21	22	23	24	25	26	27
gregorian (since 1581)					15	16	
	17		18		19	20	
	21		22		23	24	
	25		26		27	28	
	29		30		31	32	

Table of Days					
Sunday	1	8	15	22	29
Monday	2	9	16	23	30
Tuesday	3	10	17	24	31
Wednesday	4	11	18	25	
Thursday	5	12	19	26	
Friday	6	13	20	27	
Saturday	7	14	21	28	

certainty that the examined year is a leap year (accordingly with divisions by 100 and 400).

Once again, in other words, j/4 = INT(j/4) therefore checks if the number j can be divided by 4 without remainder. If so, the Boolean value -1 (yes) is returned; otherwise, there will be 0 (no) instead. By the way, the operator NOT in line 5505 turns the truth statement around -- "not yes" equals "no" and "not no" equals "yes". That does make sense, doesn't it? Yes, indeed. So much (for now) for the algorithmical side of our problem.

A Little Trivia about Tables

The most simple form of a table is the list, for example my shopping list this morning. Quite neatly, line after line, one entry after the other (ouch, this is gonna be expensive!). Okay. Lists are called "arrays" or "field variables" in BASIC. I stick to the word "list", because with that I know what is meant. For our program we will be creating several lists, for instance, for the names of the days of the week or for the names of the months. The good thing about a list is that it reflects a certain order; every line of a list contains a certain piece of information. If you structure other lists accordingly, you can therefore complete information from one list with information from the same location on another list. We will do this with the months with different lengths. In a second list, we will store the number of days the corresponding month has.

But sometimes, you may wish to store several pieces dealing with the same item instantly and directly. For this, an actual table is more advantageous; you can think of it as having a list created on every line of the original list. We have already used such a list of lists ("table") in BP1; the different graphical chars that defined a box had been made available in a table with four lines (for the four different frame designs) and nine columns (for the nine chars, which make up every box). We also stored the menus and submenus in a table.

In BASIC, tables are created with the command DIM. DIM is followed by the name of the list or table, its type (text, integer or floating point), and in brackets its design -- in effect the number of lines in a list and if needed the columns of a table. DIM mo$(12), for instance, would create a list for the names of the 12 months; and DIM mn$(mn,sm), a table for the names of the menus. You can also create further structures, for instance, a table in three dimensions: line, column, and plane, of what one has to think as being similar to a cube. But these multi-dimensional structures tend to be impractical, so you should try to avoid them.

Back to the Plan

Now, how do I find out what day of the week March 14, 1997 is? (This was the day when the first issue of the GO64! hit my home! It had been a Friday.) Or how about August 6, 1945? (The day, when the A-Bomb was dropped. A Monday.) Quite a lot of mathematics, isn't it? You would need to know which day of the week is the first day of the Gregorian calendar (a Friday), and then you would have to add up all the months up to the desired date, taking into consideration the leap years. Is there no easier way to do it? I have also asked this myself, and I instantly looked for something appropriate on the Internet, key word "Ewiger Kalender" (Eternal Calendar). And I found something right away (what else?). On the page

http://home.netsurf.de/stefan.grussling/zeit/ewiger_kalender.htm

you can find exactly what we are programming here as well -- an eternal calendar that is based on the fast-consulting Sigmaof corresponding tables. Exactly what's best needed for this job!

I take this occasion to gladly thank Stefan Grussling for our being allowed to use his tables here. He has taken them from an old nature encyclopedia from the '50s. Unfortunately, you do not find such things in encyclopedias nowadays anymore. Stefan used five tables; we are copying in a slightly modified way by first making them six (by splitting the second) and then dropping the first (it's simpler in that way).

Five Tables

Most important are the number squares table 2 and table 5. From the values found there, the position in the list of weekdays is derived in the last step and thus the corresponding day itself. The tables 1, 3, and 4 are exclusively used to find the correct line and column in the aforementioned two tables. In this process, table 1 provides the column for table 2; the line is derived from the table of days, which could be achieved just as well with a little calculation (see illustrations). Table 4 gives us the column for table 5, and in table 3 we find the line for table 5. Complicated? Now that I am writing it down, I also get that feeling, but actually, it's simple. Let's take the example of August 6, 1945.

In the table of days we check in which line there is the 66 -- in the sixth (where else?). Then we search the new table 1 for the column for the month August (8); that would be the fourth column. Now we can read the value in line 6, column 4 of table 2; there we find a 6. We keep that in mind. Now for the year. We look in table 3 for the last two digits, namely 45. It's right in the first line. In table 4 we look for the 19 in the Gregorian department and discover it in column 5. Checking table 5 in line 1, column 5, returns the value of 3. This 3 is then added to the value we had before (6), and we get a 9. Now we check the table of days once more with the 9, and see what weekday is printed on the same line -- Monday. August 6, 1945 is a Monday. Try it with your dates of birth! (Mine is a Wednesday.)

By the way, have you recognized the programming problem? No? Well, usually arrays are used by programmers to "store" data in them, but here it's just the other way around! There are already pieces of data in it, and we have to find out "where exactly" they are located! How do I get the information that 45 is located in the first line without much fuss? And that 19 is in the fifth column? Isn't this a nice little task I leave with you to find out for the next time? Have fun with it!

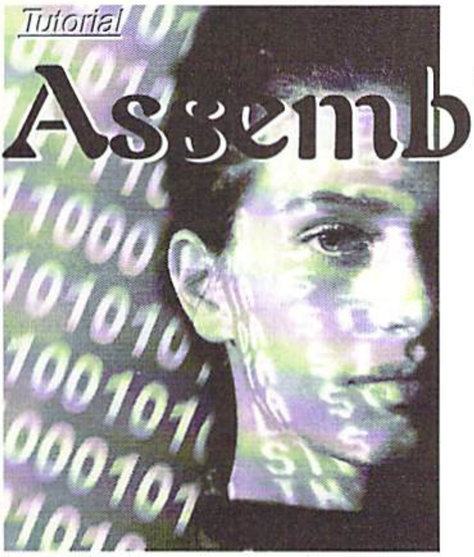
Yours, Arndt

Variables of the Menu System

BASIC line	Function of module	used variables
20 to 110	Main routine	(not influenceable) used internally: i4, i5, i6, i7, i8, i9, en$, st$, x9$, lg, ri, zl, sp, br, ho
1100	Drawing a rectangle	zl (line on screen, 0 to 21) sp (column, 0 to 36) br (rectangle's width, measured inside, 1 to 37) ho (rectangle's height, measured inside, 1 to 23) i5 (rectangle's type, 0 to 3) used internally: i9
1200	Positioning the cursor	zl and sp (valid values as above)
1300	Deleting a rectangle	zl, sp, and br, ho (measured inside, but the rectangle's frame will also be deleted) used internally: i9
1400	Getting chars	lg (acceptable length of input, 1 to 36) en$ (return value) used internally: i9, x9$
1500	Getting cursor buttons	ri (return value, 1=right, 2=left, 3=down, 4=up, 5=RETURN) used internally: x9$
1600	Drawing a box for a menu	(not influenceable) used internally: i6, i7, i8, i9, zl, sp, br, ho
1700	Status Line Issues	st$ (Text to be issued out) used internally: en$, lg, i9, zl, sp
1800	Drawing a line	lg (length of the line, up to 40) en$ (char the line consists of, 1 char) used internally: i9
2000	Menu distributor	i7 (menu item) i8 (submenu item)

The loop variables i4, i5, and i9 can be used freely; i6, i7, and i8 are set and required by the main routine. All other variables should only be changed for their provided purpose.

Assembly Language



Picture: Bernhard Pehall/Gaucher Bauer (www.lichtbogen.at)

In our last installment we discussed writing an interrupt service routine. The only problem was that the effect ran too fast and that we are not allowed to integrate any delay loops in the interrupt, because we could "overload" an interrupt. Overloading in this context, means that the interrupt routine takes so long, that the computer will try to start a new interrupt while the old one is still being serviced. This shows up as a rather displeasing flicker, distorted music or some other stuff ... and we don't want that to happen, of course.

by Wanja Gayk

Part 5

Thus, we have to think of another way to slow down effects that are just too fast. The simplest method is by running the program (or part of it) only every other interrupt. By doing this will will cut the speed exactly in half.

As before, we will need an interrupt initialisation routine that we can use for all following programs. Here is our code:

```
$1000 SEI ; disable interrupt
$1001 LDA #$00 ; redirect interrupt
vector to $1100
$1003 STA $0314
$1006 LDA #$11
$1008 STA $0315
$100B LDA #$FB
$100D STA $D012 ; interrupt starts at
raster line $FB
$1010 LDA #$1B
$1012 STA $D011 ; text screen "on" and at
the same time clearing bit 7 in $D011
$1015 LDA #$F1
$1017 STA $D01A ; activate raster line
interrupt
$101A CLI ; enable interrupt
$101B RTS
```

Now we will create a simple effect. Let's say we want to make the screen shake a little. To do so, we use the VIC horizontal scroll register at $d016.This register allows one to scroll the entire display (or a portion of it) up to eight pixels horizontally without using any significant processor power.

```
-- IRQ Main Routine --

$1100 JSR $1110 ; scroll routine
$1103 LDA #$01 ; confirm interrupt as
finished
$1105 STA $D019
$1108 JMP $EA31 ; and on with the
original ROM routine

- Hardware Scroll -

$1110 DEC $1200 ; decrease delay counter
$1113 BEQ $1115 ; if result = 0, branch
to $1115
$1114 RTS ; otherwise immediate return
from subroutine
$1115 LDA #$02 ; $02 in Accumulator
(start value for counter)
$1117 STA $1200 ; and write to delay
counter
$111A LDY $1201 ; load table counter
(index) to y- register
$111D LDA $1202,Y; and retrieve scroll
value from table + Y to accumulator
$1120 ORA #$08 ; set bit 3 (for 40-
columns mode) in accumulator
$1122 STA $D016 ; and write acuumulator
to scroll register
$1125 INY ; increase y- register by 1
$1126 CPY #$13 ; and compare it to $13
$1128 BNE $112C ; branch to $102c if not
equal
$112A LDY #$00 ; all through, reset y-
register to zero
$112C STY $1201 ; and write to table
counter
$112D RTS ; Return from subroutine

- table counter, delay counter and scroll
value table ($13 values) -

m $1200 02 00 01 02 03 04 05 06
m $1208 07 07 07 07 06 05 04 03
m $1210 02 01 00 00 00
```

Now, are you confused? I hope that this will be over soon. This whole assembly works like this: in interrupt init I choose the raster line in which the interrupt shall be triggered, in this case, raster line $FB, which is in the lower screen border. Every interrupt I call the scroll routine. At that time the electron beam is defintely still in the lower border and we remember: when it leaves the lower border, it automatically returns to the top - but our routine won't take that long. This way we make absolutely sure that the scroll value is already set before the screen is drawn, so that we don't have any "twisting" on the screen, where a new scroll value replaces the old. But allright, these are details that you will surely investigate more closely later. Back to the scroll routine.

It starts immediately with our "delay". The principle is quite simple: I take a counter that I keep decreasing by 1 and as long as it isn't zero, I leave the subroutine without doing anything. But once the counter reaches zero, I reset it to its initial value (the $02 in $1200 in the table) and execute my real routine. This starts by getting an index on the table and then the scroll value from table + index. The next instruction is new:

ORA #$xx - Perform a Logical OR operation on the Accumulator against the value #$xx, leaving the result in the Accumulator. With ORA, one can set single bits as desired (See illustration 1 - Logical Operations). Other logic instructions include AND #$xx (Logical AND operation) and EOR #$xx (Logical Exclusive-OR operation).

Using the ORA instruct ion, I set the bit which activates 40 column mode. Keep in mind that register $d016 has several functions (refer to illustration 2 - Registers $d011 and $d016).

Anyway, now we have the final value that we write to register $d016. The following program lines increase the table counter by 1 and, if it exceeds the number of table entries, reset it to zero. Then the subroutine ends, and we find ourselves in the main IRQ routine, which jumps to the normal interrupt routine found in the Kernal ROM.

You should take another long look at it all, so that you place everything correctly. If you want to experiment a bit, try to do the same thing with register $d011, so as to make the screen go up and down. Or maybe even both movements together? I don't want to limit your adventurous spirit - go at it!

In the next part, there will once again be something basic about adressing methods. Maybe that will seem somewhat boring at first sight, but everything will advance so much easier afterwards.

Because the English version of GO64! began with the August 1999 edition, parts 1 - 4 of the tutorial (published prior to August 1999) were only published in the German version. We will, however, offer the tutorial parts 1 - 4 in English on our homepage (www.go64.c64.org). If you don't have access to the internet, you can get the first four parts by mail for free. Simply send a letter to:

CSW-Verlag
Goethestr. 22
D-71364 Winnenden
Germany
or FAX us at: +49 7195 61120

The Logical Operands AND and OR

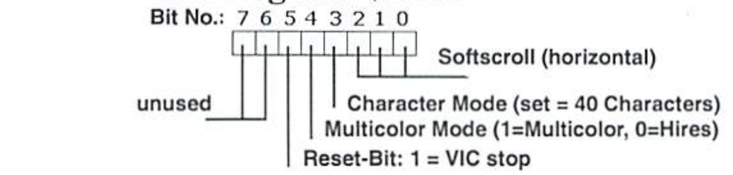
1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 0 = 0

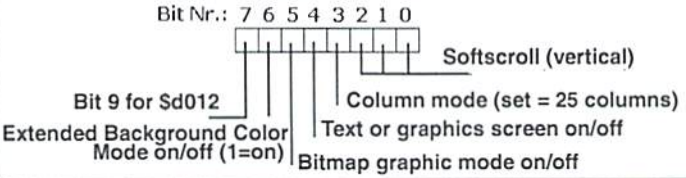
```
    01110000 ($70)
AND 00101011 ($53)
  = 00100000 ($20)
```

```
   01110000 ($70)
OR 00101011 ($53)
 = 01111011 ($7B)
```

The Register $D016



The Register $D011



Bit Nr.: 7 6 5 4 3 2 1 0
Softscroll (vertical)
Bit 9 for $d012
Column mode (set = 25 columns)
Extended Background Color Mode on/off (1=on)
Text or graphics screen on/off
Bitmap graphic mode on/off

```
.,2013 BD
.,2016 9D
.,2019 BD
A2 00    LDX #$00
BD 00 04 LDA $0400,X
9D 00 10 STA $1000,X
BD 00 05 LDA $0500,X
```

Anatomy of a Mouse Driver

The Commodore 1351 mouse and its compatible siblings, by Scanntronik or CMD for instance, are very comfortable input devices. However, the software driver can be seriously improved.

by Marco "Mac Bacon" Baye

For the C64 the most common input devices besides the keyboard are certainly joysticks. For most purposes, they are indeed well suited, but those users that have already used a proportional mouse will certainly agree with me that moving a mouse pointer over the screen is much more pleasant. No matter whether one moves the mouse fast, slow, short, or long, the pointer on the screen copies the movement exactly.

Well, at least it should. However, many mouse drivers are based on the program that can be found on the disk accompanying the 1351. Since this had been programmed quite poorly, the user is confronted with three unpleasantries. In the following text, I am going to show these weaknesses one by one, analyze the problems, and point out possible solutions for each. On the coverdisk there are some files related to this:

"inputdrv128.bin" - finished mouse driver for the C128
"inputdrv64.bin" - finished mouse driver for the C64
"inputdrvsrc.txt" - source code for both (!) drivers
"pointersprites" - two sprites for the mouse pointer

All these files are public domain and can be integrated into your own programs - even into commercial ones! My main interest in this is to make more programmers put mouse support into their programs.

The two finished drivers are meant for BASIC programmers; at the end of the text, integrating them into your own programs will be explained.

For assembly language programmers, the source code is certainly more interesting. Since some optimizations utilized there, like the reverse subtraction for example, may not be known to everyone, they will be explained in a separate article. following this one

Since both finished drivers already contain sprites, the fourth file is really not necessary, but it could prove useful for your own first tries with the source code; when using other (personal) sprites, you may need to change the source code depending on their shape.

But now back to the real issue - the three weak points of the standard driver (the new drivers on the coverdisk don't suffer from them, of course).

The pointer's speed

All those that have already used a mouse on a different system will certainly notice this weakness first. Even for very small pointer movements on the screen, you need to move the mouse relatively far. This is due to the fact that the driver unnecessarily halves the resolution of the mouse, which is not that high to start with. I have run some very simple tests; according to them, the 1351 has a resolution of about 180dpi (dots per inch). Due to the cutting in half mentioned before, that means that with every movement of the mouse by one inch, the mouse pointer on the screen just moves by 90 pixels.

Some may now argue that this ratio is enough and that the "ideal" resolution would depend on your point of view anyway. From the user's point of view, this argument seems reasonable, but from the programmer's point of view, it always makes sense to offer coordinates in the highest resolution possible. If the resolution is too high for a certain purpose, you can convert it with ease, for instance by cutting the value in half. But if the resolution is already too low, the situation will not improve by doubling the value; the value will become higher, but the number of *possible* values remains the same. This fact can be exemplified in photography; you may enlarge a photo by any factor, but it will never become sharper by doing so. Because the number of possible values remains the same, I'd always prefer the original instead of a scaled-down version.

Anyhow, for a human as pathologically lazy as I am, it is an unacceptable state of affairs that when using the standard driver of the 1351, a movement of the mouse pointer by a few meager pixels always degenerates into a training program for the biceps and triceps. Since you can't increase the resolution of the 1351, the problem can only be solved (or at least diminished) by suppressing the driver's cutting in half of the value. Without fail, this leads to the question, "Why does the driver halve anyway?"

The answer is related to the signal transmission to the C64/128. Because computer mice were totally unknown at the time of the C64's design, no corresponding connection had been provided in the breadbox. For that reason, Commodore simply misused the joystick ports' POT lines for the mouse; these lines were originally meant for the connecting of paddles. In a paddle there is a potentiometer (from which the lines' name "POT" originated), so that turning it changes the electric resistance. The resistance is measured by the sound chip's analogue/digital converter and can then be queried there (addresses $d419/$d41a in the I/O area) by the software.

In order to transmit its coordinates to the C64, the 1351 therefore creates two analogue signals. The SID then converts them into digital values. This method certainly had the advantage of making it possible to cheaply produce the 1351 then. However, it has the disadvantage that the digital values can be messed up by the analogue transmission quite easily.

And that happens indeed... Even while you are not moving the mouse at all, the bytes that can be read at the SID can jump back and forth between two consecutive values because of the tolerances of the analogue technology. This means that if the value to be read at the SID changes by 1, it does not necessarily mean that the mouse has really been moved.

The standard driver solves this issue by the aforementioned cutting in half; after calculating the difference between the new and old position, the result is halved. By doing so, "-1" as well as "+1" would become "0" - the messed-up values don't cause any problems anymore. Unfortunately, this solution causes a side effect; the mouse pointer moves only half as far as would be possible (the "photo" has been scaled down). Simply moving the pointer on the screen twice as far (enlarging the "photo" again) won't do any good either; doing that, you would only reach every second-pixel column (the "photo" becomes blurred).

There is also another possibility; you can indeed solve both problems simultaneously. My proposition is simple but very effective. After the difference to the last value has been calculated, negative values are increased by 1 while positive ones are decreased by that amount, zero remaining zero.

This solution means that the computer "loses" one pixel from every mouse movement. No problem - those that have not just read this text will not notice that they have to move the mouse about 0.15 millimeters (!) farther than would be the case with an ideal solution (which is impossible).

With this little change, the pointer already moves at twice the speed over the screen, and still every pixel may be reached (to stay with the example, since the photo has not been scaled down, it has not lost any of its sharpness). On to the next problem.

Jumps during Fast Movements

That the 1351 and compatibles are also referred to as "proportional mice" is due to the fact that the pointer's speed on the screen is proportional to that of the mouse on the mousepad, but that only holds true up to a certain upper limit. During very fast movements of the mouse, the pointer may very well stand still at times or may even jump back a short distance.

The reason for this is the relatively small range of values the mouse provides. In theory, the SID converters could provide a byte value, in effect 256 different values. But the 1351 creates less than 128 different values; mine only creates just 125 (due to the analogue nature of the signals, this may slightly vary from mouse to mouse and from SID to SID - more details on that later on). The smaller the range of values, the sooner the counter overflows; that means if the range is small, you mustn't be too fast in moving the mouse, or the mouse begins to stumble. Since we can't increase the number of values delivered by the mouse, we have to solve the problem in another fashion.

First the good news. The problem has already been solved. Due to our solution for the first problem discussed, the mouse pointer already moves twice as fast as before; the user only needs to move the mouse half as fast as before. This alone already avoids most of the pointer's jumps.

And now for the better news... There is even an ideal solution; one simply queries the mouse more often (the mouse pointer's speed is not changed). As simple as that suggestion sounds, it's as difficult in its implementation. Simply calling the routine twice in a row wouldn't do much good, of course; instead you need to spread the calls evenly over time. Since the mouse driver usually sits in the interrupt, this means that now you have to trigger twice as many interrupts and to really call the mouse driver several times and not just all the other IRQ routines as well (otherwise, the clock ticks faster and so on).

For an experienced programmer, this shouldn't be that difficult, but it is so complex that I have not put it into my example source code, because that code should be universally usable, and the interrupt handling differs quite greatly between the C64 and the C128.

Pixels beyond Reach

The first two weaknesses could be "evaded" in the past by simply moving the mouse painfully slowly. But when dealing with the third problem, even that won't help. Once in a while, when using the 1351 standard driver, you have the situation in which you can't go to a certain pixel column or line precisely; the pointer jumps over it, time and time again, no matter how often you try.

This error always occurs when the range of values provided by the mouse is exhausted, and a "wraparound" to the other end of the range is made. The standard driver does indeed trust that both ends fit exactly.

Here's a practical example. My 1351 provides values in the range from $44 to $c0 inclusive for both x- and y- direction, therefore only 125 different values. If the current x-position is $c0 and I move the mouse one pixel to the right, the range of values overflows and resets to $44. However, the standard driver expects the interval to include exactly 128 values and calculates:
(NewPosition - OldPosition) AND $7f.
In the example, that would be ($44-$c0) AND $7f =$4. Although I have only moved the mouse by one pixel, the driver thus finds the distance "four". The disparity of three dots is due to the disparity between the width of the value interval expected by the standard driver and the real one (128 - 125= 3).

Since the standard driver then cuts the computed difference in half (see first problem), it reduces the effects of this error; the pointer only moves by two instead of four pixels. But that is no great help either, since you wanted to move the pointer by but **one** pixel.

What causes this problem exactly? Why don't the value intervals of the 1351 and the driver match? The reason for that lies in the analogue signal transmission as mentioned previously. Judging from the standard driver's behaviour and the values from my 1351, it was originally planned that the 1351 provided values in the range from $40 to $bf (or $41 to $c0), thus 128 different values. Certainly, it is also trying to do so. But due to the analogue transmission, it is not certain that the SID, after its analogue/digital conversion, registers exactly the value the mouse "meant" it to. Of course, you may be lucky, and the SID and mouse may exactly fit each other. In that case, this problem does not occur at all (therefore, I am not one hundred percent sure that this problem really exists with every SID/mouse combination, but it should be very common).

The problem can only be solved if the driver exactly tunes in on the value interval provided by the SID. This means that the driver needs to save the maximum and minimum and to calibrate its difference calculations to that. It works this way; at first, the difference between the new and the old value is computed. Then you have to find out whether the difference is really valid or is corrupted by a wraparound. In order to do so, the absolute value of the difference is compared with half the width of the value interval. If the mouse movement is less than half the interval width, the difference is most likely correct. But if the movement is greater, a wraparound is more probable. In the latter case, the difference is corrected to the real value by subtracting (or adding, if the difference is negative) the full width of the interval. Actually quite simple, isn't it? By doing so, we have a self-calibrating mouse driver.

The New Drivers' Usage

The drivers on the coverdisk already contain the explained improvements. In addition, when compared to the standard driver, they have the following advantages:

- The pointer consists of **two** sprites; if you give them different colours, the pointer can be seen more clearly.
- The pointer movement is limited to the visible screen.
- You may query the coordinates in pixels as well as in chars.
- In addition to the mouse driver, there is also a joystick driver (port 2), which even has an

acceleration function.

This is how you integrate the C64 driver in your own BASIC programs:

```
IF a=0 THEN a=1:
LOAD"inputdrv64.bin",8,1: REM must be in
*one* BASIC line, preferably the very
first one
SYS 49152: REM puts the driver into the
interrupt
POKE 53287, inner colour: REM colour
value, for example, 1 for white
POKE 53288, outer colour: REM colour
value, for example, 0 for black
REM Query:
IF PEEK(164) THEN ... : REM queries left
mouse button or joystick button
IF PEEK(165) THEN ... : REM queries right
mouse button
cx=PEEK(155) : REM queries x- position,
measured in chars
cy=PEEK(156) : REM queries y- position,
measured in chars
px=PEEK(251)+256*PEEK(252) : REM queries
x- position, measured in pixels
py=PEEK(253) : REM queries y- position,
measured in pixels
```

Similarily for the C128 driver:

```
BOOT "inputdrv128.bin" : REM loads the
driver and puts it into the
interrupt
SPRITE 1,1, inner colour : REM activates
sprite 1 (colour value, for
instance, 2 for white)
SPRITE 2,1, outer colour : REM activates
sprite 2 (colour value, for
instance, 1 for black)
REM Query:
IF PEEK(250) THEN ... REM queries left
mouse button or joystick button
IF PEEK(255) THEN ... REM queries right
mouse button
cx=PEEK(179) : REM queries x- position,
measured in chars
cy=PEEK(180) : REM queries y- position,
measured in chars
px=PEEK(251)+256*PEEK(242) : REM queries
x- position measured in pixels
py=PEEK(253) : REM queries y- position
measured in pixels
```

Please note: Both versions of the driver make use of zero page addresses that are really supposed to be used with datasette usage. In addition, the C64 version also uses the cassette buffer as memory for the pointer sprites. Therefore, driver and datasette can't be used together.

Because I am not in possession of JiffyDOS, I have not been able to test the driver together with it. But according to my information about JD memory usage, the driver should be compatible with JD.

Possible Further Developments

The term, "mouse driver", certainly will make most people think of GEOS first. But because I have nothing to do with GEOS, I myself will probably not develop a GEOS input driver from my source code, but anyone who would like to do so is hereby encouraged.

As far as further development is concerned, there are hardly any limits, for instance, you could forge a keyboard input whenever the buttons are pressed, by writing a certain character to the keyboard buffer. You could use different characters for "button was pressed" and "button was released". In both cases if you also store the corresponding mouse positions in two other addresses, you could write programs with drag & drop support without any difficulties.

Another idea would be to set the maximums for x- and y- position to bigger values than (319,199) and then managing a virtual screen of which only a part is shown at any given time. If the main program checks the pointer position and adjusts the screen contents when the currently-shown- coordinates-window is left, you could scroll through documents simply by pushing the mouse pointer against the screen border. For such an project, some further changes of the driver would be required, for instance, when setting the sprite position. That would then depend on the screen window. That is going to be as far as we get for now.

Anyhow... Happy coding!

Optimum or what?

Computing the expression "A minus B" in assembly language is not hard. Interestingly, you obtain the same result by computing "NOT (B minus (A plus 1))" instead. It sounds crazy, but sometimes the latter way is faster... and even uses less memory.

by Marco "Mac Bacon" Baye

The original reason for writing this text was that I consider the source code for my self-calibrating mouse driver to be relatively hard to understand; the reverse subtraction just mentioned is a good example for that. Because I really would not like putting extensive English explanations in my source codes, and because such information can be read with greater ease from paper than from a screen, it turned into a second article. Therefore, this is not about optimization in general but "only" about a description of the difficult parts of my mouse driver code.

I have written the source code on an RiscPC; therefore, it is in ACME syntax. But since there is no C64 version of ACME (yet, hehe), I have converted the source at least to the PETSCII common to the C64. On the coverdisk, the file name is "inputdrvsrc.txt".

Admittedly, an ACME source code in PETSCII format is neither fish nor fowl, but converting it (to the format, Turbo Assembler, for instance) shouldn't be that difficult (for example, "!word" becomes ".wo", "!byte" becomes ".by"). Those, who would like to assemble the source again with ACME, first need to reconvert it to ASCII or simply just get the original files from:

http://home.pages.de/~mac_bacon/dd/driversrc.a

When assembled, the source code generates the C64 binary; the changes needed for the C128 are described in the comments. Of course, I could also have defined a label "type" and then have used the conditional assembly of ACME:

```
!if type=64 (C64 stuff) else (C128 stuff)
```

... but since you would need to adjust the syntax for other assemblers anyhow, I just left it; the source code is already darn long for a tiny 3-block program anyway.

But now for the explanations (and the oh-so-swell reverse subtraction):

At the code's beginning, under "; --- Configurable values", there are all the settings with which the driver's function can be changed within certain limits, for example, the start address, maximum for x- and y-coordinates, delay and maximal speed for the acceleration function of the joystick control, the number of sprites used, and the position of the "hot spot" in the pointer's point.

With "hot spot", I am referring to the really relevant pixel in the pointer - therefore, its point in an arrow-shaped sprite. If you were to exchange that sprite for a targetting cross, the hot spot wouldn't be up and to the left but closer to the center. Since the driver has to consider this change, the position of the hot spot can be reconfigured here and adjusted to the sprite used.

The memory addresses, in which the driver manages the pointer position and buttons, are also set here. When integrating the driver into your own programs, you could, of course, substitute the zero page memory addresses with your own byte and word variables directly in the code.

The driver depends on the words for x- and y-direction to be directly sequential in the memory. Due to that limitation, you can use the same routine for x- and y- computations by having the routine address the data with an index register. When calling the routine with a 0 in the x-register, the first word is used; when there is a 2 in the x- register, the second word is used instead.

Below ";--- System constant", some labels for the I/O access are defined - nothing special.

Below ";--- Label definitions", labels are defined as well. But here, there are not numbers on the right side of the equations but only mathematical expressions; this is because these labels are completely dependent on the configurable values and therefore need no changing. One of the allocations uses the exponential operator (in ACME it is the "^" - character), in order to compute a bit mask for the VIC from the configured sprite numbers. Sprites 0 and 5, for instance, would become the binary value %0010001. Those of you that use an assembler without exponentiation therefore have to "hardcode" the bit mask.

At ";--- Entry point", we finally get started with the code. The first routine is the one putting the driver into the interrupt. Most programmers certainly do that automatically, but it also makes sense the way it is done here. Then you may load the file on a C128 with the command BOOT, which is the same as a

```
LOAD"...",8,1
SYS <Loading address>
```

When integrating into the interrupt, the current IRQ vector is adopted (as it should always be) so that expansion routines already in effect are not turned off.

Fun with the CIAs

Confusion may be caused by the commands

```
;Init mouse buttons
   lda #$11
   sta CIA1_prb
```

They are necessary for querying the mouse buttons. It is not enough to set port A of the CIA 1 to "input"; you must also set the desired bits in the data register. As to the reasons for that, I don't have any idea as well...

Beneath ";--- Variables", there is then yet a variable integrated into the program code - "JoyWaittime". Since this value doesn't have to be read by the main routine, I haven't made its address configurable.

It really starts with ";--- Main code", consisting of eight different parts. Depending on which function the driver doesn't currently need, the corresponding parts may be omitted.

First thing is that the driver turns off the decimal mode with "CLD". It could be the case that the main routine interrupted was using that mode, in which case the ADC and SBC commands in the driver would return wrong results. Therefore, we simply put in CLD. Since the processor status register is overwritten with an old copy from the stack at the end of an interrupt, there is no need to worry about a reactivation of the decimal mode that may be necessary.

Do you remember my introduction? Now, below ";--- Part 1, handling mouse movements", we get down to business - the reverse subtraction.

The driver jumps in this part twice to the routine "PotDelta"; at first it calculates the mouse movement in x- direction, then the movement in y-direction. The differentiation between the two possibilities is done using the method with the index register mentioned before.

The first call's result (the movement in x-direction) is simply added to the former x-position. However, things are different for the second call; in the computer's coordinate system, the y- axis points down, but in the mouse's system, it mathematically correctly points up. Therefore, the second result provided by "PotDelta" is not added but rather subtracted from the former position. In assembly language, that would normally look like this:

```
    lda Position
    sec
    sbc Change
    sta Position
```

If you definitely know that the carry bit is always set, when this routine is reached, you can omit the command "SEC", of course. When computing 16-bit values, the routine becomes a bit longer.

But this isn't of interest here; my point is just the following. After calling the routine "PotDelta", the result is in a/y (the low byte in the accumulator and the high byte in the y register). For reasons of understandability, we ignore the high byte for the moment and limit our considerations to the low byte in the accumulator. The program would then have the form:

```
    jsr PotDelta; returns result in a
    sta temporary memory
    lda Position
    sec
    sbc temporary memory
    sta Position
```

Those optimizing their programs with self-modification would maybe prefer writing:

```
    jsr PotDelta; returns result in a
    sta mod+1
    lda Position
    sec
mod sbc #$ff; argument will be modified
    sta Position
```

Reverse Subtraction

That's better but not good by a long shot; it would be ideal if you could immediately continue to calculate with the result of "PotDelta" somehow, instead of first storing it and instead loading another number. In effect, we need an alternative to "SBC", because instead of subtracting a number from the accumulator, we want to subtract the accumulator from a number. The ARM processor, for example, has for such purposes the command RSB, which is the abbreviation for "reverse subtract"; it is also from that definition that I have deduced the name for my solution.

Fundamentally, the solution is based on at first using the SBC command the "wrong way around". Therefore the routine is:

```
    jsr PotDelta; returns result in A
    sec
    sbc Position
    sta Position
```

In this form, the routine doesn't make sense, of course, because now the calculation isn't correct anymore. This can be helped; when computing (b-a) instead of (a-b), only the result's sign is changed (because -(b-a)=-b+a=a-b). Thus, we only need to negate the result, and once more, everything is correct. In order to negate a value on the 6502 processor, you need to create the two complements, in effect inverting it and adding "1":

```
    jsr PotDelta; returns result in a
    sec
    sbc Position
;Negation:
    eor #$ff: invert
    clc ; add "1"
    adc #$01
;Store corrected result:
    sta Position
```

This still doesn't look better than the initial routine, but we continue. Since (--a=a) is valid, you could also negate the other way around, that is, first subtracting "1" and then inverting:

```
    jsr PotDelta; returns result in a
    sec
    sbc Position
;Negation:
    sec; subtract "1"
    sbc #$01
    eor #$ff; invert
;Store corrected result:
    sta Position
```

Oh, boy, just have a close look at this. Suddenly our code contains two subtractions directly following each other, and the second one always only subtracts "1"! Those exactly knowing the SBC command's behaviour will now understand the trick. For those who don't, here a little hint; why do we set the carry bit before the SBC command anyway? Well? What would happen if it were to be clear once? Of course, the result would be corrupted. But in what way?

It would be "1" less than it was really meant to be! Those not completely understanding that should consider the functioning of the carry bit in 16-bit subtractions. Those that don't understand anything about that yet should stick to Wanja's assembly language course; sooner or later, our Uncle Wanja will certainly deal with the carry bit subject.

Equipped with this knowledge we can once more greatly simplify the latest routine. By replacing "SEC" with "CLC", we can leave out the second subtraction completely:

```
    jsr PotDelta; returns result in a
    clc; provide for a result reduced by
"1"
    sbc Position
;Inversion, thus together with "clc"
negation
    eor #$ff; negate
;Store result:
   sta Position
```

Isn't that a wonderful sight? That is only *one* command more than the normal subtraction (two bytes). This reverse subtraction can be connected for 16-bit calculations without problems. In the mouse driver's source code, for instance, it looks this way:

```
    jsr PotDelta; returns result in a/y
    clc
    sbc PointerYnow
    eor #$ff
    sta PointerYnow
    tya
    sbc PointerYnow+1
    eor #$ff
    sta PointerYnow+1
```

The only disadvantage of this trick is that it doesn't work in the decimal mode (but you don't ever use that mode anyway). Besides that, the reverse subtraction can always be applied simply by using CLC instead of SEC and inverting every byte with EOR #$ff before storing. In fact, quite simply, you only need to know it. Thus, remember this the next time you want to compute (a-b) and you already have (b) in the accumulator.

Variables without Memory

Let's get to the next tip. In the source code, a variable by the name "JoyStepsize" is used which occupies one byte of the main memory. At various places in the driver, this value is loaded, saved, increased, or decreased. But the memory space for that variable is not reserved by "!byte" but rather like this:

```
    JoyStepsize=*+1
    lda #$00; (self-modifying)
```

Thus the variable is put into the argument of an LDA command. This is usually nothing new; self-modification is common on the C64 anyway. Yet, it is applied far too rarely, I think. A great number of the variables, whose memory space is reserved by "!byte", could just as well be put into the argument of a command working on that variable. Of course, this is not possible with STA, it lacking immediate addressing. INC, DEC, LSR, ASL, ROL, and ROR won't do for the same reason. However, LDA, ADC, SBC, AND, ORA, EOR, LDX, LDY, CMP, CPX, and CPY work without problems, and these are used often enough.

What purpose is this supposed to serve? Well, to be honest, the storage place makes no difference for the variable itself. But if replacing one absolute addressing by one immediate one, you save an entire byte. If you consistently apply this change to all byte-variables, you can save lots of memory within big programs.

And we continue with ";--- Part 6, restrict coordinate range". In this very short part, the initially mentioned simple method of signaling a subroutine with the help of an index register to which data is to be accessed is applied. Okay, this not that fundamental.

In ";--- Part 8, making char coordinates", things become interesting once more. This part converts the pixel coordinates into char coordinates, in effect simply dividing by eight. For the y- coordinate, this is in effect LSR:LSR:LSR, but the x- coordinate is more of a problem; the pixel value in x- direction can also be greater than 255. Thus, you also have to consider the high byte. There are now several possibilities:

```
lda PixelLow
lsr; shift down
lsr
lsr
sta CharLow
lda PixelHigh
asl; shift up
asl
asl
asl
asl
ora CharLow
sta CharLow
```

Ugly - long and slow. Only being able to reach 319, the x- value is a great help to us, because then the high byte can become 0 or 1. Thus we can write:

```
lda PixelLow
lsr; shift down
lsr
lsr
ldx PixelHigh
beq *+4
ora #$20
sta CharLow
```

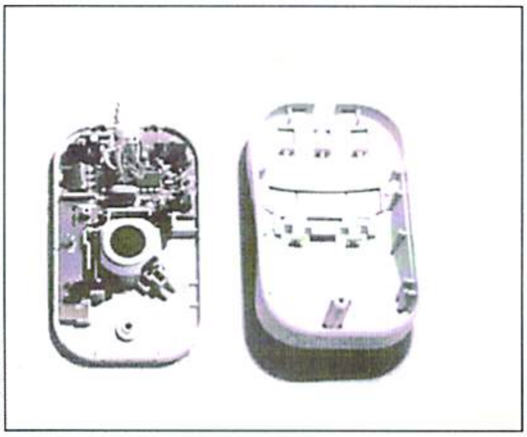
If the high byte is not zero, we insert the missing bit into the result by hand using ORA. Much better. But what about rewriting the driver for the VDC, for instance? There the x- value can become up to 639, so that the high byte could take on the values 0, 1, or 2. The source code contains my suggestion:

```
lda PixelLow
lsr
lsr
lsr
ldx PixelHigh
ora OrTable,x
sta CharLow
...
OrTable !byte $00,$20,$40
```

The deviation in the requirements is minimal, but the suggestion's approach is completely different. This only the wisdom of the day; in effect, hardly anyone will benefit from this example.

Now it's only the part ";--- 'Pot' subroutine".... Then you've done it. This part contains the really important items of the driver, namely the improvements as compared to the standard driver.

At first the variables... The driver stores each of the greatest and smallest values provided by the mouse (the reasons for this can be found in the other article). You certainly know the problem when you wish to check the state "smaller than or equal" with the "CMP" command. Unfortunately, it is not possible; the zero flag allows us the differentiation between "equal" and "not equal", the carry bit allowing us the differentiation between "smaller than" and "greater than or equal". The n- flag does the same thing as the c- flag, only related to numbers with sign. In order to query the state "smaller than or equal", behind the CMP, you therefore always have to use both a BEQ and a BCC for the same target.


Picture: Sebastian Bock

At First Glance no Advantage - But at a Second Look an Advantage indeed

But since this comparison is absolutely necessary and occurs at every interrupt, I don't store the biggest value but the biggest value+1 in the variable "PotMax". Of course, I buy into the advantage of the now unnecessary BEQ with the necessity of increasing "PotMax" by 1 before storing. Since the value is read more often than written, this is still an advantage - no saving in memory space but in time.

The same purpose is fulfilled by "HalfPotWidth" - this variable always containing the contents of the variable "PotWidth" but shifted by one bit to the right and thus halved. Both the complete and the halved value are needed fairly often, but they are only seldom written. Therefore, I take care of the cutting in half right after resetting "PotWidth" and thus have the halved value effectively "in the cache".

And finally, a small example for the usefulness of personal calculations in advance. At the label "minus", the driver needs to compare the negated contents of the accumulator with "HalfPotWidth". Just like in the case of the reverse subtraction, we immediately continue to compute with the accumulator:

```
;Negate value:
   eor #$ff
   clc
   adc #$01
;Comparison:
   cmp HalfPotWidth
   bcc SmallerOrEqual
   beq SmallerOrEqual
```

Here we again have the problem to branch with BEQ **and** BCC. However, this can also be done another way:

```
;Invert value only instead of negating:
   eor #$ff
;Comparison:
   cmp HalfPotWidth
   bcc SmallerOrEqual
```

Because the inverted value being always smaller by "1" than the negated value, the conditions for the comparison change, and the BEQ is no longer needed. In addition, this method also has the advantage that you can immediately restore the original value in the accumulator by "EOR #$ff" - something this driver does indeed do here (even before branching).

So, that's it. I am at the end, both of my wisdom and of my source code. And bodily, mentally, and intellectually, I am always finished. The text turned out to be longer than I expected; I hope it will be advantageous for someone and does not waste precious space in GO64! magazine...

Happy coding!

CMD WAREHOUSE BARGAINS
1-800-638-3263

CMD's Warehouse Bargains include many hard-to-find used and (where noted) new items. In most cases we have only one or two of each item in stock, so be sure to place your order before your selection has been taken! A minimum order of $20.00 applies to all Warehouse Sale orders. Shipping charges are not included in the prices shown. (*See our main advertisement elsewhere in this issue for shipping information*).

TELEPHONE ORDERS ONLY ON THESE ITEMS

Mailed in orders for non-reserved out-of-stock items will result in a merchandise credit. No cash refunds. All items listed here are sold "as-is", and all sales are final. Prices and specifications subject to change without notice. Not responsible for typographical errors.

BOOKS

32 BASIC Programs for the C64	$8.00
Anatomy Of The 1541	$8.00
Basic Computer Games	$6.00
C128 Computer Aided Design (new)	$8.00
C64 Programmer's Reference Guide	$25.00
Commodore 64 Favorite Programs Explained	$10.00
Commodore 64 For Kids from 8 to 80	$8.00
Commodore 64 Games Book	$5.00
Commodore 64 Programs for the Home	$8.00
Compute's First Book of C=64 Sound &Graphics	$10.00
Computes Second Book of Commodore 64	$10.00
Computes Second Book of Machine Language	$10.00
"Guide to Care, Feeding and Training of C-64"	$6.00
How To Use The Commodore 64 Computer	$10.00
Inside Commodore DOS	$10.00
Osborne 1 User's Reference Guide	$15.00
Superbase The Book	$10.00
The Elementary Commodore 64	$10.00
The Official Book Of King's Quest-Daventry	$10.00
Using & Programming the C64	$8.00
Using Newsroom	$10.00
Your Commodore 64-A Guide to the C 64	$10.00

BUSINESS

64 Powerpack-NEW	$10.00
Better Working File & Report (new)	$15.00
Business System- Sales Analysis Management	$10.00
Datamanager 128	$15.00
Datamanager 128	$20.00
Datamanager 2 (new)	$15.00
Easy Finance 1	$5.00
EPYX Microsoft Multiplan Spreadsheet	$10.00
Financial Cookbook-NEW	$8.00
General Ledger for the C=64-NEW	$15.00
Paperback Filer 64	$10.00
Paperback Planner 64	$10.00
Payables for the C=64-NEW	$15.00
Planner's Choice -Personal Planning System	$8.00
Power Assembler-(new)	$10.00
Practicalc 64	$5.00
Practicalc-(new)	$10.00
Practifile 64-(new)	$10.00
Receivables for the C=64-NEW	$15.00
RUN Productivity Pak II	$8.00
Silent Butler (Check Writing system)	$15.00
Superbase Starter-NEW	$10.00
Swiftcalc 128	$20.00
Swiftcalc 64	$10.00
The Consultant (Database Management)	$15.00
The Home Accountant	$15.00
The Home Banker	$10.00
The Home Manager	$10.00
The Manager	$10.00
The Word Machine/Name Machine (new)	$5.00
Valuecalc	$8.00

CARTRIDGES

Alf in The Color Caves	$10.00
Alpha Build	$10.00
Big Bird's Special Delivery	$10.00
Bubble Burst	$10.00
Dragonsden	$8.00
Facemaker-(new)	$10.00
Heskit	$10.00
Kids on Keys-(new)	$10.00
Kindercomp	$10.00
Lemans	$8.00
Linking Logic	$10.00
Mach 5 Cartridge	$15.00
Mastertype (new)	$10.00
MS Pac-Man	$8.00
Pinball Spectacular	$8.00
Radar Rat Race	$8.00
Sea Speller	$10.00
Speed Bingo/Math	$10.00
The Final Cartridge	$15.00
Up & Add 'em	$10.00

CARTRIDGES (No Docs)

Blueprint	$8.00
Clowns	$8.00
Ducks Ahoy	$8.00
Kindercomp	$8.00
Lemans	$8.00
Linking Logic	$8.00
Magic Desk I	$8.00
Maze Master	$8.00
Memory Manor	$8.00
Popeye	$8.00
Speed/Bingo Math	$8.00
Up & Add 'em	$8.00
Vic 20 -Hesware-Gridrunner	$5.00
Vic 20- Avenger	$5.00
Vic 20-Jupiter Lander	$5.00
Vic 20-Omega Race	$5.00

DESIGN

Awardware	$8.00
Delta Drawing-(new)	$6.00
Jingle Disk-NEW	$6.00
Printer's Devil for Print Shop	$8.00
The Christmas Kit-NEW	$10.00
The Newsroom	$10.00
The Newsroom (new)	$15.00
The Print Shop	$15.00
The Print Shop Companion	$10.00

DISK UTILITY

Commodore 64 Software Bonus Pak	$5.00
Disk Utilities	$5.00

EDUCATIONAL/CHILDREN

Computer Science	$5.00
Early Learning Friends	$10.00
Easy Lesson and Easy Quiz	$5.00
English I	$5.00
ENGLISH II	$5.00
First Men on the Moon -Math-(new)	$6.00
Grade A Math	$5.00
In Search of the Most Amazing Thing-(new)	$6.00
Jungle Book Reading-(new)	$6.00
Koalagrams Spelling 1-use with Koala Pad-(new)	$6.00
Logic Builders	$8.00
Math Busters	$8.00
MECC Expeditions	$8.00
MECC Odell Lake	$8.00
Paint-A-Rhyme -for use with Koala Pad-(new)	$6.00
Peter Rabbit Reading-(new)	$6.00
Snooper Troops Case #1 (new)	$6.00
Snooper Troops Case #2	$5.00
Snooper Troops Case #2 (new)	$6.00
Snoopy's Skywriter Scrambler	$10.00
Story Machine	$8.00
SUNBURST-Memroy Building Blocks	$10.00
Tales of Mystery	$8.00
The Argos Expedition	$8.00
Up for Grabs-(new)	$6.00
Weather Tamers	$8.00
Where in Europe is Carmen San Diego (new)	$20.00
Where in the USA is Carmen Sandiego	$20.00
Where in the World is Carmen San Diego (new)	$25.00
Wizard of Words	$10.00

GAMES

20 Solid Gold C64/128 Entertainment Hits	$10.00
20 Solid Gold Hits -5 Disks	$10.00
Aliens	$4.00
Amnesia	$6.00
Archon	$10.00
Ardok-The Barbarian	$6.00
Batman	$10.00
Batman-The Caped Crusader	$10.00
Beach-Head	$6.00
Below The Root (NEW)	$10.00
Beyond Zork-NEW	$6.00
Blood Money-NEW	$6.00
Blue Angels-Formation Flight Simulation	$10.00
Bop'N Wrestle-NEW	$6.00
Bureaucracy	$10.00
CavemanUgh-lympics	$4.00
Championship Baseball	$6.00
Championship Golf (new)	$6.00
Championship Wrestling	$6.00
Chomp (new)	$4.00
Classic Concentration	$6.00
Commodore 64C Test Pilot-NEW	$6.00
Conflict in Vietnam-NEW	$6.00
Countdown to Shutdown	$4.00
Crossfire	$4.00
Cyrus Chess	$6.00
Deadline	$4.00
Demon's Winter	$6.00
Dig Dug	$4.00
F15 Strike Eagle	$10.00
Feud	$6.00
Fist - The Legend Continues	$6.00
Flight Simulator II	$15.00
GBA Championship Basketball-2 on 2-NEW	$6.00
GFL Championship Football	$6.00
Gold Record Race-(new)	$6.00
Grand Prix Circuit-(new)	$10.00
Guerilla (new)	$6.00
Hacker	$4.00
Hardball (new)	$6.00
Heartland (new)	$6.00
High Roller-NEW	$6.00
Howard the Duck-Adventure on Volcano Island	$4.00
Invaders of the Lost Tomb-(new)	$6.00
Jack Nicklaus Greatest 18 Holes of Golf-NEW	$6.00
Jaws	$6.00
Jeopardy-Junior Edition	$6.00
John Elway's Quarterback	$6.00
Jordan vs Bird	$10.00
Karate Chop (new)	$6.00
Kings of the Beach (new)	$6.00
Knight ORC	$6.00
Kung Fu II-Sticks of Death (new)	$6.00
Last Ninja 2-NEW	$6.00
Legionnaire	$5.00
Lords of Conquest-(new)	$6.00
Lords of Conquest-NEW	$6.00
Mean Streets	$10.00
Megaplay Volume 1	$4.00
Minit Man	$6.00
Murder by the Dozen	$8.00
Navy Seal	$6.00
Navy Seal (new)	$8.00
Operation Whirlwind	$6.00
Pathwords	$6.00
Power Drift	$4.00
Predator	$6.00
Presidential Campaign	$6.00
Project Space Station	$4.00
Q-Bopper	$6.00
Qix (new)	$6.00
Questron II	$4.00
Raid Over Moscow	$4.00
Raid Over Moscow	$6.00
Rambo III	$4.00
Rick Dangerous	$4.00
Robbers of the Lost Tomb	$4.00
Sanxion	$6.00
Sargon II-(new)	$6.00
Sargon III-(new)	$8.00
Serve and Volley (new)	$6.00
Shinobi	$6.00
Silent Service	$4.00
Sky Fox	$6.00
Snow strike-NEW	$6.00
Spiderman	$4.00
Sporting News Baseball	$6.00
Star Trooper-(new)	$6.00
Stationfall	$6.00
Summer GamesII	$4.00
Superman	$4.00
Superstar Ice Hockey	$6.00
Suspended -A Cryogenic Nightmare	$5.00
Sword of Fargoal	$4.00
Tag Team Wrestling	$4.00
TETRIS	$6.00
The Last Ninja	$4.00
The Lords of Midnight	$6.00
The President is Missing (new)	$5.00
The World's Greatest Baseball Game	$4.00
Thud Ridge-NEW	$6.00
Times of Lore	$4.00
Titan-NEW	$6.00
Total Eclipse (new)	$5.00
Ultimate Wizard	$6.00
War in Middle Earth-NEW	$6.00
Wheel of Fortune	$8.00
Wild Streets-NEW	$6.00
Word Flyer	$6.00
Z-Pilot	$4.00
Zenji	$6.00
Zork I	$6.00
Zork II	$10.00
Zork III-NEW	$8.00

GEOS

Deskpack 1-NEW	$10.00
Fontpack 1-NEW	$10.00
GEOBASIC (new)	$10.00

HARDWARE

1351 Mouse w/Manual-no disk	$20.00
1702 Monitor -Color 40 Column-refurbished	$99.00
1802 Monitor-Color Monitor 40 Column-refurb	$139.00
Aprotek 1200 Baud Mini-Modem (new)	$30.00
Assorted Joysticks	$5.00
C128D Keyboards (NEW)	$69.95
C128D Keyboards-REFURBISHED	$59.95
C64 Keyboard (NEW)	$49.95
C64 Keyboard (refurbished)	$39.95
CPU64 Version 1 Base Model	$139.00
Datasette	$5.00
EPYX Fast Load Cartridge w/Manual	$10.00
Homewriter 10 Printer w/manual	$45.00
Koala Pad Only no docs	$20.00
KXP 1080I Printer	$75.00
Laser 190E Printer	$30.00
Magic Mouse	$10.00
Magic Voice Speech Module	$50.00
Multiplexer	$50.00
Okimate 10 Color Printer w/manual,disk,ribbons	$45.00
STAR Gemini 10X w/Manual	$25.00
Super Expander 64	$8.00
The Voice Messenger -Speech 64	$20.00
Used SlimLine cases for Commodore 64	$10.00
VICMODEM for VIC20	$5.00

MISCELLANEOUS

10 Assorted Commodore Magazines 1985-86	$10.00
64 Doctor	$5.00
Home Video Producer	$10.00
Instant Music-NEW	$6.00
LOADSTAR Disks#46,65,66,67,68	$10.00
Synthesound 64	$5.00
Vic 20 Cassette-Intro to Basic Part 2	$10.00

MISCELLANEOUS SOFTWARE

Aerobics	$6.00
Bobsterm Pro (new)	$10.00
Commodore	$5.00
Commodore Technology	$5.00
Music Construction Kit	$5.00
SAT Score Improvement System	$5.00
Writer/File Pak 1541	$8.00

PROGRAMMING

Better Working Power Assembler	$10.00
GEOBASIC (new)	$10.00
Hesware - Graphics Basic	$10.00
Introduction to BASIC/Part I	$8.00
Oxford Basic	$10.00
Oxford Pascal	$10.00
Simon's Basic	$10.00
Start Programming w/Gortek and the Microchips	$5.00

WORD PROCESSING

Easy Script	$5.00
Mastertypes Writer	$8.00
Mirage Concepts-Word Processor -Professional	$10.00
Outrageous Pages-NEW	$20.00
Super-Text Professional	$5.00
Supertext Word Processor	$10.00
The Printed Word	$8.00
The Write Stuff 128-NEW	$20.00
Word Writer 128	$20.00
Word Writer 6-NEW	$15.00
Wordpro 3 Plus/64-NEW	$15.00
Wordpro w/turbo Load & Save-NEW	$15.00
Writers Choice	$8.00

Shatter the 1 MHz Barrier

SuperCPU

20 MHz Accelerators for the Commodore 64 & 128

- Breathes New Life into your Commodore 64 or 128
- Increases System Performance by up to 22 Times
- Makes Sluggish Old Programs Quick and Useful

Take Your Commodore Into The Next Millenium!

Programs will suddenly spring to life, screens and menus pop up instantly, scrolling becomes fluid, and graphics fly. A Commodore equipped with a SuperCPU is like having a Volkswagon with a V-8. There's one small difference: the SuperCPU rides much smoother. And while the SuperCPU provides your computer with a radical increase in horsepower (up to 22 times faster), its operating system makes that power easy to use by maintaining compatibility with most existing programs and by making itself virtually transparent to the user and the computer.

Looking to the future, the powerful 65816 microprocessor gives programmers vast new resources to create powerful and sophisticated software. With its 16-bit registers, up to 16 Megabytes of system memory and an enhanced instruction set based on the 6502 core, the SuperCPU makes it easy to enhance existing software while providing the muscle needed to create a whole new generation of Commodore programs integrating more extensive graphics, sound and animation.

To get the most from your existing programs and to get ready for the future with just one puchase, get a SuperCPU. You'll wonder why you waited this long!

SuperCPU 64	$179.00*
SuperCPU 128	$239.00*

*prices do not include shipping

Features:

- High-speed microprocessor, 128K or 256K of fast RAM, advanced hardware logic, and up to 512K ROM enable you to run all types of existing commercial software at speeds up to 22 times faster than stock machines.
- BASIC programs, including most Bulletin Board Systems, seem to run as fast as machine language, and compiled BASIC programs run faster than ever!
- Commercial machine language programs (databases, spreadsheets, word processors, telecommunications, graphics, etc.) run so fast that you'll think you're running them on a completely different computer!
- GEOS comes alive—No typing or scrolling delays, graphic operations happen instantly, plus faster printing and disk access! Includes an enhanced GEOS setup utility!
- Easy-to-use: Plug it into your cartridge port** and start working—no special adjustments needed.
- Built-in JiffyDOS Kernal boosts disk access speed up to 15 times faster with JiffyDOS-equipped drives.
- Pass-thru port provides compatibility with RAMLink, SwiftLink, Turbo-23 2, REUs and other cartridges.
- Unit Enable/Disable, JiffyDOS On/Off, Normal/Turbo, and Reset switches make using the SuperCPU a snap!

**Additional MMU SuperAdapter installation required for 128 version.

SuperRAM Card (0 MB) $79 · 4 MB $114 · 8 MB $139 · 16 MB $179
SuperCPU 64 w/SuperRAM (0 MB) $219 · 4 MB $254 · 8 MB $278 · 16 MB $318
SuperCPU 128 w/SuperRAM (0 MB) $279 · 4 MB $314 · 8 MB $338 · 16 MB $378
Prices do not include shipping and are subject to change without notice. Not responsible for typographical errors.