Copyright 1986 MicroPACE. All rights reserved.

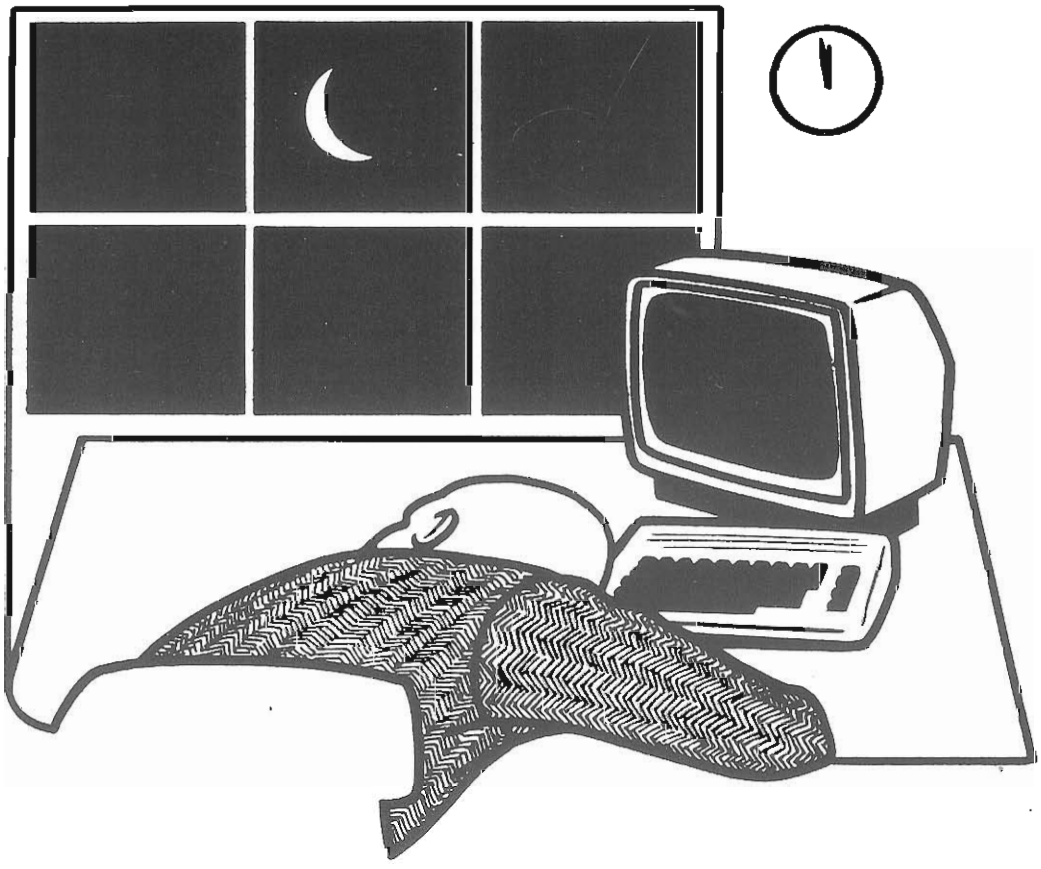
$2.00 us

ISSUE 35

1986

Midnite Software Gazette

The First Independent U.S. Magazine for users of Commodore brand computers.



IN THIS ISSUE:

I N T R O D U C I N G

SUPER KIT/1541

BY MARTY FRANZ & JOE PETER

SINGLE/DUAL NORMAL COPIER
Copies a disk with no errors in 32.68 seconds. dual version has graphics & music.

SINGLE/DUAL NIBBLE COPIER
Nibble Copies a disk in 34.92 seconds. dual version has graphics & music.

SINGLE/DUAL FILE COPIER
6 times normal DOS speed. Includes multi-copy, multi-scratch, view/edit BAM, & NEW SUPER DOS MODE.

TRACK & SECTOR EDITOR
Full editing of t&s in hex, dec, ascii, bin. Includes monitor/disassembler with printout commands.

GCR EDITOR
Yes disk fans, a full blown sector by sector or track by track GCR Editor. Includes Bit Density Scan.

SUPER DOS I
Fast boot for SUPER DOS. 150 blks in 10.12 seconds.

SUPER DOS II
Screen on and still loads 150 in 14.87 seconds.

SUPER NIBBLER
Quite frankly, if it can be copied on a 1541 this will do it! Including Abacus, Timeworks, Accolayde, Epyx, Acti-vision, Electronic Arts.

$29.95

PLUS $3.00 SHIPPING/HANDLING CHARGE — $5.00 C.O.D. CHARGE

PRISM SOFTWARE

401 LAKE AIR DR., SUITE D • WACO, TEXAS 76710
ORDERS (817) 757-4031 • TECH (817) 751-0200
MASTERCARD & VISA ACCEPTED

(C)opyright 1986 Micro-PACE, *
All Rights Reserved

Published by: Micro-PACE Computers,
Robert Wolters
Editor-In-Chief: Jim Oldfield Jr.
Editor: Tim Sickbert
Assc. Editors: Art Lewis Kimball
Mike Stout
Robert Baker
Dr. Immers
Elizabeth Kasper

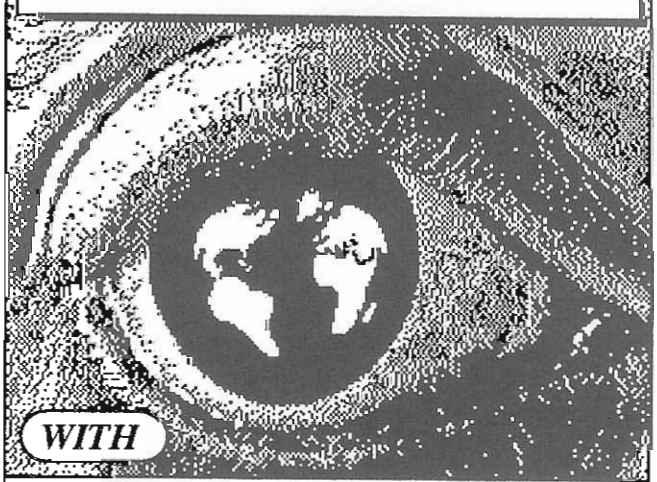
Address for all correspondence:
PO Box 1747
Champaign, IL 61820

Telephone: (217) 356-1885
BBS: (217) 356-8056
Punter 300/1200

Issue 35 July 1986
All contents Copyright 1986 Micro-PACE Inc.

Commodore, PET, VIC-20, Commodore 64, Amiga, Commodore 128 PC, PC-10, PC-20, B-128, are all copyrights and/or trademarks of Commodore Electronics, LT Commodore Business Machines.
Circulation this issue: 3000

SEE THE WORLD



WITH

WORLD GEOGRAPHY

Introducing an amazing new educational game for the Commodore 64/128 featuring a FULL COLOR 3D ROTATING GLOBE and detailed graphics of over 175 countries for 1 or 2 players.

only **$24.95**

Add $2 shipping and handling. CA residents add 6.5% sales tax.

BOBCO

200 7th Ave., suite 111 Santa Cruz, CA 95062
1-800-331-4321 **(CA) 1-800-851-1986**

★COMMODORE USERS★

Join the largest, active Commodore users group.

Benefit from:

— Access to hundreds of public domain programs on tape and disk for your **Commodore 64, VIC 20** and **PET/CBM.**

— Informative monthly club magazine

Send $1.00 for Information Package. (Free with membership).

TPUG yearly memberships

Regular member (attends meetings)	-$30.00 Cdn.
Student member (full-time, attends meetings)	-$20.00 Cdn
Associate (Canada)	-$20.00 Cdn.
Associate (U.S.A.)	-$20.00 U.S.
	-$25.00 Cdn.
Associate (Overseas – sea mail)	-$30.00 U.S
Associate (Overseas – air mail)	-$40.00 U.S.

TPUG Inc.
DEPARTMENT "N"
1912A Avenue Road, Suite 1
Toronto, Ontario, Canada M5M 4A1

* LET US KNOW WHICH MACHINE YOU USE *

C128 QUIRKS

by David Blezard

In issue #31, I told of the possible effects of doing a list in INSERT (<ESC> A) mode on the C128. I have since discovered that this can be used to merge program lines together.

Two lines can be joined by listing the second line and deleting the line number on the screen. Then, place a colon (:) at the start of the line. Now, move up to the line that the LIST command was typed on. Replace the line number of the line listed with the number of the first program line. Now go into insert mode with an '<ESC> A' and then <RETURN>. Finally, delete the second of the two lines that were merged.

Here is an example to help clarify things. First, enter the following two lines:

```
10 PRINT "HELLO"
20 GOTO 10
```

Now, clear the screen and 'LIST 20'. Then, move the cursor to where line 20 is on the screen, and use the <DELETE> key to erase the '20' at the beginning of the line. Insert a colon so the line looks like

```
:GOTO 10
```

Hit the <HOME> key to move the cursor to where the LIST command was typed. Change the 20 to a 10. Activate the automatic INSERT mode with an '<ESC> A' and hit the <RETURN> key. Lastly, delete line 20. If you list the program, it should now look like:

```
10 PRINT "HELLO": GOTO 10
```

This technique can be very useful for program crunching, especially since the C128 can have lines up to 160 characters long. Multiple lines can be done at once by repeating the process. Take note that some lines such as those containing IF...THEN statements or other branching statements cannot always be merged without causing problems.

C128 PROGRAM MERGE

found by

Jim Butterfield

Mr. Butterfield put a message on **Quantum Link's** C128 message base telling how to merge two program files, and was kind enough to demonstrate it when he came to Champaign to speak to the local users group.

Merging two programs is a relatively easy process on the C128, and it does not require any special utilities or hardware. All you need to do to merge 'PROGRAM 1' and 'PROGRAM 2' is:

```
DLOAD"PROGRAM 1"
DOPEN#8,"P1",W
CMD8:LIST
DCLOSE#8
NEW
:
DLOAD"PROGRAM 2"
DOPEN#8,"P1"
SYS(DEC("FFC6")),0,8,0
wait until things calm down
SYS(DEC("FFCC"))
DCLOSE#8
```

This will give you a fully merged, ready to run program. It is easy, relatively quick, and you cannot beat the price. Many thanks to Mr. Butterfield for finding this feature, and for all the work he has done over the years to make things easier for all Commodore users.

If you have any tricks, tips, or undocumented features of any of Commodore's PET line of computers, please send them in!

DATA BUILDER

By Robert W. Baker

Here's a handy utility program for all Commodore systems. Although written for the PET and CBM systems, it will run on the VIC-20, Commodore-64 and Commodore-128 as well. The program was designed to read a machine language program from disk and create a BASIC program on disk with the same machine language program converted to DATA statements. It also adds a FOR-NEXT loop with the correct parameters to read

the data and poke it into memory. Now you have a very easy way to get machine language programs into a form that can be merged with a BASIC program.

When this program runs, it first asks for the filename of the machine language program to be read and converted (lines 210-260). The machine language program must be a standard, loadable program file. It cannot be any kind of intermediate file created by an assembler. If you use an assembler to create the machine language program, you can load it following normal procedures then use an available monitor to save it onto disk as a program file. Once it's saved as a program file this program can be used.

After getting the machine language program's filename, this program prompts for the name of the new BASIC program to be created (lines 260-310). There cannot be any file on disk already using this name. If either file cannot be opened, an error message is displayed and the program terminates.

Note that the OPEN commands in lines 250 and 300 open a program (P) file for read and write respectively. This cannot be done with the BASIC 4.0 or 7.0 DOPEN command. The standard OPEN command must be used as shown. You probably won't find this in the Commodore documentation, but you can open and use program files just like sequential data files. Just keep in mind that the first two bytes of a program file specify the load address, indicating where the program will be loaded in memory.

Once the appropriate files are opened, this utility first reads and displays the load address of the machine language program (lines 330-370). When the program file is opened for reading, the first two bytes read are the load address in 6502 format (lo-byte/hi-byte). Thus, the address is converted to it's decimal value by adding the first byte (the lo-byte) to 256 times the second byte (the hi-byte).

A load address of 1025 is then placed at the start of the BASIC program being created with the PRINT#2 in line 390. The 1025 value is the standard load address for all BASIC programs on the PET and CBM systems. If you have a VIC-20, Commodore-64 or Commodore-128 system, there's no need to change this value since the BASIC loader will properly relocate the program when it's loaded on these systems. Thus, this value works for all Commodore systems.

After knowing the starting address, the utility program enters the main loop that reads a byte from the machine language program (line 410), gets the decimal value of the byte (line 420), and adds the data to the current BASIC program line being constructed in L$ (lines 430-440). The byte count for the length of the machine language program (NB) is also incremented.

The length of the BASIC program line created is checked in line 450 to see if more data can be added. If there's still room for more data on the same line, the program returns to line 410 to read the next byte from the machine language program. Otherwise, the subroutine at line 580 is called to add this line to the BASIC program being written on disk. After writing the line to disk, the program returns to line 400 instead of line 410 to place the DATA token (131) at the start of the BASIC line.

The subroutine in lines 580-610 adds the length of the line in L$, plus the five byte overhead for every BASIC line, to a pointer in LK to compute the link or starting address of the next BASIC program line. The two byte link value is written to the BASIC program file followed by the two byte BASIC line number from LN. The actual line from L$ then follows, along with a zero byte end flag to indicate the end of the BASIC program line. Another subroutine in lines 620-630 is used to convert the link address and BASIC line number into two byte 6502 address format and write them to the BASIC program file.

When the utility program reads the last byte of the machine language program file and detects the end of file, any remaining data will be output to the BASIC program file (lines 470-480). The status from ST is saved in SS after every read to the machine language file. This value will be 64 when the end of the file is reached.

After all data has been written to the BASIC program file, a BASIC FOR-NEXT loop will be created in L$, inserting the length of the machine language program as the loop count and the load address read used as the poke offset address (lines 500-540). This program line is then written to the BASIC

program file along with a zero link (two bytes, both zero) to indicate the end of the BASIC program created. Before closing all files, the utility program indicates the length of the BASIC program created.

I've included a short example of the type of program the utility program might create. Note that the line numbers of the BASIC program created start at 10 and increment by 10. If you want to change the starting line number, simply redefine the value of LN in line 180 as desired. Likewise, changing the value of LI in line 190 will change the increment between line numbers.

As I mentioned earlier, the program creates BASIC program lines limited to 78 or fewer characters when displayed on the screen. This allows you to use the screen editor to change the lines if necessary. If you want to compact things and don't care about not being able to edit lines, you can change the test value in line 450 from 65 to 250. The larger value will cause the utility program to create the maximum length program lines that BASIC can handle. The lines can be displayed but cannot be edited!

As usual, I'll supply copies of this program on disk for $5 to cover costs. I'll take care of the disk, mailer, and first class postage.

```
100 rem ******************************
110 rem
120 rem    d a t a  b u i l d e r
130 rem
140 rem       by: robert w. baker
150 rem
160 rem ******************************
170 :
180 ln=10: rem starting basic line number
190 li=10: rem line number increment
200 :
210 print"[CLR]machine language pgm"
220 print"to be converted is -": print
230 input f$
240 open 15,8,15
250 open 1,8,5,"0:"+left$(f$,16)+",p,r)"
260 input#15,en,em$: if en<>0 then print"disk error
-";en;em$: goto 650
270 print: print"basic pgm to be built"
280 print"should be called -": print
290 input f$
300 open 2,8,6,"0:"+left$(f$,16)+",p,w"
310 input#15,en,em$: if en<>0 then print"disk error
-";en;em$: goto 650
320 print: print"ok,": print"building new pgm file":
 print
330 get#1,c$: if st<>0 then 640
340 ad=0: if c$<>"" then ad=asc(c$)
350 get#1,c$: if st<>0 then 640
360 c=0: if c$<>"" then c=asc(c$)
370 ad=ad+(256*c): print"starting address =";ad: pri
nt
380 lk=1025: nb=0
390 print#2,chr$(1);chr$(4);
400 l$=chr$(131): rem "data" token
410 get#1,c$: ss=st: if ss<>0 then 470
420 c=0: if c$<>"" then c=asc(c$)
430 if len(l$)>1 then l$=l$+","
440 l$=l$+mid$(str$(c),2): nb=nb+1
450 if len(l$)<65 then 410
460 gosub 580: goto 400
470 if ss<>64 then 640
480 if len(l$)>1 then gosub 580
490 if nb=0 then 650
500 rem following lines create a basic line
510 rem for x=0 to ...:read c: poke...+x,c:next
520 l$=chr$(129)+"x"+chr$(178)+"0"+chr$(164)
530 l$=l$+mid$(str$(nb-1),2)+":"+chr$(135)+"c:"
540 l$=l$+chr$(151)+mid$(str$(ad),2)+chr$(170)+"x,c:
"+chr$(130)
550 gosub 580: print#2,chr$(0);chr$(0);
560 print"length =";nb+1;"bytes": print
570 print"done conversion": goto 650
580 l=len(l$): lk=lk+5+l: x=lk: gosub 620
590 x=ln: gosub 620: ln=ln+li
600 for x=1 to l: print#2,mid$(l$,x,1);: next
610 print#2,chr$(0);: return
620 x1=int(x/256): x2=x-(x1*256)
630 print#2,chr$(x2);chr$(x1);: return
640 print: print"disk error,": print"program aborted
"
650 close 1: close 2: close 15
```