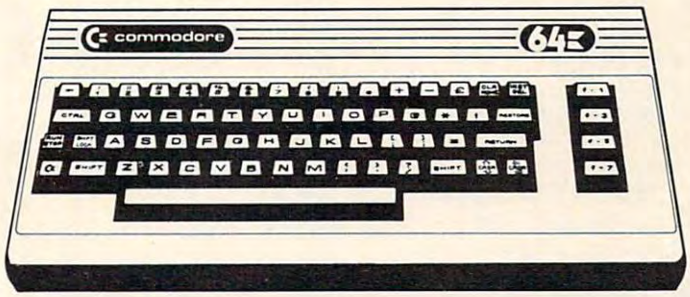
BUSIWRITER

BUSIWRITER A Honey of a Word Processor

Why word processors?

Word processors allow the user to quickly and easily create letters, memos, notes, reports, term papers, manuals, poetry and any other written information using the memory of the computer as a pencil and paper. The computer display or terminal acts as a window through which the user views the information as it is entered. The outstanding advantage of using BUSIWRITER is that it acts not only as a pencil and paper but as a perfect eraser and automatic typewriter.



For Commodore CBM-64
Commodore 1515, 1525, Epson, C. Itoh, Qume, Diablo, NEC Spinwriter, Starwriter, Prowriter, Okidata, Microline, Gemini-10
And many more printers

BUSIWRITER The Queen Bee of Word Processors

BUSIWRITER allows the user to quickly and easily make any number of alterations to the text. BUSIWRITER will instantly reformat your text and show you exactly and continuously how the final output will appear. BUSIWRITER has more functions than any other known microcomputer word processor. With BUSIWRITER assisting in the entry of text, providing a **20 page memory** and performing an enormous number of editing/composing functions, the preparation of written data is far faster and outstandingly more accurate than if it were prepared by hand.

BUSIWRITER With the Sting Removed from the Prices

BUSIWRITER 64. . . . . only **$99.00** for the CBM 64

BUSIWRITER AVAILABLE NOW FROM YOUR LOCAL DEALER
(800) 227-9998
FOR THE NAME OF YOUR NEAREST DEALER

California, Canada, Alaska and Hawaii please call (415) 965-1735

Skyles Electric Works
231G South Whisman Road
Mountain View, CA 94041

Europe please contact Supersoft, Winchester House, Harrow Wealdstone, England HA3 7SJ, Tel. 01 861 1166

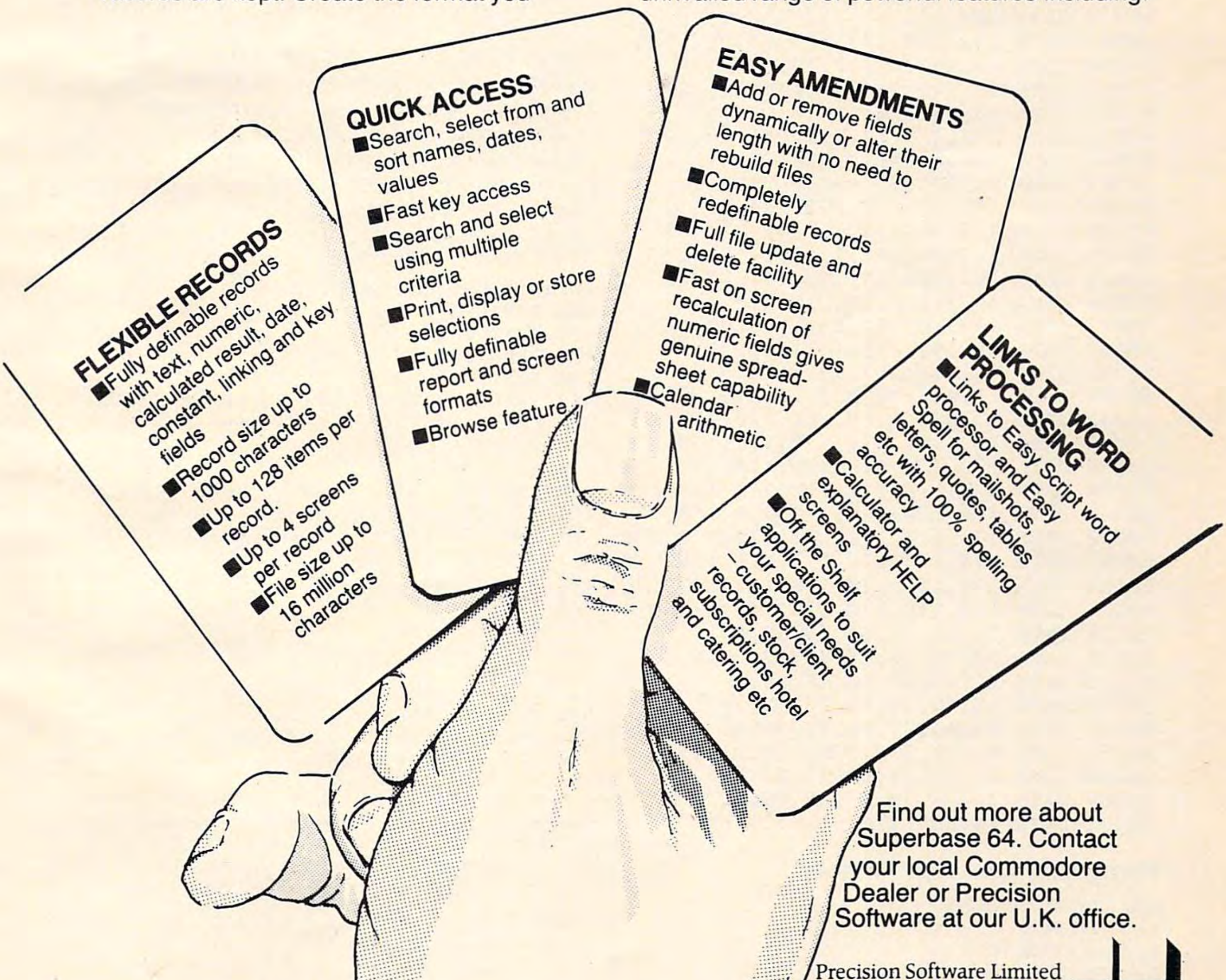
```
7424 DATA 80,24,165,81,109,5,-26
7432 DATA 133,81,165,82,109,6,-26
7440 DATA 133,82,136,208,223,96,160
7448 DATA 0,140,11,-26,177,79,141
7456 DATA 7,-26,177,81,141,8,-26
7464 DATA 200,152,205,7,-26,240,2
7472 DATA 176,13,205,8,-26,240,21
7480 DATA 144,19,238,11,-26,76,30
7488 DATA -29,205,8,-26,240,2,176
7496 DATA 62,206,11,-26,76,30,-29
7504 DATA 140,9,-26,160,1,177,79
7512 DATA 133,77,200,177,79,133,78
7520 DATA 172,9,-26,136,177,77,141
7528 DATA 10,-26,140,9,-26,160,1
7536 DATA 177,81,133,77,200,177,81
7544 DATA 133,78,172,9,-26,177,77
7552 DATA 200,205,10,-26,208,3,76
7560 DATA 195,-28,144,184,76,224,-28
7568 DATA 96,160,2,177,79,72,177
7576 DATA 81,145,79,104,145,81,136
7584 DATA 16,243,96,169,0,141,11
7592 DATA -26,173,17,-26,205,19,-26
7600 DATA 144,6,240,8,238,11,-26
7608 DATA 96,206,11,-26,96,173,16
7616 DATA -26,205,18,-26,144,244,208
7624 DATA 238,96,173,16,-26,24,109
7632 DATA 18,-26,141,16,-26,173,17
7640 DATA -26,109,19,-26,141,17,-26
7648 DATA 96,169,0,141,11,-26,56
7656 DATA 173,16,-26,237,18,-26,141
7664 DATA 16,-26,173,17,-26,237,19
7672 DATA -26,141,17,-26,176,3,206
7680 DATA 11,-26,96,238,16,-26,208
7688 DATA 3,238,17,-26,96,256
```

Program 3: Ultrasort For 64

```
10 I=49152
20 READ A:IF A=256 THEN END
30 POKE I,A:I=I+1:GOTO 20
49152 DATA 76,100,192,170,170,170,170
49159 DATA 170,170,170,170,170,170,170
49166 DATA 170,170,170,170,170,170,170
49173 DATA 170,170,170,170,170,170,170
49180 DATA 170,170,170,170,170,170,170
49187 DATA 170,170,170,170,170,170,170
49194 DATA 170,170,170,170,170,170,170
49201 DATA 170,170,170,170,170,170,170
49208 DATA 170,170,170,170,170,170,170
49215 DATA 170,170,170,170,170,170,170
49222 DATA 170,170,170,170,170,170,170
49229 DATA 170,170,170,170,170,170,170
49236 DATA 170,170,170,170,170,170,170
49243 DATA 170,170,170,170,170,170,170
49250 DATA 170,170,32,253,174,32,158
49257 DATA 173,32,247,183,165,20,141
49264 DATA 12,192,165,21,141,13,192
49271 DATA 32,253,174,32,158,173,56
49278 DATA 165,71,233,3,133,75,165
49285 DATA 72,233,0,133,76,162,1
49292 DATA 173,12,192,157,20,192,173
49299 DATA 13,192,157,40,192,169,1
49306 DATA 157,60,192,169,0,157,80
49313 DATA 192,189,60,192,141,16,192
49320 DATA 189,80,192,141,17,192,189
49327 DATA 20,192,141,18,192,189,40
49334 DATA 192,141,19,192,32,47,195
49341 DATA 173,11,192,48,4,202,208
49348 DATA 221,96,189,60,192,141,16
49355 DATA 192,189,80,192,141,17,192
49362 DATA 169,1,141,18,192,169,0
49369 DATA 141,19,192,32,101,195,189
49376 DATA 20,192,141,18,192,141,14
49383 DATA 192,189,40,192,141,19,192
49390 DATA 141,15,192,32,47,195,173
49397 DATA 11,192,48,3,76,167,193
49404 DATA 32,131,195,173,16,192,141
49411 DATA 3,192,173,17,192,141,4
49418 DATA 192,173,14,192,141,5,192
49425 DATA 173,15,192,141,6,192,32
49432 DATA 132,194,32,180,194,173,11
49439 DATA 192,48,218,173,16,192,141
49446 DATA 3,192,173,17,192,141,4
49453 DATA 192,173,18,192,141,16,192
49460 DATA 173,19,192,141,17,192,169
49467 DATA 1,141,18,192,169,0,141
49474 DATA 19,192,32,101,195,173,16
49481 DATA 192,141,18,192,173,17,192
49488 DATA 141,19,192,173,3,192,141
49495 DATA 16,192,173,4,192,141,17
49502 DATA 192,32,47,195,173,11,192
49509 DATA 16,35,173,14,192,141,3
49516 DATA 192,173,15,192,141,4,192
49523 DATA 173,18,192,141,5,192,173
49530 DATA 19,192,141,6,192,32,132
49537 DATA 194,32,180,194,173,11,192
49544 DATA 48,152,32,47,195,173,11
49551 DATA 192,16,18,173,16,192,141
49558 DATA 3,192,173,17,192,141,4
49565 DATA 192,32,132,194,32,31,195
49572 DATA 76,241,192,234,189,20,192
49579 DATA 141,3,192,189,40,192,141
49586 DATA 4,192,173,16,192,141,5
49593 DATA 192,173,17,192,141,6,192
49600 DATA 32,132,194,32,31,195,173
49607 DATA 16,192,141,18,192,141,3
49614 DATA 192,173,17,192,141,19,192
49621 DATA 141,4,192,32,81,195,189
49628 DATA 20,192,141,18,192,189,40
49635 DATA 192,141,19,192,32,101,195
49642 DATA 173,11,192,48,15,189,60
49649 DATA 192,141,18,192,189,80,192
49656 DATA 141,19,192,32,101,195,169
49663 DATA 1,141,18,192,169,0,141
49670 DATA 19,192,173,3,192,141,16
49677 DATA 192,173,4,192,141,17,192
49684 DATA 173,11,192,16,52,189,60
49691 DATA 192,232,157,60,192,202,189
49698 DATA 80,192,232,157,80,192,32
49705 DATA 101,195,173,16,192,157,20
49712 DATA 192,173,17,192,157,40,192
49719 DATA 32,131,195,32,131,195,202
49726 DATA 173,16,192,157,60,192,173
49733 DATA 17,192,157,80,192,76,128
49740 DATA 194,32,131,195,232,173,16
49747 DATA 192,157,60,192,173,17,192
49754 DATA 157,80,192,202,189,20,192
49761 DATA 232,157,20,192,202,189,40
49768 DATA 192,232,157,40,192,202,32
49775 DATA 101,195,32,101,195,173,16
49782 DATA 192,157,20,192,173,17,192
49789 DATA 157,40,192,232,76,162,192
49796 DATA 160,3,165,75,133,79,133
```

Strengthen your hand with Superbase 64

The complete information control system for the Commodore 64. Ideal for any home, business or professional environment where records are kept. Create the format you need and enter your records. If the layout or data field sizes are not quite right, correct them and carry on. Superbase gives you an unrivalled range of powerful features including:



Find out more about Superbase 64. Contact your local Commodore Dealer or Precision Software at our U.K. office.

Precision Software Limited
Park House 4 Park Terrace
Worcester Park
Surrey KT4 7JZ England
Telephone: 01-330 7166
Telex: 8955021 PRECIS G

Precision Software

OFFICIALLY APPROVED BY COMMODORE

```
49803 DATA 81,165,76,133,80,133,82
49810 DATA 24,165,79,109,3,192,133
49817 DATA 79,165,80,109,4,192,133
49824 DATA 80,24,165,81,109,5,192
49831 DATA 133,81,165,82,109,6,192
49838 DATA 133,82,136,208,223,96,160
49845 DATA 0,140,11,192,177,79,141
49852 DATA 7,192,177,81,141,8,192
49859 DATA 200,152,205,7,192,240,2
49866 DATA 176,13,205,8,192,240,21
49873 DATA 144,19,238,11,192,76,30
49880 DATA 195,205,8,192,240,2,176
49887 DATA 62,206,11,192,76,30,195
49894 DATA 140,9,192,160,1,177,79
49901 DATA 133,77,200,177,79,133,78
49908 DATA 172,9,192,136,177,77,141
49915 DATA 10,192,140,9,192,160,1
49922 DATA 177,81,133,77,200,177,81
49929 DATA 133,78,172,9,192,177,77
49936 DATA 200,205,10,192,208,3,76
49943 DATA 195,194,144,184,76,224,194
49950 DATA 96,160,2,177,79,72,177
49957 DATA 81,145,79,104,145,81,136
49964 DATA 16,243,96,169,0,141,11
49971 DATA 192,173,17,192,205,19,192
49978 DATA 144,6,240,8,238,11,192
49985 DATA 96,206,11,192,96,173,16
49992 DATA 192,205,18,192,144,244,208
49999 DATA 238,96,173,16,192,24,109
50006 DATA 18,192,141,16,192,173,17
50013 DATA 192,109,19,192,141,17,192
50020 DATA 96,169,0,141,11,192,56
50027 DATA 173,16,192,237,18,192,141
50034 DATA 16,192,173,17,192,237,19
50041 DATA 192,141,17,192,176,3,206
50048 DATA 11,192,96,238,16,192,208
50055 DATA 3,238,17,192,96,170,170
50062 DATA 170,170,170,170,170,170,170
50069 DATA 170,170,170,170,170,170,170
50076 DATA 170,170,170,170,170,170,170
50083 DATA 170,170,170,170,170,170,170
50090 DATA 170,170,170,170,170,170,170
50097 DATA 170,170,170,170,170,170,170
50104 DATA 170,170,170,170,170,170,170
50111 DATA 170,170,170,170,170,81,85
50118 DATA 73,67,75,83,79,82,84
50125 DATA 32,76,79,65,42,32,32
50132 DATA 3,255,50,48,44,82,69
50139 DATA 65,68,32,69,82,82,79
50146 DATA 82,44,49,56,44,48,48
50153 DATA 0,170,170,170,170,81,85
50160 DATA 73,67,75,83,79,82,84
50167 DATA 32,76,79,65,68,69,82
50174 DATA 16,255,256
```

Program 4: Sort Test Program

```
100 PRINT "{CLR}"
110 N=1000
120 DIM AA$(N)
130 PRINT "CREATING"N" RANDOM STRINGS"
140 SD=-TI : A=RND(SD)
150 FOR I=1 TO N
160 PRINT I"{UP}"
170 N1=INT(RND(1)*10+1)
180 A$=""
190 FOR J=1TO N1
```

Special PET Version Note

PETs with BASIC 4.0 do not have the problem of lengthy *garbage collection times* (this occurs when the computer finds that it has run out of memory, and must eliminate all strings that are no longer "active"). The price of this convenience is that all dynamic strings are now two bytes longer. Those two bytes are a "back-pointer" from the top of the memory (where the actual data contained in the string is kept) to the bottom of memory where the variable keeps a pointer to that data.

This sort does *not* modify the back-pointers. So, if after sorting you continue using the new data, it will eventually be garbled.

There is a solution. Immediately after sorting, write the data to disk as a file. Then issue a CLR command. This will remove all your variables. Then read the data back off the disk into a new array.

This problem does not occur on the VIC-20 or the Commodore 64.

```
200 B$=CHR$(INT(RND(1)*26+65))
210 A$=A$+B$
220 NEXT J
230 AA$(I)=A$
240 NEXTI
250 PRINT "HIT ANY KEY TO START SORT"
260 GET A$:IF A$="" THEN 260
270 PRINT "SORTING..."
280 T1=TI
290 REM SYS 31744,N,AA$(1) FOR PET/CBM
291 REM SYS 49154,N,AA$(1) FOR 64
292 REM USE SYS VALUE GENERATED BY THE
    LOADER FOR VIC
300 SYS 31744,N,AA$(1)
310 T2=TI
320 PRINT "DONE"
330 PRINT "HIT ANY KEY TO PRINT SORTED S
    TRINGS"
340 GET A$:IF A$="" THEN 340
350 FORI=1TON:PRINT I,AA$(I):NEXT
360 PRINT:PRINT N" ELEMENTS SORTED IN"(T
    2-T1)/60"SECONDS"
```
©

INSIGHT: Atari

Bill Wilkinson

The new 600XL and 1400XL computers were exactly what I expected (except that Atari goofed and changed the number on the 1201 XL – and that's a joke until you study the case designs of the 1200XL and 1400XL). The 800XL was a little bit of a surprise, but kind of a logical step now that I have the benefit of hindsight. The 1450XLD was a pure delight.

I really could envision a 1450XLD doing some nice, small business work. Especially if you put one of the new three-inch hard disk drives (that's over four megabytes of disk space) into that empty space supposedly designed for a second floppy.

If Atari has any problems at all with the XL line of computers, it may be simply that they are priced too close together. After all, an 800XL is essentially a 600XL with 64K of RAM, and the already announced RAM-pack for the 600XL ends up producing an equivalent machine for the same price. Redundancy.

The 1400XL suffers a little, also. After all, if the rumored price of the 1450XLD holds up ($800-$900 retail), why would you buy a 1400XL and then add a snail's-pace 1050 drive when you can have the much faster XLD for less money? And who but the more sophisticated user will buy a 1400XL when the 600XL (even with expansion to 64K) is so much less? Will the modem and speech synthesizer really prove attractive to a first-time user? Atari marketing obviously thinks so. I think that people who know they want those features will also know enough to want a disk drive.

Anyway, all of that is crystal-balling and nit-picking on my part. The new lineup of computers is one that any company could be proud of. Atari should be doubly complimented after the fiasco with the 1200.

The New Disk Drives

Before I stop making observations about Atari, though, I would like to carp a bit about one thing: the new Atari disk drives and DOS III (or is it DOS 3?). When I first heard that Atari was going to throw away a potential 50K per disk drive, I thought there was an almost-good excuse. After all, Atari DOS 2.0S could, with absolutely minimum modifications, utilize all the sectors of the one-and-one-third density 1050 drive, so the change, though inefficient when compared to true double-density drives, would allow many current programs to work without modification.

It is not to be. Atari DOS III is just as different from DOS 2.0S as our own Version 4 OS/A+ is. Which means many, many programs (including data base programs, etc.) simply will not work without modification. I do *not* feel this is inherently bad. Let's face it: DOS 2.0S is not a particularly good DOS and it is totally inadequate for larger disk drives. DOS III is actually a very nice DOS for small drives (say up to 128K per drive). It goes downhill rapidly when used on larger drives. This means that if you convert your programs and data files from DOS 2.0S to DOS III this year, you will have to convert to some other DOS again next year, when you move to one of those nice little hard disks I mentioned.

Anyway, when the 1050 finally appears, watch here (I hope) for instructions for using DOS 2.0S (or OS/A+ Version 2) in one-and-one-third density mode, so you won't have to convert all your programs. (You'll still have to convert the diskettes themselves, which won't be easy or fast if you only have one drive, but the same holds true of DOS III – and, to be fair, OS/A+ Version 4 – so you won't have lost anything.)

Self-Relocatable Machine Language, Part III

This month, I will discuss some more techniques which can be used to make your machine language self-relocatable. Last month, we noted which kinds of instructions were implicitly "safe" (register-only instructions, branches, etc.). There was also a list of "Safe Relocatable Techniques." To summarize, the safe techniques mentioned were:

1. Change JMPs to branches.

2. Save register values in the stack, not in fixed memory.

3. From BASIC, pass the address of a string as a location (or series of locations) to load from or store to.

4. Move code from relocatable memory to fixed memory temporarily.

I also promised to discuss two points this month: (1) where the "safe" locations in Atari memory are; and (2) some special techniques usable only with Atari BASIC. Let me fulfill my promise.

Safe Locations

There are none. Next topic.

GREAT PRODUCTS for GREAT COMPUTERS

BOOKS

ATARI BASIC. Learning by Using. This is an action book. You program with it more than you read it. You use it, you discover with it. Learn ATARI BASIC easily through the short programs provided. A great source of work problems for teacher or student. 73 pages.
ISBN 3-92-1682-86-X $5.95

Games For The ATARI. Provides ideas on how to create your own computer games. Contained primarily BASIC examples but, for very advanced programmers, a machine language example is included at the end of the book. 115 pages.
ISBN 3-911682-84-3 $7.95

How to Program Your ATARI In 6502 Machine Language. To teach the novice computer user machine language, the use of an assembler, and how to call subroutines from the BASIC interpreter. 106 pages.
ISBN 3-921682-97-5 $9.95

FORTH on the ATARI. Explore this versatile programming language with numerous graphics and sound examples. Designed for both the novice and experienced programmer. 118 pages.
ISBN 3-88963-170-3 $7.95

ATARI BASIC Faster and Better. Programming tricks and techniques. Three companion software diskettes available (sold separately). 280 pages. ISBN 0-936200-29-4 $29.95

SECRETS OF ATARI I/O. Theory of operation and application programs for input/output to disk, screen, cassette, and RS232 serial port. Machine language with POKE tables for use with BASIC programs. Companion software available on disk (sold separately). 285 pages.
ISBN 0-936200-33-2
$29.95 retail

SOFTWARE

CASDUP 1.0 & 2.0. To copy most BOOT tapes and cassette data files. 1.0 is a file copier. 2.0 is a sector copier. Cassette only $24.95

CASDIS. To transfer most BOOT tapes and cassette data files to disk. Disk only $24.95

FULMAP. BASIC Utility Package. VMAP-variable cross-reference, CMAP-constant cross-reference (includes indirect address references), LMAP-line number cross-reference, FMAP-all of the above. Will list "unlistable" programs. Also works with Editor/Assembler cartridge to allow editing of string packed machine language subroutines. All outputs may be dumped to printer. Cassette or disk $39.95

DISASM. To disassemble machine language programs. Works with or without Editor/Assembler cartridge. May be used to up or down load single boot files. All output can be dumped to printer. Cassette or Disk $24.95

DISDUP. For disk sector information copying. May specify single sector, range of sectors, or all. Copies may be made without read verify. Disk $24.95

V-COS Cassette Operating System. Control baud rate, leader time, screen width, background and letter color, cassette motor (on/off); provides cassette file verification. Cassette $24.95

DOWNLD Diskette Download Utility. Allows single BOOT files and Binary DOS files to be transferred from disk to cassette. Fast, easy, menu-driven. NOT FOR PROTECTED SOFTWARE. Disk $19.95

DISKPAK. A program that frees the unused sectors on a boot disk for storage of normal DOS files without disturbing the boot file. May be used on all boot files including multi-stage files. NOT FOR PROTECTED SOFTWARE. Disk $24.95

ABFAB Assembly Disk. Companion software to ATARI BASIC Faster and Better. Ten assembly language source programs and ten object programs. Disk $19.95

ABFAB Library Disk. 81 subroutines that can be included in your BASIC programs. Includes BASIC and machine language (some programs POKEd into memory). Disk $19.95

ABFAB Demo/Applications Disk. Eleven application programs and fourteen demonstration programs from the ATARI BASIC Faster and Better book. Disk $19.95

SECRET Library Disk for the ATARI. More than a dozen I/O routines that exemplify material in SECRETS OF ATARI I/O (sold separately). Includes Super Menu, Screen Dump, BASIC AutoRUN, Binary Loader, String Search, Disk Copier, Cassette Copier and much, much more. Disk $19.95

IJG products are available at computer stores, B. Dalton Booksellers and independent dealers around the world.

If IJG products are not available from your local dealer, order direct. Include $4.00 for shipping and handling per item. Foreign residents add $11.00 plus purchase price per item. U.S. funds only, please.

IJG, Inc. 1953 W. 11th Street
Upland, California 91786
Phone: 714/946-5805

IJG

ATARI TM Warner Communications, Inc.

Oh, all right, I admit that is a bit of an exaggeration, but it is dismayingly close to the truth. When I write machine language routines, I really do prefer that they be usable with as many products as possible. Just as a start – and *not* as a comprehensive list – I would hope that they would work with the following software: Atari BASIC, Atari DOS, OS/A+, BASIC A+, Atari Microsoft BASIC, *Atariwriter*, Atari Assembler Editor Cartridge, MAC/65, AMAC, and a few more.

Okay. Not too long a list. How many zero page locations are not used by any of those? None. How many Page Six locations ($600 through $6FF) are not used by any of those? None. How many.... But I think you get the idea. Is all this strictly true?

Actually, there are quite a few bytes which can be used for *your* temporary storage. And I suggest you consult your *Atari Technical User's Notes* or *Mapping the Atari* (from COMPUTE! Publications) to find where they are. (Caution: Watch out for changes in the new XL computers.) But even these locations are suspect. What happens if I write this neat new printer-spooler routine which uses location $00 (believe it or not, that's free in almost all the above programs), and then you come along and add a driver for graphics mode 27 and you use location $00?

Perhaps I am being a bit of a purist here. Certainly very little of my own programming is this clean, this free of conflict with other potential programs. And yet it really does require only a little more work to write a program "correctly" (by my definition), so why not do it right? Let's try.

So, we must assume that *no* location outside our own, self-relocatable, properly-loaded-at-LOMEM program is safe at all times. Unpleasant. However, that does *not* say that we can't use some almost-safe locations while our routine has control. In particular, you should be able to use several reserved locations in zero page (for indirect-Y pointers, etc.) by, if necessary, moving values into them from within your relocatable block; using and/or changing the zero page locations in your program; and then moving the values back into your relocatable block.

Sounds complicated? It is. And yet you might be surprised at how seldom you really need to go through all that.

So what zero page locations are safe, even as temporaries? Probably the safest spots, as long as you aren't writing an interrupt handler, are those locations used as temporaries by the DOS File Manager. Locations $43 through $49, inclusive, are always reinitialized by FMS every time it gets control. FMS does *not* presume the locations have maintained their contents from one call to the next. (In fact, the locations should properly be called "Device Driver Zero Page Temporaries," since that is what they were intended for.)

And one more comment before I leave you with the impression that absolutely nothing is safe to do on the Atari computers. If you are writing routines specifically designed to be used with Atari BASIC (as I suspect the majority of you are), there are several safe temporaries. First, you can always use the floating point work area, $D4 through $EF, whenever BASIC calls either a USR routine or an I/O routine. Also, BASIC does not use locations $CB through $CF (only four bytes!) at all. Again, let me give you the caution about adding your routine to a system which already has a custom routine. Be sure there is no conflict.

A Built-In Relocatable Pointer

It's true. There really is such a thing. There are some *if*s though: *if* you are using Atari BASIC or OSS BASIC A+ or OSS BASIC XL; *if* you have placed your relocatable program in a string and are calling a machine language routine via USR(ADR(STRING$)) or USR("...machine-language-string..."); *if* you don't mind a small trick.

First, the trick. It's really quite simple. Whenever BASIC calls a USR routine, it calls the routine by placing the routine's calculated address in location $D4-$D5 (which just happens to be the first two bytes of floating point register zero). It then JSRs to a routine which simply does a "JMP ($D4)", a jump indirect to the USR routine.

But why can't we take advantage of that pointer? It already points to our relocatable program, so why can't it point to our relocatable data? Perhaps a demonstration is in order.

```
FRØ     =     $D4
USRROUTINE
        CLC
        BCC   START        ; branches are ok
;
SAVEBYTE
        .BYTE Ø            ; some data
;
; begin actual code
;
START
        LDY   #SAVEBYTE-USRROUTINE  ; index
        PLA                ; count of parameters
        CMP   #1           ; how many?
        BNE   NOPARAMS     ; none, we presume
; the user is passing a byte to us
        PLA                ; high byte...ignored
        PLA                ; low byte...stored
        STA   (FRØ),Y      ; thusly
; we join here, whether a byte is passed or not
NOPARAMS
        LDA   (FRØ),Y      ; get the byte
        STA   FRØ          ; to be returned
        LDA   #Ø
        STA   FRØ+1        ; high byte zero
        RTS
```

This program is a *very* dumb one, for demonstration purposes only. If you call it from BASIC via, for example, "PRINT USR(routine)", your program will print the byte value saved in location SAVEBYTE (zero, initially). On the other hand, if you use "JUNK = USR(routine, 97)", the routine will store the second parameter (97) in location

SAVEBYTE. Presumably, you could then later recover the 97.

The point to be made, however, is that this program is completely self-relocatable and yet is able to load and store data from within its own relocatable block! The secret is the "LDY #SAVE-BYTE-USRROUTINE" line directly after the label START. Since location FR0 contains the address of USRROUTINE, loading the Y-register with the proper offset (SAVEBYTE-USRROUTINE, which happens to be 3 in our example) will allow us to do indirect loads and stores to any location within 255 bytes following USRROUTINE.

Can I put that more clearly? Since, when we do either the "LDA (FR0),Y" or the "STA (FR0),Y", the Y register contains the value 3 and location FR0 points to the location USRROUTINE, the LDA and STA instructions will reference the third byte after USRROUTINE. Which just happens to be SAVEBYTE.

And just a reminder if you don't know or remember what the PLA instructions in this program are for. Whenever BASIC calls a USR routine, it pushes all the parameters it is given onto the CPU stack (after first converting them to 16-bit integers, of course). Then, the last thing it does before the call is to push a count of the number of parameters (presumed to be 1 or 0 in our example) onto the same stack. Thus, the first PLA lets us discover how many parameters were passed. The other two PLAs are necessary if a parameter is passed; otherwise the RTS instruction will return to an unknown location and will likely crash the system. (Note that in our simple-minded example you can probably crash BASIC by calling the routine with two parameters, since no check is made for more than one parameter.)

Next month we're going to take this technique a couple of steps further. We will discover how to have more than 255 bytes of relocatable storage (which may or may not be useful to you) and how to generate similar self-pointers when the routine in question has not been called from BASIC. ©

Easy Atari Page Flipping

Chris Allen

Here's a short program that lets you display one screen creation while drawing another offscreen. Put them together and you've got page flipping.

Have you ever wished that you could just POKE a couple of locations and have a complicated picture appear on your Atari? This demo program will show you how to use *page flipping* – changing the addresses that tell the Atari where screen memory is. Page flipping will allow you to show one picture and, at the same time, draw another picture offscreen. You don't see it drawn: it just "appears" instantly.

Page flipping allows you to draw offscreen using the normal graphics commands (PLOT, DRAWTO, etc.), or, if you use a text mode, to PRINT normally. You don't have to do any spectacular POKEing.

The method is simple. The Atari keeps two separate two-byte registers for the address of screen memory. The first register, locations 88 and 89 (decimal), is used *solely* for PRINTing, PLOTting, etc.; it is not concerned with display. The second register, bytes five and six of the display list (located by PEEK(560) + PEEK(561)*256), is used *only for display*. Having two locations simplifies matters – changing the first register allows you to draw offscreen, while changing the display list register will "flip" your new screen into view.

A few cautions are in order. First, page flipping uses a lot of memory. Since one GRAPHICS 7 screen uses 3200 bytes, two such pictures are impossible on an 8K machine. However, GRAPHICS 5 uses only 800 bytes, ideal for computers with limited memory. Second, be sure to clear any garbage from the area you have reserved for your new screen. Third, if you modify the display list, be aware that your new display list may not have the screen address register in the same location as a normal list. (If you can change the display list, you should be able to handle this minor problem.)

Now that the warnings are out of the way, let's do some page flipping. First, type in this short program:

```
10 GRAPHICS 5
20 GOSUB 200
60 END
200 COLOR 1:FOR I=0 TO 79:PLOT I,0:D
    RAWTO I,39:NEXT I:RETURN
```

When you run it, notice that you can see the screen being filled in. Now add these lines to enable page flipping:

```
5 POKE 106,PEEK(106)-4:SCREEN2=PEEK(
  106)
15 SCREEN1=PEEK(89):POKE 89,SCREEN2
25 B=PEEK(560)+PEEK(561)*256
30 FOR I=1 TO 100
35 POKE B+5,SCREEN2
40 FOR J=1 TO 200:NEXT J
45 POKE B+5,SCREEN1
50 FOR J=1 TO 200:NEXT J
55 NEXT I
```

The picture is drawn offscreen, where you can't see it. By switching values in the display register (B + 5 is the sixth, or high byte), you can alternate or "flip" between screens. Here's a line-by-line explanation:

Line 5 reserves memory for the second screen and sets up a pointer to the reserved area.

Line 15 sets a pointer to the present screen, then flips the draw register over to screen two.

Line 25 finds the start of the display list.

Lines 30-55 simply loop 100 times, alternating the screen displayed each time.

Although we changed only the high byte, the low byte (88 or B + 4) can also be changed. (Try changing just B + 4 – and you're screen scrolling.) ©

How To Create A Data Filing System

Part III. Planning The Input

Jim Fowler

A little foresight in planning your input can save a lot of time and frustration. In Part III, the author tells how to handle some common input problems and offers some advice on how to prevent problems down the road.

In the first two installments we discussed setting goals for the kind of system you want, the types of files, and what kind of output is best. For most cases, a relative file structure with index files gives flexibility and speed. The index files will be composed of index words which will be either shortened versions of data in the records themselves or bytes encoded with some kind of bitmapping.

Before discussing input strategies, let's review some of the ideas from Part II in a bit more detail. We discussed setting up a buffer for inputting keys or index words. This buffer can be any free area of unused RAM memory. It must be large enough to accommodate the record or field to be compared. For example, if your index word is the first eight letters of the author's name, create an eight-byte buffer for your comparison.

A Closer Look At Indexing

Another technique we discussed was building your index file into your record format. For example:

AUTHOR	SUBJECT	TITLE	YEAR	INDEX	AUTHOR

←——— Record 1 ———→←— Record 2 ——

After entering your first record – author, subject, title, year – you can reserve several bytes at the end of that record to create an index file. If you choose to bitmap here, as illustrated in last month's installment, you gain search efficiency, although it may at first seem tedious when creating the index this way.

If you use one byte in the index for each field, you then have 256 possibilities for each field, which in most cases would be more than adequate. Using last month's illustration, a bit configuration of 1000 0000 would indicate a subject on computers. Since the integer equivalent of a binary 1000 0000 is 128, you can use this with an AND for compare. Let's say you've chosen the variable SU (for subject field). The appropriate line would be:

```
IF SU AND 128 THEN GOTO n
```

where n is a line that will direct a PRINT to screen or printer.

When using an AND, the computer will test individual bits. The value in SU, 1000 0000, is compared to 128:

```
1000 0000  (SU)
1000 0000  (128)
---------
1000 0000  (result)
```

The Boolean truth table, remember, makes this compare result "true," thus a "hit" is made in your search.

In some cases, depending on the total number of subjects you want to index, it might be practical to assign variable names to the binary equivalents:

A=1	E=16
B=2	F=32
C=4	G=64
D=8	H=128

Then, IF SU AND H THEN n.

Let's say you're searching for a more specific

subject, computers in education. We'll assign the subject of education a binary 0100 0000 (or integer 64). A computer subject, remember, was assigned 1000 0000 (128). A book dealing with computers and education would then be 1100 0000 (192). Our search statement would be:

```
IF SU AND 192 THEN n
```

Obviously, if you use this method, you'll have to be very thorough in creating your index. No matter what method of indexing you choose, do it carefully – your search speed and accuracy depend on it.

If you choose not to use the bitmapping method, a word of caution is in order: be sure to write the data that makes up your index file(s) also in the *records themselves*. You may later decide to change the format of an index file to rewrite a search routine. Maybe you will be forced to do this to accommodate an index file you found you needed. The easiest way to create the new index file is to read it item by item from the disk and assemble the index that way, rather than to type it in by hand. The accuracy will be much greater. Remember that one wrong bit in an index makes the record it refers to "invisible" to a search.

System Input Problems

Now for the problems with input. You want a system which is easy to use. This means giving cues that tell the user what is going on. One way is to use the top one or two lines on the screen to indicate what the program is doing or expecting at all times. Another important feature is to make the screen format logical and easy to understand.

Finally, when inputting new records, there should be ample opportunity to edit, erase, change, or abort without disturbing or crashing the program.

Some computers, including my CBM, cannot handle a string input containing commas. The operating system looks for these delimiters in an input string. When I input titles of publications, commas are important punctuation. That means I have to use a roundabout way of getting the string in without having it cut off at the comma. There are several ways of doing this. You can use GET and assemble the string byte by byte.

I have used a nice routine for Commodore equipment written by Jerry Dunmire (COMPUTE!, December 1981). This routine takes up to 80 characters in a string which can contain any symbols you want. If the 80-character limit is exceeded, you can tell by the value of ST, a status byte in the operating system. Problems like this should be handled at the outset. Make the system easy to use. A little frustration becomes a big one when you are typing in data. Having to substitute something else for commas would be very frustrating.

One thing to remember in connection with input is that the program must "know" at all times the number of records on the disk and the length of each index file. When you enter a new record, it must go into the very next empty location on the disk. The new record's index words must be put at the end of the appropriate index files. The way to save this information from one run to the next is to have a register pointing to the next record number. Inputting a new record will cause the register to be incremented by one. When you SAVE the index files, you should also SAVE this register and if the register is adjacent to the index files, you can save them all at once.

Writing The Input

Any writing of data should be done as it is input. For example, if there is to be a change from ASCII letters (or in my case, PETSCII), then that ought to be done when the time delay is not objectionable. After you type a name, and after you have a chance to edit it, you should be asked to give a final approval. Once this is given, the program ought to translate parts of the input before writing (sending the input) to the disk. This might take a few seconds, but if you are typing records from a list or card file, you will be reading the next item or moving the pointer on the copy stand while this goes on.

For example, this is how I handle my index file of authors. On the disk, the author's name is in capital and lowercase, last name first, with commas and periods after initials. In the index file all letters are written as pseudo-ASCII caps, and the index word ends with the eighth letter of the last name. To make pseudo-ASCII, all you need to do is shorten each ASCII byte to five bits with "AND 31" (or AND #$1F). If the last name is shorter than eight letters, I let the following comma and initials appear, too. The key used in searching for an author is also changed to pseudo-ASCII caps. After the last letter, the extra bytes, if any, are nulls. As mentioned, the search program then considers it a match when the next byte of the key is a null. That way you can search for SMITH,J. or SMITH, or even all the S's. That's very helpful when you aren't sure about the spelling of a name. Program 1 in the previous article illustrates this search technique.

Bitmapping is not hard. You can do it in machine language, but there is no particular advantage in doing so, except saving program space. The byte in question is zeroed and then the *n*th power of two is added to it whenever you want the bit in the *n*th position set. You can clear the same bit by subtracting. Be sure the bit is set before you do any subtracting and vice versa, and be sure it is clear before setting it. You must arrange it so the user cannot inadvertently set a bit twice

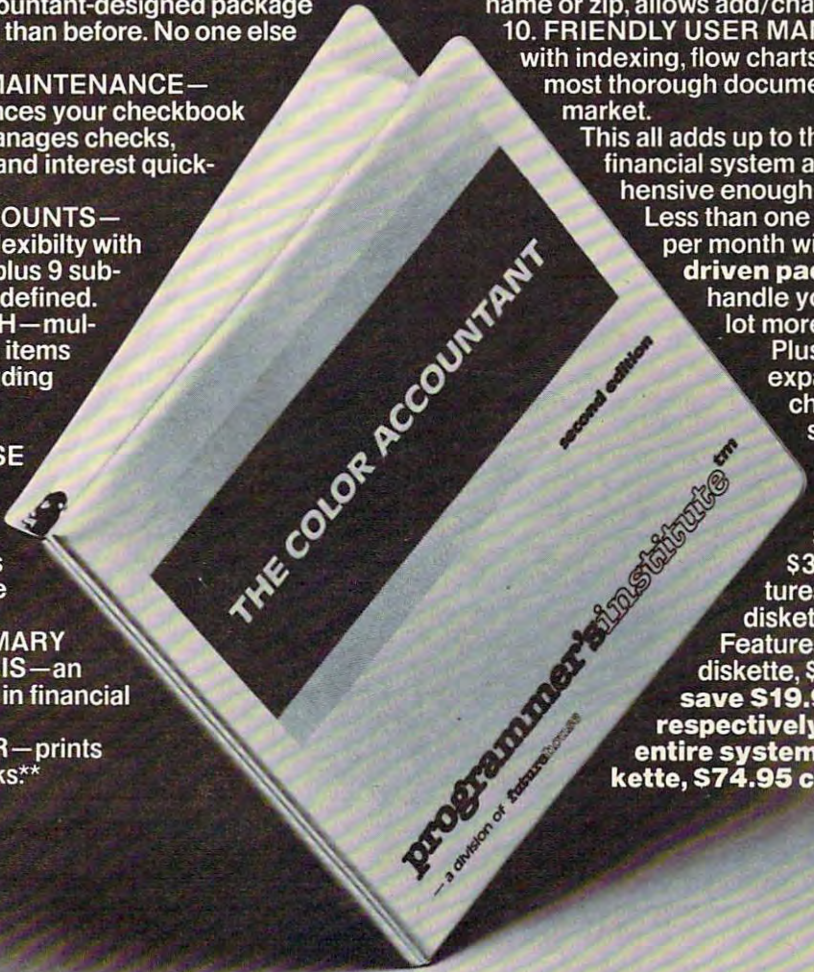
Complete Personal Accountant: we've made the best much more friendly.

If you have any doubts that we offer the best and most complete personal financial package available, look over the features listed below. Now we have the only package with **full screen editing** for Atari 400/800,* TRS-80 COLOR, Commodore 64* and VIC-20; the ability to move the cursor in any direction makes our accountant-designed package even more friendly than before. No one else offers all of these:

1. CHECKBOOK MAINTENANCE—automatically balances your checkbook with each entry; manages checks, charges, deposits, and interest quickly and accurately.
2. CHART OF ACCOUNTS—maximum of user flexibilty with up to 99 accounts plus 9 subcategories may be defined.
3. CHECK SEARCH—multi-reference; tracks items on every field including tax deductibles.
4. NET WORTH/INCOME/EXPENSE STATEMENT—know-exactly-where-you-stand program generates statements with the touch of a key.
5. DETAIL & SUMMARY BUDGET ANALYSIS—an absolute necessity in financial planning.
6. CHECK WRITER—prints personalized checks.**
7. PAYMENTS/APPOINTMENTS CALENDAR—monthly displays of up to 250 bills and 200 appointments.
8. COLOR GRAPH DESIGN PACKAGE—graphs all monthly files.
9. MAILING LIST—maintains all records, sorts by name or zip, allows add/change/delete.
10. FRIENDLY USER MANUAL—complete with indexing, flow charts and diagrams; the most thorough documentation on the market.

This all adds up to the finest personal financial system available—comprehensive enough for a small business. Less than one hour of data input per month will allow this **menu-driven package** to help you handle your finances with a lot more fun than drudgery.

Plus, ours is the only expandable system; purchase the package in sections and add on as your financial needs grow. Features 1, 2, 3 and 6: $39.95 diskette, $36.95 cassette; Features 4 and 5: $29.95 diskette, $26.95 cassette; Features 7, 8 and 9: $29.95 diskette, $26.95 cassette; **or save $19.90 or $15.90 respectively by buying the entire system for $79.95 diskette, $74.95 cassette.**



*Random Access available

Prices subject to change without notice. See your local dealer or order direct. New catalog available. Add $3.00 for postage and handling. Credit card orders call toll free:

1-800-334-SOFT

DEALER INQUIRIES INVITED

programmer's institute

a division of futurehouse

p.o. box 3470, dept. C, chapel hill, north carolina 27514, 919-967-0861

Routine To Set Bits In An Index Word.

(This routine is based on Y/N response with cursor moving down list on screen. You must arrange a stop or wraparound when N gets to maximum and P=7. Same at N=0:P=0.)

1. DIM the array IW(x) to nr of bytes in index word.
   Zero IW if not already done initially.
2. Print subjects in list on screen.
   N=0;P=0 Zero byte nr and bit nr.
   Print "cursor" opposite zeroth subject.
3. GET loop.
4. If SPACE, move cursor down: P=P+1.
   IF P=8, then N=N+1;P=0.
5. If SHIFT-SPACE, move cursor up.
   IF P=-1, then N=N-1;P=7.
6. If Y, then move cursor down.
   If subject is marked, then GOTO 3.
   Else, add 2^P to IW(N); mark subject.
7. If N, move cursor down.
   If subject not marked, GOTO 3.
   Else, subtract 2^P from IW(N); clear mark.
8. Other inputs invalid: GOTO 3.

or clear a bit that isn't set. The table shows a routine for inputting subjects by bitmapping.

Particularly sticky situations can always be handled with a table. An array with the existing value for each value of the input is one way of doing this: A(N) contains the value used for N, the input value.

Editing The Files

By all means, make it easy to display a record entered some time ago, edit the display, and write the newly changed data in place of the original record. If you use subroutines for inputting each kind of data, this is easy to program.

For example, I have a subroutine that takes as input an author's name, then when it's acknowledged to be correct, writes it in the correct place on record "n" and also puts a corrected entry in the author index file in the right place. The record "n" may be an old one or the one we are writing for the first time. All you need to do is branch to such routines as one of the options given on a menu at the top of the program. Some errors will inevitably get by in your initial input. You need a way to correct errors both at the input and later as well.

Next issue we will outline the main program and talk about other techniques. ©

Driving Commodore is getting easier

Steer your way to greater efficiency and meet all your administrative needs of the future with the SUPER range of quality software.

Whether you want a fast and flexible database, a comprehensive and easy to use word processor, or a completely integrated office 'manager', Precision Software has the answer. With software that harnesses the full power of the new Commodore "B" Business Computer (known as the 700 in Europe).

Superbase Database manager for all business environments where accurate and up-to-date information is the key to success.

SUPERBASE sets new standards in flexibility and ease of use, with large record sizes, fully re-definable multi-screen record formats, spreadsheet-like calculation facilities, fast 'B+ tree' keyaccess with selective retrieval, transaction linking, sorting and fully definable report formats. SUPERBASE can also be linked to SUPERSCRIPT II.

Superscript II Enhanced version of the popular full feature word processor. Unrivalled table handling with 240 column wide screen, scrolling in all directions, arithmetic, memory calculator, column manipulation and mail merge with record selection. Comprehensive editing and finish quality output formatting. Handles up to 2,100 lines of 80 column text. And it spells!

Versatile, fast, simple to learn and use, SUPERSCRIPT II processes letters, quotations, reports, mailshots and standard forms with professional ease, enhancing presentation and ensuring perfect copies every time.

Superspell Fast, effective spelling checker ensures 100% typographic accuracy. An integral part of SUPERSCRIPT II, SUPERSPELL checks documents against a standard 30,000+ word dictionary and displays a list of every unrecognised word. SUPERSPELL handles both English and American spelling variants. You can accept, ignore, correct or add new words, building up your own dictionary extension. SUPERSPELL also includes a word look- up facility.

Superoffice The ultimate integrated office administration system, with full records management, calculation, word processing and spell checking.

SUPEROFFICE combines the database handling of SUPERBASE with the document editing and formatting power of SUPERSCRIPT II.

The availability of SUPERSCRIPT II as an integral word processor enables lists selected from data files to be used for mailshots, tables, standard forms and labels. SUPEROFFICE includes a powerful programming capability, enabling you to build up a library of your own programs. System-wide help screens are supported by clear comprehensive manuals and tutorials.

Choose an off-the-shelf application to suit your business from a steadily expanding library.

Find out more about the Superseries range of Software. Contact your local Commodore Dealer or Precision Software at our U.K. office.

OFFICIALLY APPROVED BY COMMODORE

Precision Software

Precision Software Limited
Park House, 4 Park Terrace,
Worcester Park, Surrey, KT4 7JZ.
Telephone: 01-330 7166
Telex: 8955021. PRECIS G

Mixing Graphics Modes On The 64

Part II

Sheldon Leemon

The two programs in the first part of this article (last month's COMPUTE!) showed you how to have different graphics modes simultaneously on the 64 screen. To conclude this discussion of mixing modes, here is a machine language program which uses a mixed graphics mode display to demonstrate the raster interrupt.

The interrupt uses a table of values that are POKEd into four key locations during each of the three interrupts, as well as values to determine at what scan lines the interrupts will occur. The locations affected are Control Register 1, Control Register 2, the Memory Control Register, and Background Color 0.

Control Register 1 (at location 53265) allows the selection of extended background color text mode, bitmap mode, screen blanking, and 24 or 25 rows of text. Control Register 2, at 53270, controls the selection of multicolor mode, and of a 38- or 40-column display. The Memory Control Register (53272) allows you to select which portion of VIC memory will be used for the video display (screen memory), and which for the data that defines the shape of text characters. Background Color Register 0 (53281) controls the background color in text mode. More detailed information about the bit assignments of these locations can be found in Appendix O of the *Commodore 64 User's Guide* and in the *Programmer's Reference Guide*.

The data for the interrupt routine is contained in lines 49152-49276. Each of these line numbers corresponds to the location where the first data byte in the statement is POKEd into memory. If you look at lines 49264-49276 of the BASIC program, you will see REMark statements that explain which VIC-II registers are affected by the DATA statements in each line. The numbers in these DATA statements appear in the reverse order in which they are put into the VIC register. For example, line 49273 holds the data that will go into Control Register 2. The last number, 8, is the one that will be placed into Control Register 2 while the top part of the screen is displayed. The first number, 24, is placed into Control Register 2 during the bottom part of the screen display and changes that portion of the display to multicolor mode.

The only tricky part in determining which data byte affects which interrupt comes in line 49264, which holds the data that determines the scan line at which each interrupt will occur. Each DATA statement entry reflects the scan line at which the *next* interrupt will occur. The first item in line 49264 is 49. Even though this is the entry for the third interrupt, this number corresponds to the top of the screen (only scan lines 50-249 are visible on the display). That is because after the third interrupt, the next to be generated is the first interrupt, which occurs at the top of the screen. Likewise, the last data item of 129 is used during the first interrupt to start the next one at scan line 129, in the middle of the screen. Try experimenting with these values to see what results you come up with. For example, if you change the number 169 to 209, you will increase the text area by five lines (40 scan lines).

Changing Effects

By changing the values in the data tables, you can alter the effect of each interrupt. Change the 20 in line 340 to 22, for example, and you will get lowercase text in the middle of the screen. Change the first 8 in line 49276 to 24, and you will get multicolor text in the center window. Each of these table items may be used in exactly the same way that you would use the corresponding register, in order to change background color, to obtain text or bitmap graphics, regular or multicolor modes, screen blanking, or extended background color mode.

```
0000                ; VIC CHIP EQUATES
0000                ;
0000                SCROLY = $D011        ;CONTROL REGISTER 1
0000                RASTER = $D012        ;RASTER LOCATION
0000                SCROLX = $D016        ;CONTROL REGISTER 2
0000                VMCSB  = $D018        ;V/C BASE ADDRESS
0000                VICIRQ = $D019        ;LATCHES ON IRQ FROM VIC
0000                IRQMSK = $D01A        ;VIC IRQ ENABLE
0000                BGCOL0 = $D021        ;BACKGROUND COLOR 0
0000                CIAICR = $DC0D        ;INTERRUPT CONTROL
0000                ;
0000                INTNO  = $FB          ;INTERRUPT COUNTER
0000                ;
0000                       * = $C000
C000                ;
C000 78             SETIRQ SEI            ;DISABLE ALL INTERRUPTS
C001 A9 7F                 LDA #$7F
C003 8D 0D DC              STA CIAICR     ;DISABLE CIA INTERRUPTS
C006 A9 01                 LDA #01
C008 8D 1A D0              STA IRQMSK     ;ENABLE RASTER IRQ
C00B A9 03                 LDA #03
C00D 85 FB                 STA INTNO      ;INITIALIZE INTERRUT NO.
C00F AD 70 C0              LDA RASTBL
C012 8D 12 D0              STA RASTER     ;SET SCAN LINE OF TOP RIRQ
C015 A9 18                 LDA #24
C017 8D 11 D0              STA $D011      ;SET HIGH BIT OF RIRQ SCAN LINE
C01A                ;
C01A AD 14 03              LDA $314       ;SAVE OLD IRQ VECTOR AND
C01D 8D 6E C0              STA OLDIRQ+1   ;MODIFY OLDIRQ TARGET ADDRESS TO
C020 AD 15 03              LDA $315       ;INSURE AGAINST CHANGE IN ADDRESS
C023 8D 6F C0              STA OLDIRQ+2   ;OF NORMAL INTERRUPT ROUTINE
C026                ;
C026 A9 32                 LDA #<RASIRQ   ;SET IRQ VECTOR
C028 8D 14 03              STA $314       ;TO USER ROUTINE
C02B A9 C0                 LDA #>RASIRQ
C02D 8D 15 03              STA $315
C030 58                    CLI            ;RE-ENABLE INTERRUPTS
C031 60                    RTS
C032                ;
C032                ;
C032 AD 19 D0       RASIRQ LDA VICIRQ
C035 8D 19 D0              STA VICIRQ     ;CLEAR VIC INTERRUPTS
C038 29 01                 AND #01        ;IS RASTER THE SOURCE OF IRQ?
C03A F0 2B                 BEQ INTRT      ;NO, EXIT
C03C C6 FB                 DEC INTNO      ;NEXT INTERRUPT
C03E 10 04                 BPL RAS1       ;NOT LAST INTERRUPT
C040 A9 02                 LDA #2         ;LAST INTERRUPT, RESET COUNTER
C042 85 FB                 STA INTNO
C044                ;
C044 A6 FB          RAS1   LDX INTNO
C046 BD 73 C0              LDA COLTBL,X   ;SET BACKGROUND COLOR
C049 8D 21 D0              STA BGCOL0
C04C BD 76 C0              LDA CR1TBL,X   ;SET CONTROL REG 1
C04F 8D 11 D0              STA SCROLY
C052 BD 79 C0              LDA CR2TBL,X   ;SET CONTROL REG 2
C055 8D 16 D0              STA SCROLX
C058 BD 7C C0              LDA MEMTBL,X   ;SET MEMORY CONTROL
C05B 8D 18 D0              STA VMCSB
C05E BD 70 C0              LDA RASTBL,X   ;RESET INTERRUPT SCAN LINE
C061 8D 12 D0              STA RASTER
C064 8A                    TXA            ;LAST INTERRUPT EXITS
C065 F0 06                 BEQ OLDIRQ     ;THROUGH OLD VECTOR
C067                ;
C067 68             INTRT  PLA            ;RESTORE STACK
C068 A8                    TAY
```

```
C069 68                 PLA
C06A AA                 TAX
C06B 68                 PLA
C06C 40                 RTI
C06D            ;
C06D 4C 31 EA OLDIRQ JMP $EA31        ;OLD IRQ--ADDRESS MODIFIED ABOVE
C070            ;
C070 31         RASTBL .BYT 49,170,129;SCAN LINE OF NEXT INTERRUPT
C071 AA
C072 81
C073 00         COLTBL .BYT 0,6,0     ;BACKGROUND COLORS
C074 06
C075 00
C076 3B         CR1TBL .BYT 59,27,59  ;CONTROL REGISTER 1 VALUES
C077 1B
C078 3B
C079 18         CR2TBL .BYT 24,8,8    ;CONTROL REGISTER 2 VALUES
C07A 08
C07B 08
C07C 18         MEMTBL .BYT 24,20,24  ;MEMORY CONTROL REGISTER VALUES
C07D 14
C07E 18
C07F
C07F                    .END
```

©

C. Regena

Subscripted Variables

TI BASIC allows variable names to be *subscripted*, or used in arrays of up to three dimensions. Examples of subscripted variables are A(1), ING$(2,6), and N(7,2,8).

Both numeric and string variables may use subscripts, which are written as numbers in parentheses after the variable name. The subscript itself may be a numeric variable or numeric expression. One constraint is that you cannot use the same variable name both with and without a subscript; that is, you cannot use the variable N and the variable N(3).

Just Like Mailboxes

I often think of variables as a mailbox system in memory:

A	B

Here are two variables, named A and B. Initially, they each have the value of zero. As your program runs, you may assign values to these boxes. Suppose you have the statements:

```
100 B=7
150 A=A+1
```

The computer will put the value 7 in B's mailbox, then any later statement using B will simply use 7 in the formula instead of B. Line 150 says to add 1 to the value that is currently in A, then place the new value in A.

Some mailboxes are larger than others, and I compare these to subscripted variables. You might think of it as a big box for the Smith family – the first part of the box for John, the second part for James, and the third part for Jeremy. Here is our mailbox again:

A	B
C_1	C_2

The C box actually holds two values, which are written in TI BASIC as C(1) and C(2).

Boxes can be even larger – representing 1, 2, or 3 "dimensions," or using 1, 2, or 3 numbers in the subscripts. C(2) is the second element in the one-dimensional array of C above. N(2,4) would be an element in a two-dimensional array. X(3,4,2) would be an element in a three-dimensional array.

Arrays Are Workhorses

Arrays or subscripted variables can make a computer program more efficient in many cases. If you use a process several times, it may be worth using a variable with a subscript rather than several variables.

For example, suppose you are using your computer to sort a list of 25 students with their scores on a particular test. You could use the following method:

```
200 INPUT A$,A (FIRST STUDENT, SCORE)
210 INPUT B$,B (SECOND STUDENT, SCORE)
220 INPUT C$,C (THIRD STUDENT, SCORE)

ETC., FOR 25 STUDENTS

.
.
.
(SORT ROUTINE USING 25 VARIABLES)

.
.
.
600 PRINT A,A$
610 PRINT B,B$
620 PRINT C,C$

ETC., FOR 25 SORTED SCORES AND STUDENTS.
```

Using arrays or subscripted variables, you could INPUT the names as the N$ array and the corresponding scores in the SC array, sort, and then print using this method:

```
200 FOR C=1 TO 25
210 INPUT N$(C),SC(C)
220 NEXT C
(SORT ROUTINE)
600 FOR C=1 TO 25
610 PRINT SC(C),N$(C)
620 NEXT C
```

Here is another example program that would be considerably longer if you did not use subscripted variables. Lines 110-130 READ from DATA a subject, a verb, and a phrase and put them in the S$, V$, and P$ arrays. Lines 140-190 contain the data (you could combine data lines if you wish). For the first time through the loop, S$(1) would be "I", V$(1) would be "RAN", and P$(1) would be "TO OUR HOUSE." S$(2) is "HE", V$(2) is "WALKED", and P$(2) is "TO THE STORE."; and so forth.

Line 200 uses the DEF function to define R6 as a random integer from 1 to 6. Each time R6 is used in the program, the computer will choose a random number from 1 to 6.

Line 210 clears the screen, and line 220 prints a title. Lines 230-240 choose a random S$, a random V$, and a random P$ to make up a sentence and print it. Line 250 returns to line 230 to repeat the process until you press CLEAR.

```
100 REM RANDOM SENTENCES
110 FOR C=1 TO 6
120 READ S$(C),V$(C),P$(C)
130 NEXT C
140 DATA I,RAN,TO OUR HOUSE.
150 DATA HE,WALKED,TO THE STORE.
160 DATA SHE,HOPPED,AROUND THE ROOM.
170 DATA IT,SPED,UP THE HILL.
180 DATA WE,ZOOMED,ACROSS THE GRASS.
190 DATA YOU,JUMPED,ALONG THE PATH.
200 DEF R6=INT(6*RND)+1
210 CALL CLEAR
220 PRINT "** RANDOM SENTENCES **":::
230 RANDOMIZE
240 PRINT :S$(R6);" ";V$(R6);" ";P$(R6)
250 GOTO 230
260 END
```

Memory Reserved

As soon as you specify a variable name with a subscript, the computer automatically reserves memory for an array with that name. If you use a variable D(3), the computer will automatically reserve elements up to D(10). In two-dimensional arrays, the computer will reserve up to N(10,10); and in three-dimensional arrays, the computer will reserve up to X(10,10,10).

If you need more than ten elements, use a DIMension statement to clear enough space. For example, for our 25 students and 25 scores in the program discussed previously, we would need a DIMension statement:

```
100 DIM N$(25),SC(25)
```

If your program is running nearly full memory and you do not need all the elements automatically reserved, you may save memory by dimensioning the array for the exact number you need:

```
100 DIM N(6)
```

The DIMension statement must appear before any reference to the array. I usually put my DIMension statements near the beginning of the program. You may specify several variables in one DIMension statement.

The computer actually starts all subscripts with the zero element, N(0). Thus, the automatic dimensioning includes 11 elements in arrays. If you prefer to use only elements numbered 1 and above, you may use the OPTION BASE statement to avoid reserving space for the zero elements:

```
100 OPTION BASE 1
110 DIM D(25,6)
```

Note: The OPTION BASE 1 statement must precede the DIM statement.

Combining The Ingredients

Following is an educational program which illustrates the use of subscripted variables. The program prints a recipe conversion problem for a math competency test. First, one of three recipes is printed. A random ingredient is chosen, and a random multiplication factor is chosen to print the problem. The student must choose from four possible answers.

Line 140 DIMensions the R$ array and the R array so the first subscript may go up to 3 and the second subscript may go up to 6. The first subscript will actually be 1, 2, or 3, which will correspond to the first, second, or third recipe. R$(C,0) will contain the title of the recipe for each of the three recipes. R(C,0) will be the number of servings each of the three recipes will make. R(C,I) and R$(C,I) contain the amount and the ingredient, where C is the recipe number and I is from 1 to 6. The values are read in as DATA in lines 150-230.

Lines 410-440 define values for the elements of the J array. These elements are multiplication factors for the conversion problem. These variables are used first to choose a factor for the problem, then to calculate the multiple-choice answers.

Program Structure

Lines	
100-130	Print title screen.
140	DIMension arrays for recipe elements.
150-200	READ from DATA the values for the R$ and R arrays.
210-230	DATA for recipes (please be careful while copying these lines – watch the commas and decimals).
240	Branch around subroutines.
250-390	Subroutines to convert decimals to fractions for printing the recipes and the multiple-choice answers.
400	Clear screen for problem.
410-440	Define multiplication factors.

450-460 Randomly choose Recipe 1, Recipe 2, or Recipe 3. The variable C refers to the recipe number.
470-480 Print title of recipe and number of servings.
490-530 Print amount, measure, and ingredient six times. One of the recipes contains only five ingredients, so line 500 checks for a zero value. Line 510 converts the amount from a decimal to a fraction if necessary.
540-560 Randomly choose a multiplication factor for the problem. If F = 1 then J(1) = 1 which indicates no recipe conversion, and another number is chosen.
570-590 Draw a horizontal line of a random color under the given recipe.
600-640 Print the question, where A is the randomly chosen ingredient.
650 Calculate correct answer as N1.
660-750 Randomly print multiple-choice answers.
760-780 Sound a "beep" then wait for answer.
790-820 If answer is incorrect, play "uh-oh" and return for another answer.
830-870 Indicate correct answer and play arpeggio.
880-910 Print option to try another problem and branch appropriately.
920-930 Clear screen and END.

Math Competency Recipe Conversion

```
100 CALL CLEAR
110 PRINT TAB(6);"MATH COMPETENCY"
120 PRINT :::TAB(5);"RECIPE CONVERSION"
130 PRINT
140 DIM R$(3,6),R(3,6)
150 FOR C=1 TO 3
160 READ R$(C,0),R(C,0)
170 FOR I=1 TO 6
180 READ R(C,I),R$(C,I)
190 NEXT I
200 NEXT C
210 DATA CHEESE SOUFFLE,2,2,TBSP BUTTER,2,TBSP FLOUR,1,C. MILK,.75,C. GRATED CHEESE,2,EGGS,.5,TSP SALT
220 DATA DUMPLINGS,4,1,C. FLOUR,2,TSP BAKING POWDER,.5,TSP SALT,.5,C. MILK,2,TBSP SALAD OIL,0,""
230 DATA PRONTO PUPS,6,2,EGGS,.5,C. MILK,.75,C. FLOUR,1,TSP BAKING POWDER,1,TSP SALT,.5,C. CORN MEAL
240 GOTO 400
250 N=R(C,I)
260 IF N<1 THEN 290
270 N$=STR$(N)
280 RETURN
290 IF N<>.75 THEN 320
300 N$="3/4"
310 RETURN
320 IF N<>.5 THEN 350
330 N$="1/2"
340 RETURN
350 IF N<>.375 THEN 380
360 N$="3/8"
370 RETURN
380 N$="1/4"
390 RETURN
400 CALL CLEAR
410 J(0)=.5
420 J(1)=1
430 J(2)=2
440 J(3)=4
450 RANDOMIZE
460 C=INT(RND*3)+1
470 PRINT TAB(7);R$(C,0)
480 PRINT :"SERVES";R(C,0)::
490 FOR I=1 TO 6
500 IF R(C,I)=0 THEN 530
510 GOSUB 250
520 PRINT N$;TAB(5);R$(C,I)
530 NEXT I
540 F=INT(RND*4)
550 IF F=1 THEN 540
560 F=J(F)
570 H=INT(RND*12)+5
580 CALL COLOR(13,H,H)
590 CALL HCHAR(24,1,128,32)
600 PRINT :::"IF YOU WANTED TO MAKE "
610 PRINT R$(C,0);" TO SERVE";F*R(C,0)
620 A=INT(RND*5)+1
630 PRINT "HOW MANY ";R$(C,A)
640 PRINT "WOULD YOU NEED?"::
650 N1=F*R(C,A)
660 FOR CH=1 TO 4
670 X=INT(RND*4)
680 IF J(X)=-1 THEN 670
690 N=J(X)*R(C,A)
700 IF N1<>N THEN 720
710 ANS=CH
720 GOSUB 260
730 PRINT TAB(6);CHR$(64+CH);"  "&N$
740 J(X)=-1
750 NEXT CH
760 CALL SOUND(150,1497,2)
770 CALL KEY(0,K,S)
780 IF S<1 THEN 770
790 IF K=ANS+64 THEN 830
800 CALL SOUND(100,330,2)
810 CALL SOUND(100,262,2)
820 GOTO 770
830 CALL HCHAR(19+ANS,7,42)
840 CALL SOUND(100,262,2)
850 CALL SOUND(100,330,2)
860 CALL SOUND(100,392,2)
870 CALL SOUND(200,523,2)
880 PRINT :"ANOTHER PROBLEM? (Y/N)";
890 CALL KEY(0,K,S)
900 IF K=89 THEN 400
910 IF K<>78 THEN 890
920 CALL CLEAR
930 END
```

©

Jim Butterfield, Associate Editor

Bagel Break, Part 2

Last month we outlined the logic of a simple machine language program to play "Bagels," a well-known guessing game. Let's pause and look at the various ways we can change our planned program into a real machine language program.

You may have a tiny assembler that is built into your monitor system. This type of simple assembler is often called a *nonsymbolic* assembler for reasons we'll discuss in a moment. If so, you'll work out all the addresses yourself and write them in as you jot down the program coding. The type of outline you write will be similar to that in Program 2. You'll need to guess at some of the "forward" branches; at the time you write the branch instruction, you won't know what the exact destination address will be. No matter, as long as you remember to put the correct addresses in later.

You may have purchased a full-scale assembler, in which case you'll write the program as shown in Program 1. It's still the same logic flow, but now we can give a name (or "symbolic address") to the various parts of the program. We'll let the assembler figure out where these locations are and compute the correct branch for us. This type of assembler, where we can name locations with symbolic names, or "labels," is often called a "symbolic assembler" to distinguish it from the simple assemblers mentioned previously.

Symbolic names, or labels, seem like a convenience feature at first: not too important, but handy. In fact, they change the nature of the work in a couple of ways. First, we now have the freedom to give meaningful names to our program and work locations. The program is easier to read. Second – and this can be very powerful – we can move the logic to an entirely new part of memory with very little work; the assembler will figure everything out for us. Perhaps most important of all: if we wish to change or correct the program, we can do so without needing to type everything in again; the "source" coding will be saved on a file and may be recalled and corrected as desired.

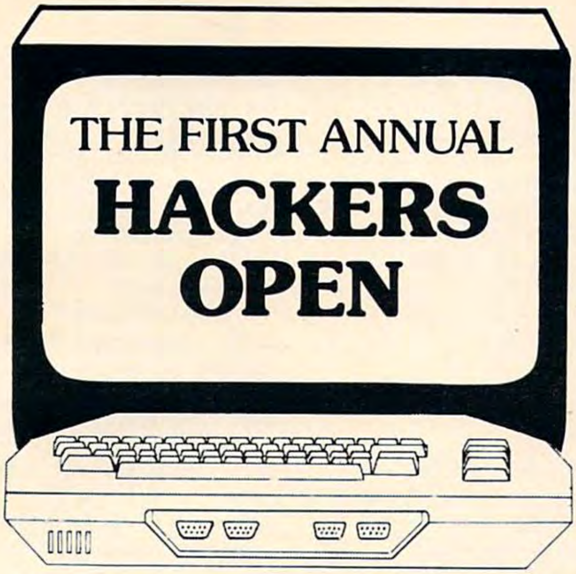
In whatever fashion we write our program outline, we'll still need to change it into machine language. We may use an assembler – symbolic or nonsymbolic – or we might do the job by hand. Program 3 shows the output from a typical assembly. It's full of information, but the only data that really count are the two-digit hexadecimal numbers found to the left of the printout. (The four-digit hex numbers at the extreme left are addresses, to help you know where the code is located.)

An assembly listing is a rich source of information, especially if it's well commented. But the business end – the two-digit hex numbers – is all that is needed to do the job. Those numbers are all that we need to put into the computer. Program 4 shows a hexadecimal dump of memory with the program in place. All the pretty trimmings from the assembly listing are gone. All that we have are the instructions, ready to go to work. That's probably the way we would type them in, using the screen-editing feature of the machine language monitor to change memory until it looked like Program 4.

But our game isn't completed yet. We need to generate the mystery numbers from BASIC, and tie all the pieces together. Next time....

Program 1: Code As Prepared For A Full Assembler

```
NGUESS =$0240        ;number of guesses
EXACT  =$0241        ;exact count
MATCH  =$0242        ;other match count
INCHAR =$0243        ;input character count
SECRET =$0244        ;secret code
SCOPY  =$0248        ;copy of secret code
UGUESS =$024C        ;user's guess
       *= $033C      ;start program here
       ; start game
START  LDA #$00        ; guesses to zero
       STA NGUESS
       ; accept next guess
GUESS  INC NGUESS      ; count the guess
       LDA NGUESS      ; look at it
       CMP #10         ; over nine?
       BEQ QUIT        ; yup, quit
       JSR PLAY        ; take guess
       BNE GUESS       ; not finished?  back
```



★ 1st PRIZE ★
$1000.00
and Royalties

★ 2nd PRIZE ★
$500.00
and Royalties

★ 3rd PRIZE ★
—UNLIMITED—
$100.00 and 1 year
Magnetic Magazine
subscription

Do something with your Programs! Send them to us and receive $$s, royalties, directions, suggestions, and marketing assistance. Winning programs will be marketed individually or as a part of our Magnetic Magazines. If you have an Atari 400/800, Commodore 64, VIC-20, TRS-80 Color, IBM PC, Apple II — you could be a contestant.

For complete information write or call toll-free:

programmer's institute

a division of FUTUREHOUSE, INC.
p.o. box 3470, chapel hill, north carolina 27514, 919-967-0861

1-800-334-SOFT

For All Walks of Life

MIRAGE CONCEPTS offers you a step in the right direction by presenting the All New DATABASE MANAGER and ... WORD PROCESSOR programs, the most powerful and easy to use software available for the Commodore 64.

DATABASE MANAGER
The Most Powerful Database Management System
- 100% Machine Language
- Free Form Design and Input
- Sort on Any Field/Any Level
- Calculated Fields
- Max. Record size = 2,000 Characters

WORD PROCESSOR
80 Column Screen Display Without Additional Hardware
- 100% Machine Language
- Over 70 Single Keystroke Commands
- Printed Page/Line/Character Counters
- True Word Wrap
- Search, Replace and Block Operations

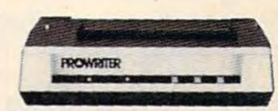
FOR THE COMMODORE 64

MIRAGE CONCEPTS, INC.
"A Step Ahead!"

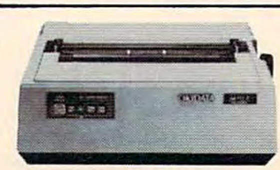
2519 W. Shaw, Suite 106 / Fresno, CA 93711 / Customer Support: (209) 227-8369
Order Number: (800) 641-1441 Order Number (in Calif.): (800) 641-1442

Large Selection – BIG Savings

Radio Shack DMP200 $599



CITOH Prowriter $375
CITOH Prowriter II $710



Okidata 92 $510
Okidata 82A $399

Smith Corona TPI Daisy Wheel $495

PRINTERS

Smith Corona TPI	$495
Silver Reed EXP 550 D.W.	665
Epson	Call
Okidata 82A	399
Okidata 92	510
Okidata 93	859
Okidata 83	655
Okidata 84	999
CITOH Prowriter	375
Star Gemini 10	319
Radio Shack DWII	1715
Radio Shack DWP410	1320
Radio Shack DMP120	410
Radio Shack DMP200	599
Radio Shack DMP2100	1779
Radio Shack CGP115	199

Full complement of Radio Shack Computers: Hardware & Software

Color Computer 16K w/extended basic	255
Color Computer 32-64K w/extended basic	345
Model IV 16K	849
Model IV 64K 2 drives & RS232	1699
Model 100 8K	679
Model 100 24K	835

TOLL FREE **1-800-343-8124**

Write for FREE catalog

computer plus

P.O. Box 926 • 480 King St. • Littleton, MA 01460

```
QUIT   RTS              ; end of game
       ; get guess & play
PLAY   ORA #$3Ø         ;ascii numeric
       JSR $FFD2        ; .. print
       LDA #$2Ø         ;ascii space
       JSR $FFD2        ; print it
       ; set counts to zero
       LDX #Ø
       STX EXACT
       STX MATCH
       STX INCHAR
       ; get 4 character guess
INLOOP JSR $FFE4        ; get char
       CMP #$41         ; less than A
       BCC INLOOP
       CMP #$47         ; over F
       BCS INLOOP       ; reject it
       JSR $FFD2        ; OK, print it
       LDX INCHAR       ; get position
       INC INCHAR       ; bump position
       STA UGUESS,X     ; store character
       LDA SECRET,X     ; copy secret char
       STA SCOPY,X      ; .. to copy area
       CPX #3           ; four chars?
       BNE INLOOP       ; nope, go back
       ; check guess for exact matches
COMPAR LDA SCOPY,X      ; test character
       CMP UGUESS,X     ; against guess
       BNE SKIP         ; nope, try next
       INC EXACT        ; yes, count it
       LDA #Ø           ; and wipe out..
       STA SCOPY,X      ; .. matching ..
       STA UGUESS,X     ; .. characters
SKIP   DEX
       BPL COMPAR
       ; check for out-of-place matches
       LDY #$ØØ         ; first secret char
RETRY  LDX #$ØØ         ; first guessed char
CHECK  LDA SCOPY,Y      ; is character wiped?
       BEQ PASS         ; yes, ignore next bit

       CMP UGUESS,X     ; compare it to guess
       BNE PASS         ; nope, move on
       INC MATCH        ; yup, count it
       LDA #$ØØ         ; and wipe out..
       STA SCOPY,Y      ; .. matching ..
       STA UGUESS,X     ; .. characters
PASS   INX              ; next guessed character
       CPX #4           ; tried them all?
       BCC CHECK        ; no, try next one
       INY              ; next secret character
       CPY #4           ; tried them all?
       BCC RETRY        ; no, keep going
       ; print results
       LDX #Ø           ; start at 'exact'
PLOOP  LDA #$2Ø         ; print a space
       JSR $FFD2
       LDA EXACT,X      ; get the number
       ORA #$3Ø         ; to ascii numeric
       JSR $FFD2        ; and print
       INX              ; move to 'match'
       CPX #$Ø2         ; too far?
       BCC PLOOP        ; nope, keep printing
       LDA #$ØD         ; print 'return'
       JSR $FFD2
       LDA EXACT        ; four exact?
       CMP #4           ; if so, set z flag
       RTS
```

Program 2: Code As Prepared For A Tiny Assembler

```
(Ø33C)  LDA #$ØØ
        STA $Ø24Ø
(Ø341)  INC $Ø24Ø
        LDA $Ø24Ø
        CMP #1Ø
        BEQ $Ø35Ø
        JSR $Ø351
        BNE $Ø341
(Ø35Ø)  RTS
(Ø351)  ORA #$3Ø
        JSR $FFD2
        LDA #$2Ø
        JSR $FFD2
        LDX #$ØØ
        STX $Ø241
        STX $Ø242
        STX $Ø243
(Ø366)  JSR $FFE4
        CMP #$41
        BCC $Ø366
        CMP #$47
        BCS $Ø366
        JSR $FFD2
        LDX $Ø243
```

Program 3: Code As Assembled By A Full Assembler

```
                   NGUESS =$Ø24Ø     ;number of guesses
                   EXACT  =$Ø241     ;exact count
                   MATCH  =$Ø242     ;other match count
                   INCHAR =$Ø243     ;input char count
                   SECRET =$Ø244     ;secret code
                   SCOPY  =$Ø248     ;copy secret code
                   UGUESS =$Ø24C     ;user's guess
                          *= $Ø33C   ;start program here
                                       ; start game
Ø33C A9 ØØ     START   LDA #$ØØ        ; guesses, zero
Ø33E 8D 4Ø Ø2          STA NGUESS
                                       ; accept next guess
Ø341 EE 4Ø Ø2  GUESS   INC NGUESS      ; count guess
Ø344 AD 4Ø Ø2          LDA NGUESS      ; get it
Ø347 C9 ØA             CMP #1Ø         ; over nine?
Ø349 FØ Ø5             BEQ QUIT        ; yup, quit
Ø34B 2Ø 51 Ø3          JSR PLAY        ; take guess
Ø34E DØ F1             BNE GUESS       ; not finished?
Ø35Ø 6Ø        QUIT    RTS             ; end of game
                                       ; get guess & play
Ø351 Ø9 3Ø     PLAY    ORA #$3Ø        ;ascii numeric
Ø353 2Ø D2 FF          JSR $FFD2       ; .. print
Ø356 A9 2Ø             LDA #$2Ø        ;ascii space
Ø358 2Ø D2 FF          JSR $FFD2       ; print it
                                       ; set counts to zero
Ø35B A2 ØØ             LDX #Ø
Ø35D 8E 41 Ø2          STX EXACT
Ø36Ø 8E 42 Ø2          STX MATCH
Ø363 8E 43 Ø2          STX INCHAR      ; get 4 character
                                         guess
```

SAVE

commodore VIC 64 $229.00

VIC 20 SPECIAL $89.00 INCLUDES GORTEK & MICROCHIPS VL110 Cassette program (retail value $19.95).

Commodore VIC20/64 1541 Single Disk Drive	$239.00
Commodore VIC20/64 1530 Datassette	65.00
Commodore VIC20/64 1525 Printer	229.00
Commodore VIC20/64 1701 Color Monitor	249.00
Commodore VIC20/64 1600 Vicmoden (Telephone Interface)	55.00

Everything you need to support your COMMODORE VIC20 AND VIC64

Unbelievable prices for game and business software for your VIC20 and VIC64.

.... *TOO MANY TO LIST BUT WE GOT IT* .....

Micro Software International Inc

PractiCalc 20
For the Commodore VIC-20— 16K RAM required

Tape (CV2T10)	$28.00
Disk (CV2D10)	33.00

PractiCalc 64 For the Commodore 64

Tape (C60T11)	$37.00
Disk (C60D11)	39.00

Cardco, Creative Software, EPYX, HES, UMI, WICO, Romox.

SAVE

Texas Instruments TI-99/4A Home Computer

ONLY $109.00*

*After TI manufacturer's rebate of $50.00.

MORE GOOD DEALS

PHA 4100 Color Monitor	$349.95
PHP 2400 TI Impact Printer	540.95
PHP 1100 Wired Remote Controllers	27.95
PHP 2700 Program Recorder	49.95

FREE EXPANSION BOX ($183. VALUE) with the purchase of any three of the following low price items:

PHP 1220 RS 232 Card	$135.95
PHP 1240 Disk Controller Card	189.95
PHP 1250 Expansion System Disk Drive	300.95
PHP 1260 Memory Expansion Card 32K	227.95
PHP 1270 P-Code Card	189.95
PHM 3111 TI Writer	75.95
PHM 3113 Microsoft Multiplan	75.95

FREE SPEECH SYNTHESIZER when you buy six solid state software, command cartridges.

QUANTITIES ARE LIMITED!

We carry in stock all hardware and software for TI home computers.

PHM 3026 Extended Basic	$ 75.95
PHM 3035 Terminal Emulator II	38.95
PHM 3058 Mini-Memory	75.95
PHM 3055 Editor/Assembler	37.95
PHM 3109 TI Logo II	74.95
PHM 3112 Parsec	30.95
PHM 3053 TI Invaders	44.95
PHM 3041-T Adventure (Pirate Adv.)	30.95
PHM 3057 Munch Man	30.95
PHM 3006 Home Financial Decisions	23.95
PHM 3044 Personal Report Generator	30.95
PHM 3002 Early Learning Fun	22.95
PHM 3090-97 Miliken Home Math Series K-8th grade	30.95
PHM 3059-62 Scholastic Spelling Level 3-6	41.95
PHM 3046-48 Scott, Foresman, Reading Pkgs.	41.95
PHM 3064 Touch Typing Tutor	30.95
PHM 7008 Speaking Scholastic Spelling	165.95
PHM 3122 Plato Interpreter	37.95
PHM 3083-88 Addison Wesley Computer Math Games I-VI	30.95

SAVE

SOUTHERN AUDIO VIDEO ELECTRONICS, INC.
1782 Marietta Blvd., N.W., Atlanta, Georgia 30318

Order Toll Free 1-800-241-2682
In Georgia (404)-351-8459

Get the best prices on TI Hardware and software. For a complete listing of all SAVE's products, send $5.00 for our catalogue (refundable with your first order).

Enjoy the convenience of in-home shopping. Call our toll free number today for orders only.

Use your American Express, VISA, Mastercard, check or money order. Minimum order of $50. Outside continental U.S.A. minimum order $250 U.S. currency. Shipping and handling charges are extra. All prices are subject to change without notice. Allow 2-4 weeks for delivery. Prices good through November, 1983. *$50.00 Rebate expires January, 1984.

```
          INC $Ø243
          STA $Ø24C,X
          LDA $Ø244,X
          STA $Ø248,X
          CPX #$Ø3
          BNE $Ø366
(Ø387)    LDA $Ø248,X
          CMP $Ø24C,X
          BNE $Ø394
          INC $Ø241
          LDA #$ØØ
          STA $Ø248,X
          STA $Ø24C,X
(Ø39A)    DEX
          BPL $Ø381
          LDY #$ØØ
(Ø39F)    LDX #$ØØ
(Ø3A1)    LDA $Ø248,Y
          BEQ $Ø3BØ
          CMP $Ø24C,X
          BNE $Ø3BØ
          INC $Ø242
          LDA #$ØØ
          STA $Ø248,Y
          STA $Ø24C,X
(Ø3B6)    INX
          CPX #$Ø4
          BCC $Ø39B
          INY
          CPY #$Ø4
          BCC $Ø399
          LDX #$ØØ
(Ø3C2)    LDA #$2Ø
          JSR $FFD2
          LDA $Ø241,X
          ORA #$3Ø
          JSR $FFD2
          INX
          CPX #$Ø2
          BCC $Ø3BC
          LDA #$ØD
          JSR $FFD2
          LDA $Ø241
          CMP #$Ø4
          RTS
```



```
Ø366 2Ø E4 FF INLOOP JSR $FFE4      ; get char
Ø369 C9 41           CMP #$41       ; less than A
Ø36B 9Ø F9           BCC INLOOP
Ø36D C9 47           CMP #$47       ; over F
Ø36F BØ F5           BCS INLOOP     ; reject it
Ø371 2Ø D2 FF        JSR $FFD2      ; OK, print it
Ø374 AE 43 Ø2        LDX INCHAR     ; get position
Ø377 EE 43 Ø2        INC INCHAR     ; bump position
Ø37A 9D 4C Ø2        STA UGUESS,X   ; store
                                      character
Ø37D BD 44 Ø2        LDA SECRET,X   ; copy secret
                                      ch
Ø38Ø 9D 48 Ø2        STA SCOPY,X    ; .. to copy
area
Ø383 EØ Ø3           CPX #3         ; four chars?
Ø385 DØ DF           BNE INLOOP     ; nope, go back
                                    ; check guess for
                                      exact matches
Ø387 BD 48 Ø2 COMPAR LDA SCOPY,X    ; test
                                      character
Ø38A DD 4C Ø2        CMP UGUESS,X   ; against guess
Ø38D DØ ØB           BNE SKIP       ; nope, try
                                      next
Ø38F EE 41 Ø2        INC EXACT      ; yes, count it
Ø392 A9 ØØ           LDA #Ø         ; and wipe
                                      out..
Ø394 9D 48 Ø2        STA SCOPY,X    ; .. matching
                                      ..
Ø397 9D 4C Ø2        STA UGUESS,X   ; .. characters
Ø39A CA       SKIP   DEX
Ø39B 1Ø EA           BPL COMPAR
                                    ; check for matches
Ø39D AØ ØØ           LDY #$ØØ       ; first secret
Ø39F A2 ØØ    RETRY  LDX #$ØØ       ; first guessed
Ø3A1 B9 48 Ø2 CHECK  LDA SCOPY,Y    ; char wiped?
Ø3A4 FØ 1Ø           BEQ PASS       ; yes, ignore
Ø3A6 DD 4C Ø2        CMP UGUESS,X   ; compare
Ø3A9 DØ ØB           BNE PASS       ; nope, move on
Ø3AB EE 42 Ø2        INC MATCH      ; yup, count it
Ø3AE A9 ØØ           LDA #$ØØ       ; and wipe
                                      out..
Ø3BØ 99 48 Ø2        STA SCOPY,Y    ; .. matching
                                      ..
Ø3B3 9D 4C Ø2        STA UGUESS,X   ; .. characters
Ø3B6 E8       PASS   INX            ; next guess
Ø3B7 EØ Ø4           CPX #4         ; tried all?
Ø3B9 9Ø E6           BCC CHECK      ; no, try next
Ø3BB C8              INY            ; next char
Ø3BC CØ Ø4           CPY #4         ; tried all?
Ø3BE 9Ø DF           BCC RETRY      ; no, keep on
                                    ; print results
Ø3CØ A2 ØØ           LDX #Ø         ; first numbr
Ø3C2 A9 2Ø    PLOOP  LDA #$2Ø       ; print space
Ø3C4 2Ø D2 FF        JSR $FFD2
Ø3C7 BD 41 Ø2        LDA EXACT,X    ; get number
Ø3CA Ø9 3Ø           ORA #$3Ø       ; to ascii num
Ø3CC 2Ø D2 FF        JSR $FFD2      ; and print
Ø3CF E8              INX            ; move on
Ø3DØ EØ Ø2           CPX #$Ø2       ; too far?
Ø3D2 9Ø EE           BCC PLOOP      ; nope, loop
Ø3D4 A9 ØD           LDA #$ØD       ; print return
Ø3D6 2Ø D2 FF        JSR $FFD2
Ø3D9 AD 41 Ø2        LDA EXACT      ; four exact?
Ø3DB C9 Ø4           CMP #4         ;   z flag
Ø3DD 6Ø              RTS
```

Program 4: Hexadecimal Dump Of Memory

```
C*
     PC   IRQ   SR AC XR YR SP
.;  B78Ø E455  2C 34 3A 9D F8
.
.:  Ø33C A9 ØØ 8D 4Ø Ø2 EE 4Ø Ø2
.:  Ø344 AD 4Ø Ø2 C9 ØA FØ Ø5 2Ø
.:  Ø34C 51 Ø3 DØ F1 6Ø Ø9 3Ø 2Ø
.:  Ø354 D2 FF A9 2Ø 2Ø D2 FF A2
.:  Ø35C ØØ 8E 41 Ø2 8E 42 Ø2 8E
.:  Ø364 43 Ø2 2Ø E4 FF C9 41 9Ø
.:  Ø36C F9 C9 47 BØ F5 2Ø D2 FF
.:  Ø374 AE 43 Ø2 EE 43 Ø2 9D 4C
.:  Ø37C Ø2 BD 44 Ø2 9D 48 Ø2 EØ
.:  Ø384 Ø3 DØ DF BD 48 Ø2 DD 4C
.:  Ø38C Ø2 DØ ØB EE 41 Ø2 A9 ØØ
.:  Ø394 9D 48 Ø2 9D 4C Ø2 CA 1Ø
.:  Ø39C EA AØ ØØ A2 ØØ B9 48 Ø2
.:  Ø3A4 FØ 1Ø DD 4C Ø2 DØ ØB EE
.:  Ø3AC 42 Ø2 A9 ØØ 99 48 Ø2 9D
.:  Ø3B4 4C Ø2 E8 EØ Ø4 9Ø E6 C8
.:  Ø3BC CØ Ø4 9Ø DF A2 ØØ A9 2Ø
.:  Ø3C4 2Ø D2 FF BD 41 Ø2 Ø9 3Ø
.:  Ø3CC 2Ø D2 FF E8 EØ Ø2 9Ø EE
.:  Ø3D4 A9 ØD 2Ø D2 FF AD 41 Ø2
.:  Ø3DC C9 Ø4 6Ø
.
```

©

ISAM

Building Your Own Random File Manager

Michael D. Lipay

There are several approaches to handling computer files (collections of data). Among the fastest and best is the random access disk file which uses special techniques to quickly locate any piece of information from anywhere within the entire file.

This tutorial explains how random access can be achieved and examines alternative ways to process data files. It includes a sample program, written in Applesoft BASIC, but which can easily be adapted to work on other computers using Microsoft BASIC.

Besides protecting earth from aliens, a main purpose of a computer is processing information. This data processing can be anything from keeping track of your stamp collection to maintaining a running inventory for your business. When it becomes necessary to retain the information long after the computer has been turned off, tape or disk storage is used.

Magnetic storage devices are capable of storing information indefinitely (provided they are kept clean and away from magnetic fields). Basically, there are two types of magnetic storage devices available to the micro computer user – tape and disk. Both devices are capable of storing large amounts of information, and do so in groups called files. A file is a collection of related information, and the user has three primary types of files to select from:

I) Sequential Tape Files
II) Sequential Disk Files
III) Random Access Disk Files

Which of the three you decide to use for a given program will depend on many factors. Each has its own advantages and disadvantages; they are discussed here in an effort to help you select the best one for your needs.

Sequential Tape Files

If you have large amounts of data which you do not need to process frequently, then tape files should be considered. Tapes can store vast amounts of data in a relatively compact space, and at a very low price. Tapes serve as an excellent medium to keep a backup of disk programs and files. The big drawback to using tapes is that they are slow, so make sure you have plenty of time.

Sequential Disk Files

Sequential disk files are best if you have small amounts of data to process. The files have the advantages of being faster than tape and more space conservative than random access files. Probably the only disadvantage of sequential disk files is the slowness of updating large files. In order to change a single record on a sequential file, you must copy all records to a work file, changing any records desired along the way, then delete the old file and rename the work file. This could be as time consuming as tape files, were it not for the speed of the disk.

Random Access Disk Files

Large volumes of data which must be updated with any frequency should be held in random access files. This type of file lets you easily update any given record without having to process or read through any other record on the file. It also has disadvantages such as requiring all records to be of the same, fixed length and needing to know where on the file a particular record is located.

There are several methods available to help

ADVERTISEMENT

Educational News

Orange Plus Educational News located at 23801 Calabasas Road/Suite 2050/Calabasas, CA 91302/(213) 999-5210

Legal Apple[Type] Compatible

New Computer runs CP/M & Apple[Type] Software

CALABASAS — Now instructors can teach Apple-type programs such as LOGO in one class and professional-style word processing in another without any hardware changes. Dual microprocessors (Z80-A & 6502, two computers in one case), with the radical new Orange Plus development the "EuroROM", allows the machine to read/write/work with Apple-type software as well as CP/M programs and access either CPU via the keyboard.

The ***"ORANGE+TWO"***™ is a brand new direction in the evolution of the personal computer. ORANGEFORTH-83, a Fig-FORTH derivative and readily accessible public domain language, is resident in the ROM*. Also included is CP/M 3.0, Digital Research's latest CP/M version. There is also a built-in disk drive controller for two Apple-type drives, a cassette interface, joystick port, color graphics and ASCII keyboard with numeric keypad. For expert word processing, the keyboard features upper and lower case (lower case characters are true descenders) with auto repeat.

This new breed of computer is a breakthrough for educators and school systems throughout the world. The ability to run both Apple-type and CP/M software on the same machine relieves financially-pressed educators from expensive equipment burdens, allowing them to spend more money where it counts...**on the teachers.**

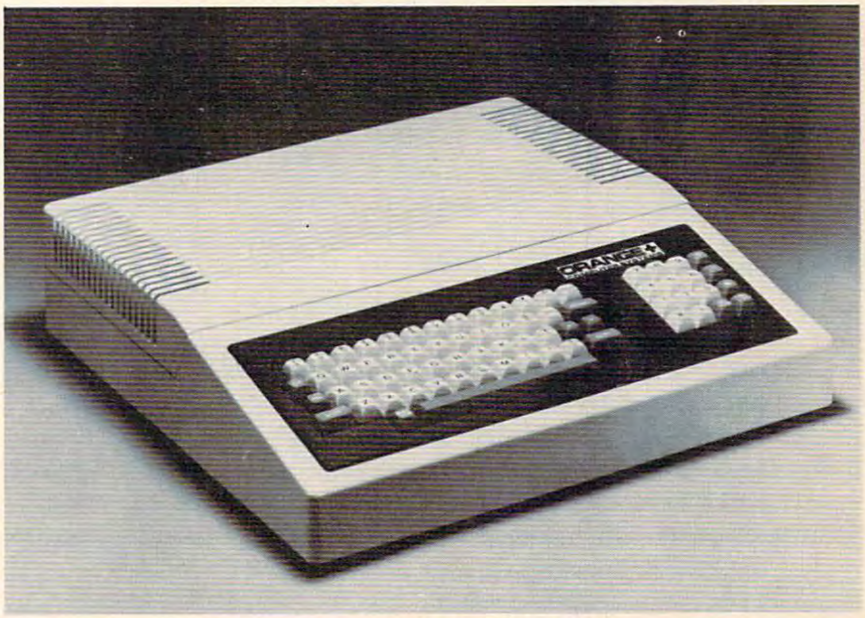
*Available on disk at extrta cost.

THE LEASING ALTERNATIVE

CALABASAS — Through select leasing companies, Orange Plus Computer Systems will offer the following lease programs to qualified corporations and educational institutions: An ***"ORANGE+TWO"***™ computer with a green monitor and one disk drive for only $59.55 per month...36 month closed end lease. Also included is a full maintenance program and all revisions and upgrades that may be available during the term of the lease. Subject to credit approval. Call for additional information.

DISTRIBUTORS: CALL (213) 999-5210

CALABASAS — Due to the tremendous amount of individual inquiries and dealer applications, qualified stocking distributors are needed. Select areas available!



MORE FOR LESS

The ***"ORANGE+TWO"***™ 64K computer includes these standard features:

Built-in CP/M 2.2 compatibility	**no charge**
Digital Research's CP/M 3.0	**no charge**
Built-in disk drive controller for two Apple-type drives	**no charge**
Numeric keypad with separate return key	**no charge**
110/220 volt switch selectable power supply, 50-60 Hz	**no charge**
Full function ASCII keyboard with auto repeat	**no charge**
Cassette interface	**no charge**
Fully grounded metal base plate	**no charge**
Adjustable audio volume control	**no charge**
ORANGEFORTH-83, Z80-A FORTH language, resident in ROM	**no charge**
Z80-A CPU, a second computer	**no charge**
6 slot double sided logic board	**no charge**
Bank switchable RAM, fully socketed, expandable to 256K	**no charge**
Programmable 2764 EPROMs	**no charge**
TOTAL: *"ORANGE+TWO"*™	**$1095.00**

*Compare these features included on the **"ORANGE+TWO"**™ with the competition's over $2000 for equivalent product.*

MORE PRODUCTS FROM ORANGE PLUS COMPUTER SYSTEMS TO ENHANCE YOUR *"ORANGE+TWO"*™, APPLE II/II PLUS, OR FRANKLIN ACE 1000

Disk-based ORANGEFORTH-83 language with full documentation	**$99.95**
Digital Research's™ CBASIC with disk and documentation	**$49.95**
Orange Plus KoalaPad™ Touch Tablet w/Micro Illustrator™	**$124.95**
Orange Plus Joystick (works on "ORANGE+TWO", Apple II, II+, & //e)	**$29.95**
Orange Plus self-centering joystick	**$39.95**
10MB (Formatted) 5¼" half-height Winchester Hard Disk Drive	**$1495.00**
Controller & Interface for Hard Disk	**$395.00**

*Includes everything necessary to be installed in **"Orange+Two"**, Apple or Franklin Computer*

The above are suggested retail prices. Prices may vary from state to state.

Collins International Trading Corporation,
23801 Calabasas Road, Suite 2050, Calabasas, CA 91302

you to determine where a particular record is located on a random access file. John Hudson covered the HASH/LINK method in the March 1982 issue of COMPUTE!. He did an excellent job; and if you desire to learn more about it, I suggest that you read this article. The HASH/LINK method does have some problems. For example:

I) If you fill the overflow area, you will have to reorganize the file again.

II) As soon as you initialize the random file, you take up more space than you may need.

III) Successive "collisions" can greatly increase access time (rec 100 links to rec 212, rec 212 links to rec 487, rec 487 links to...).

IV) Expanding the main and overflow areas of the file may require major program revisions (deciding the main area should be 2000 recs instead of 1000 recs will require changes to your hashing logic), as well as requiring you to reload the file.

V) Sequential (ascending or descending) processing is almost impossible.

VI) If you need to "key" on an alphabetic field (such as a name), you must first convert it to a numeric value.

VII) Once the file has been created, it is impossible to select an alternate key (e.g., a file is hashed on the last name, but you need a report in social security number order).

VIII) Deleting a record requires several Read/Write steps to keep the link field updated. Once a record has been deleted, the position that it occupied on the file is unusable, since all adds occur at the end of the file.

In the rest of this article I will cover an alternate method known as Indexed-Sequential Access Method (ISAM).

ISAM

ISAM can solve all the problems associated with HASH/LINK files, but it has some problems of its own. ISAM works on the principle that it is faster to search memory than a disk. Unfortunately, before you can search memory, you must have something in it, and this is the problem with ISAM.

ISAM works by loading the desired "key" field of each record in a file into an array. This is done by placing the key field of the first record into the first position of the array, the key from the second record into the second position of the array, etc. Once the array has been loaded, you simply search the array for the desired key; its position in the array is the record number for the random access file. Described below are the procedures necessary for the most common types of file processing:

I) ADD A RECORD
- a) Search the array to determine if the record already exists.
- b) Move the new "key" to the end of the array, or to the first "open" position in the array.
- c) Use the position number of the array to write the record to the file.

II) DELETE A RECORD
- a) Find the key in the array.
- b) "Open" the entry in the array by moving a "dummy" key into the array (such as zeros).
- c) Write the dummy values to the file.

III) CHANGE A RECORD
- a) Find the key in the array.
- b) Use the position number to read in the record.
- c) Make your change to the record (even change the key).
- d) Write the new record to the file using the position number.
- e) If you changed the key, move the new key into the array.

IV) PROCESS SEQUENTIALLY BY KEY
- a) Sort the array into the desired order (ascending or descending).
- b) Process the records sequentially through the array.

V) PROCESS BY A DIFFERENT KEY
- a) Load the array with the new keys from the file.
- b) Process normally using the new array.

Listed below are sample programs, written in Applesoft, which illustrate ISAM programming techniques. The programs are shells which can easily be modified to suit your own purposes. Note that all branch instructions bypass the REM statements; thus, if you want to key the program in without remarks, no line numbers will have to be changed. Variables used in the programs are:

D$ – Control-D (disk access)

IA – Index Array

IE – Index End (last entry used)

IP – Index Pointer (entry number for the part searched for)

IO – Index Open (entry number for first "open" or empty record)

FOUND – Switch to indicate if part searched for is in the index:

0 – part not in index

1 – part in index

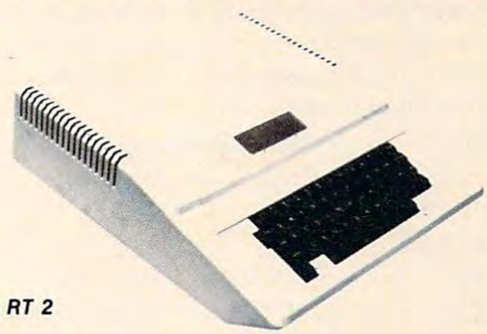
PART – Part number being searched for

IT'S HERE AT LAST!

THE FIRST LOW COST 6502 COMPUTER WITH 48K RAM THAT'S COMPLETELY COMPATIBLE WITH APPLE II®

Introducing the RAM-TECH* *RT 2* computer which offers you greater flexibility than the Apple II+® at a price that's easier on your finances. The *RT 2* will run all software and use all peripheral hardware designed for the Apple II+.® It's 100% Apple® compatible yet it offers you features not available on the Apple II+.®

COMPARE THESE FEATURES:



— supports upper & lower case characters from the keyboard
— high quality ABS case will not crack and is light weight
— RF modulator included in addition to composite video output
— high quality light-touch keyboard with upper/lower case key
— 15 key numeric key pad for fast data entry (optional)
— high power switching power supply (8.5 amps total) will power two disk drives and several peripherals easily, has internal circuit breaker
— eight expansion slots to increase flexibility with peripheral cards
— 48K RAM expandable to 64K (192K RAM upgrade card available soon)
— socketed IC's for quick and easy service
— thousands of programs (business & home) available
— operates identically to the Apple II+® with exception of extra features on *RT 2*
— full 90 days parts and labor warranty with fast service turn-around

THE RAM-TECH* *RT 2* IS EVERYTHING YOU ALWAYS WANTED FROM AN APPLE® (AND MORE), BUT THOUGHT YOU COULDN'T AFFORD. AND THAT'S GOOD NEWS!

BUT THE BEST NEWS IS THE PRICE:
$525.00 + $18.00 shipping & handling

WITH OPTIONAL NUMERIC KEYPAD:
$575.00 + $18.00 shipping & handling

COMPARED TO APPLE II® WHICH RETAILS AT MORE THAN DOUBLE THIS PRICE AND OFFERS YOU LESS, YOU CAN SEE WHY THE RAM-TECH* *RT 2* SHOULD BE YOUR CHOICE. *MAKE NO SYNTAX ABOUT IT, THE RT 2 IS FOR YOU!*

PERSONAL I.D. OPTION — We will burn your name into ROM. When you power up your Ram-Tech your name will be displayed on the screen. ***AN EXTRA MEASURE OF SECURITY AGAINST THEFT.***
Indicate on your order what name you wish to use (maximum 8 letters including spaces) — ADD: $25.00 for I.D. option.

ALSO AVAILABLE: (sub-assemblies for do-it yourselfers) — 90 days warranty

1 – ABS case, high quality, Apple® look-a-like and color matched, will replace Apple® case	$ 68.50
2 – same as #1 above but with numeric keypad cutout	$ 84.00
3 – keyboard with upper/lower case key, direct replacement for Apple® or Ram-Tech*	$110.00
4 – same as #3 above but with numeric keypad, use in combination with #2, also fits Apple®	$139.50
5 – RF modulator with TV switch and cable, also fits Apple®	$ 19.50
6 – switching power supply, 8.5 amps, internal circuit breaker, high power, also fits Apple®	$112.00
7 – motherboard, completely assembled & tested, fully operational with burned ROMS	$295.00
8 – bare motherboard w/ parts discription screened on board and instructions, highest quality G10 epoxy	$ 48.00

SHIPPING & HANDLING FOR ABOVE: cases #1 & 2 add 10%; all others add 5%

PERIPHERALS, CARDS & ACCESSORIES FOR YOUR RAM-TECH* OR APPLE®

all items #1 to 19 guaranteed for one year (disk drives for six months)

1 – 16K RAM card	$ 58.00
2 – 80 column card	$ 94.50
3 – clock card	$135.00
4 – communications card	$ 89.00
5 – disk controller card	$ 58.00
6 – EPROM writer card	$124.50
7 – FORTH card	$ 98.50
8 – IEEE-488 card w/cable	$186.00
9 – parallel printer card w/cable	$112.00
11 – interger card	$ 97.50
12 – PAL card	$114.00
13 – language card	$ 72.50
14 – RS-232 serial card	$ 84.00
15 – Z-80 CP/M card	$ 92.00
16 – VERSA card	$345.00
17 – analog/digital card	$116.50
18 – system cooling fan	$ 63.00
19 – joy stick w/button	$ 24.50
10 – parallel printer buffer and grappler card with cable	$245.00

DISK DRIVES: Guaranteed high quality for use with all Apple® compatible (or Ram-Tech*) controllers – DOS 3.3 & 3.2

USES SPECIAL LOW POWER CONSUMPTION CIRCUITRY

TEAC 55A	$325.00	$370.00 with controller
SHUGART	$285.00	$330.00 with controller

SHIPPING & HANDLING: add $3.50 per item #1 - 19 . . . disk drives add $10.00

TO ORDER

— ALL PRICES IN U.S. CURRENCY
— VISA OR MASTER CARD: SEND NUMBER, EXPIRY DATE, BANK NAME & CARD HOLDER'S NAME, INCLUDE PHONE NUMBER.
— CHECKS & MONEY ORDERS O.K. – CHECKS NEED THREE WEEKS TO CLEAR. (NO C.O.D.'S TO U.S. DUE TO POSTAL REGULATIONS)
— CANADIAN ORDERS ADD 25% FOR CDN FUNDS EXCHANGE.
— U.S. CUSTOMERS WILL HAVE TO PAY U.S. CUSTOMS 4.7% DUTY. DUTIES COLLECTED BY POST OFFICE. WE ASSUME NO LIABILITY FOR CUSTOMS CLEARANCE.

DEALER INQUIRIES REQUESTED

BYTE-RYTE

DEPT. CP
P.O. BOX 205, STATION CART.
MONTREAL, QUEBEC, CANADA H4K 2J5
(514) 335-1717

10-13 This section goes to a one-time routine to load the index array with the desired key field (in this case a part number).

100-114 Display the options available in a menu format.

120-122 This gets the option into a string. Then, using the VAL command, goes to the appropriate routine. Note that if zero, a non-numeric character, or a number greater than five is entered, the menu is displayed again.

200-215 The index array is searched sequentially in this section. If the key is found, the following values are returned:

FOUND =1
IP =Entry in array for desired key
IO =First open entry in array (entry with key of zero)

If the key is not found, the following values are returned:

FOUND =0
IO =First open entry in array

Note on lines 212 and 213 the method used to exit from the FOR/NEXT loop. This is the method suggested by Apple to exit the loop from other than normal completion. Its purpose is to prevent ?OUT OF MEMORY errors from occurring as a result of too many "open" loops.

300-324 ADD A PART

310 Accepts the part number to be added to the file.

311 Goes to the routine to search the index. If the part already exists (FOUND=1), an error message is displayed and control is returned to the menu.

321-322 The new part is written to the master file using the open entry pointer (IO) as the record number.

323 If the new part is added to the end of the file, the number of the last entry (IE) is updated.

324 Returns to the menu.

400-424 DELETE A PART

410 Accepts the part to be deleted.

411 Goes to the search routine. If the part is not on file (FOUND=0), an error message is displayed and control is transferred to the menu.

420 The part is removed from the index by making the entry zero.

421-422 The part is removed from the master file.

423 If the part was the last one in the array, the ending pointer (IE) is reduced by one.

424 Return to the menu.

800-813 UPDATE INDEX POINTER

810-811 Write the number of the last entry in the index to record zero of the master file.

812 Closes the master file.

813 Stops the program.

900-930 LOAD THE INDEX ARRAY

910 Initially sets up variables.

911 Sets up an error routine to handle end-of-data and not-found conditions.

912 Opens the master file.

913-914 Read the number of the last record on the master file.

915 Turns off the error routine, dimensions the index array to allow up to ten records to be added to the end of the array (this can be changed to allow for more expansion).

916 If no records exist on the master, control goes to the menu.

920 Sets up the error routine.

921-924 Load the key field (part number) into the array.

930 Turns the error routine off; returns to the menu.

The second program offers a different method of handling the index. Type in lines 10-630 from Program 1, then add the lines from Program 2. In this program the index is kept on a sequential disk file, for speed of loading the array.

800-833 Save the index array.

810 Check the index change switch; if it is zero, the index has not changed and does not have to be rewritten. Control goes to 832.

811 Deletes the index file.

820-823 Write the array to the index file.

830-831 Write the number of the last entry in the index to record zero of the master file.

832 Closes the master file.

833 Stops the program.

900-940 LOAD THE INDEX ARRAY

910 Initially sets up variables.

920 Opens the master file.

921 Sets up the error routine.

922-923 Read the number of entries in the index file.

930 Sets up a new error routine.

931 Dimensions the index array (with expansion of 10).

932-934 Read the index file into the array.

935 Turns the error routine off and closes the index.

940 Turns control over to the menu.

Program 1: ISAM

```
10  REM
11  REM  CALL INDEX LOAD ROUTINE
12  REM
13  GOTO 910
100  REM
101  REM  SELECT OPTION
```

```
102  REM
110  HOME : PRINT "1) ADD PART"
111  PRINT "2) DELETE PART"
112  PRINT "3) CHANGE PART"
113  PRINT "4) DISPLAY PART"
114  PRINT "5) STOP"
120  PRINT : INPUT "SELECT OPTION: ";OPT$
121  ON  VAL (OPT$) + 1 GOTO 110,310,410,510
     ,610,810
122  GOTO 110
200  REM
201  REM  SEARCH INDEX ARRAY
202  REM
210 IO = IE + 1: IF IE = 0 THEN FOUND = 0:
     RETURN
211  FOR I = 1 TO IE
212  IF IA(I) = PART THEN IP = I:I = IE + 1:
      NEXT :FOUND = 1: RETURN
213  IF IA(I) = 0 AND IO = IE + 1 THEN IO =
     I: NEXT
214  NEXT I
215 FOUND = 0: RETURN
300  REM
301  REM  ADD A PART
302  REM
310  INPUT "ENTER NEW PART NUMBER: ";PART
311  GOSUB 210: IF FOUND = 1 THEN  PRINT "PA
     RT ALREADY ON FILE": GOTO 110
320 IA(IO) = PART
321  PRINT D$;"WRITE MASTER,R";IO
322  PRINT PART: PRINT D$
323  IF IO > IE THEN IE = IO
324  GOTO 110
400  REM
401  REM  DELETE A PART
402  REM
410  INPUT "ENTER PART TO BE DELETED: ";PART

411  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
420 IA(IP) = 0
421  PRINT D$;"WRITE MASTER,R";IP
422  PRINT 0: PRINT D$
423  IF IP = IE THEN IE = IE - 1
424  GOTO 110
500  REM
501  REM  CHANGE A PART
502  REM
510  INPUT "ENTER PART TO BE CHANGED: ";PART

511  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
520  PRINT D$;"READ MASTER,R";IP
521  INPUT PART: PRINT D$
530  REM  CODING TO CHANGE PART
540 IA(IP) = PART
541  PRINT D$;"WRITE MASTER,R";IP
542  PRINT PART: PRINT D$
543  GOTO 110
600  REM
601  REM  DISPLAY PART
602  REM
610  INPUT "ENTER PART NUMBER: ";PART
611  GOSUB 210: IF FOUND = 0 THEN  PRINT "PA
     RT IS NOT ON FILE": GOTO 110
612  PRINT D$;"READ MASTER,R";IP
613  INPUT PART: PRINT D$
620  REM  CODING TO DISPLAY PART
630  GOTO 110
800  REM
801  REM  UPDATE INDEX POINTER
802  REM
810  PRINT D$;"WRITE MASTER,RO"
811  PRINT IE
812  PRINT D$;"CLOSE MASTER"
813  END
900  REM
901  REM  LOAD INDEX ARRAY
902  REM
910 D$ =  CHR$ (4):IE = 0:IP = 0:IO = 0:FOUN
    D = 0:PART = 0
911  ONERR  GOTO 915
912  PRINT D$;"OPEN MASTER,L25"
913  PRINT D$;"READ MASTER,RO"
914  INPUT IE: PRINT D$
915  POKE 216,0: DIM IA(IE + 10)
916  IF IE = 0 GOTO 110
920  ONERR  GOTO 924
921  FOR I = 1 TO IE
922  PRINT D$;"READ MASTER,R";I
923  INPUT IA(I)
924  NEXT I: PRINT D$
930  POKE 216,0: GOTO 110
```

Program 2: Index Array Routine

```
800  REM
801  REM  SAVE INDEX
802  REM
810  IF IC = 0 GOTO 832
811  PRINT D$;"DELETE INDEX"
820  PRINT D$;"OPEN INDEX"
821  PRINT D$;"WRITE INDEX"
822  FOR I = 1 TO IE: PRINT IA(I): NEXT I
823  PRINT D$;"CLOSE INDEX"
830  PRINT D$;"WRITE MASTER,RO"
831  PRINT IE
832  PRINT D$;"CLOSE MASTER"
833  END
900  REM
901  REM  LOAD INDEX ARRAY
902  REM
910 D$ =  CHR$ (4):IE = 0:IP = 0:IC = 0:IO =
     0:FOUND = 0:PART = 0
920  PRINT D$;"OPEN MASTER,L25"
921  ONERR  GOTO 930
922  PRINT D$;"READ MASTER,RO"
923  INPUT IE
930  ONERR  GOTO 935
931  DIM IA(IE + 10)
932  PRINT D$;"OPEN INDEX"
933  PRINT D$;"READ INDEX"
934  FOR I = 1 TO IE: INPUT IA(I): NEXT I
935  POKE 216,0: PRINT D$;"CLOSE INDEX"
940  GOTO 110
```

©

TI Cadette: Computer Aided Design

Bradley Rogers

This clever program should provide hours of amusement for children who enjoy creating pictures. Similar to coloring or cut-and-paste, the computer screen becomes a magic window allowing easy design, color selection, and erasure. Requires Extended BASIC and joysticks.

"Cadette" is for children. Based on a scaled-down version of CAD (the Computer Aided Design), it transforms your TV screen into an electronic easel on which children can "draw" tropical birds, planes, surreal landscapes, or any number of other fascinating pictures. Joysticks and fire buttons are used instead of conventional pens and brushes.

Using these simple instruments, children can create intricate designs from a basic stockpile of 16 different shapes. Each shape can assume five different colors chosen at the start of the program. Cadette calls upon the imagination, but does not require highly developed motor skills. Most children over five should be able to manage it nicely.

Cadette is simple to use, with only four basic activities required:

1. Choosing a page (screen) color;
2. Choosing five brush (shape) colors;
3. Moving joysticks to position the shapes or the eraser; and
4. Pressing fire buttons to print or to erase.

The process is the electronic equivalent to pasting cutouts on construction paper. However, the program involves considerably less frustration than conventional craft activities. It permits children to erase neatly or to change their minds at any point without having to start over with a clean sheet.

Running The Program

Once the RUN command has been entered, a brief message appears, instructing you to select a page color. The page in this case is, of course, the TV screen. Next, you are confronted by a display of 12 colors, each identified by a number from 1 to 12. From this menu you select a screen color by pressing the appropriate number key and then the ENTER key. If you enter anything other than numbers 1 to 12, the computer waits patiently for you to reconsider.

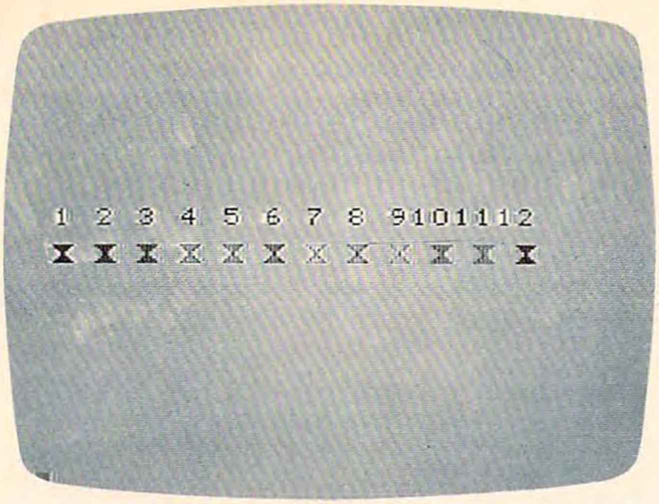
A second message now appears on the background color you chose. You are to select five brush colors. This message disappears, and you are asked to choose five from among twelve brush colors. The brushes in this case represent the colors of the shapes you will eventually use to create your design. Simply enter your five choices and remember to press ENTER after each selection.

After the color choices, the screen will blank and 16 geometric shapes will appear, eight across the top of the screen, and eight across the bottom. They consist of a circle, a square, assorted lines, triangles, and semicircles. Every few seconds the color of all 16 shapes changes, running through a cycle of five color changes, and then repeating.

Near the center of the screen is a small hollow box, which is the cursor. By using either of the joysticks, you move the cursor to capture and transport the colored shapes. After deciding which shape you want to capture, move the cursor to a position immediately adjacent to the shape. Once the shape turns the desired color, position the cursor on the shape.

The cursor will then disappear, and a duplicate of the colored shape you chose will appear immediately above or below the original, depending upon whether you selected from the top or bottom row. This duplicate may now be moved with the joysticks to any desired location. It will maintain its shape and color no matter what else happens on the screen. The original from which it was copied will remain in its display row and continue to undergo color transformations.

The duplicate shape, which now represents the cursor, can be placed at any position on the screen. Move it to the location you want and simply press the fire button. You will hear a low tone indicating that the button has done its job. If you have picked up the right joystick, the shape will "lock" at that screen location. Even if you move the cursor, the shape will remain fixed as



Twelve page and twelve brush colors are available in "Cadette" from the TI-99.

long as the program runs. If you have picked up the wrong joystick, the shape will be erased.

Assuming you have the "lock" joystick, you now have two options. You can move the cursor shape to a new position and print it again, or you can select another shape of the same color or the same shape of another color. If you choose a new shape, repeat the initial capture procedure. Remember, however, that the cursor no longer appears as a hollow box, but in the shape of your previous selection. But once it is placed on a new colored shape, it will automatically assume the new shape and color.

The "lock" joystick locks your selection at the location you want. The other joystick also controls the cursor, but is used to erase. To erase a "locked" shape, simply move the cursor on top of that shape and press the fire button. A higher tone will sound, the shape will disappear, and you can make another selection. To avoid confusion, you might label one joystick "lock" or "print" and the other "erase."

Extensions And Modifications

If you want to alter the shapes, you can change lines 540, 560, 580, and 600, which are DATA statements that contain the hexadecimal representations of the shapes. Each shape is defined by a string of 16 hexadecimal numbers.

Some children may find that the cursor moves too quickly, rushing past the space in which they wanted to print a shape. You can change the cursor's speed in line 920 by adjusting the limit (4) in the FOR/NEXT loop.

One interesting modification to the program would make it more versatile without requiring a great deal of extra programming. For example, a larger menu of shapes could be shown initially, and 16 could then be chosen from it. This would not be a terribly complicated program adjustment as long as you remember that the shapes must be read into S$. It is better to present the shape menu before the color menus; once you start fooling with color statements, all kinds of unexpected complications develop. In considering such modifications, just remember that often there is a tradeoff between versatility and user-convenience. The program could become less fun to use if a child has to make too many decisions.

Cadette

```
100 CALL CLEAR
110 CALL SCREEN(15)
120 DISPLAY AT(5,6):"SELECT PAGE CO
    LOR, 1 TO 12."
130 FOR I=1 TO 800 :: NEXT I :: CAL
    L CLEAR
140 DIM Z(5):: DIM S$(16)
150 A$="3C7EFFFFFFFF7E3C" :: X=4
160 FOR I=62 TO 142 STEP 8 :: CALL
    CHAR(I,A$):: CALL COLOR(X,X-1,1
    ):: X=X+1 :: NEXT I
170 CALL CHAR(40,A$):: CALL COLOR(2
    ,16,1):: CALL COLOR(9,14,1):: C
    ALL VCHAR(12,5,40):: X=3
180 FOR I=62 TO 142 STEP 8 :: CALL
    VCHAR(12,X*2+1,I):: X=X+1 :: NE
    XT I
190 CALL CHARPAT(56,Z$):: CALL CHAR
    PAT(57,W$):: CALL CHAR(33,Z$)::
     CALL CHAR(34,W$)
200 FOR I=1 TO 12
210 IF I=8 THEN CALL VCHAR(10,19,33
    ):: GOTO 240
220 IF I=9 THEN CALL VCHAR(10,21,34
    ):: GO TO 240
230 DISPLAY AT(10,I*2):USING "##":I
240 NEXT I
250 ACCEPT AT(24,1)VALIDATE(DIGIT)B
    EEP:Y
260 IF Y<1 OR Y>12 THEN 250
270 IF Y=7 THEN Y=13
280 IF Y=1 THEN Y=15
290 CALL SCREEN(Y+1):: CALL CLEAR
300 FOR I=1 TO 14 :: CALL COLOR(I,2
    ,1):: NEXT I :: DISPLAY AT(6,4)
    :"SELECT 5 BRUSH COLORS,{6 SPACE
    S}1 TO 12."
310 FOR I=1 TO 800 :: NEXT I :: CAL
    L CLEAR
320 FOR I=4 TO 14 :: CALL COLOR(I,I
    -1,1):: NEXT I :: CALL COLOR(2,
    16,1):: CALL COLOR(9,14,1)
330 A$="FF7E3C18183C7EFF"
340 FOR I=62 TO 142 STEP 8 :: CALL
    CHAR(I,A$):: NEXT I :: CALL CHA
    R(40,A$)
350 IF Y=15 THEN Y=0
360 IF Y=13 THEN Y=7
370 CALL COLOR(Y+2,2,1):: X=6 :: CA
    LL VCHAR(12,4,40)
380 FOR I=62 TO 142 STEP 8 :: CALL
    VCHAR(12,X,I):: X=X+2 :: NEXT I
390 FOR I=1 TO 12
400 IF I=8 THEN CALL VCHAR(10,18,33
    ):: GOTO 430
410 IF I=9 THEN CALL VCHAR(10,20,34
    ):: GOTO 430
420 DISPLAY AT(10,I*2-1):USING "##"
    :I
```

```
430 NEXT I
440 FOR I=1 TO 5
450 ACCEPT AT(24,1)VALIDATE(DIGIT)B
    EEP:Z(I)
460 IF Z(I)<1 OR Z(I)>12 THEN 450
470 IF Z(I)=1 THEN CALL VCHAR(18,I*
    2+2,40)ELSE 490
480 GO TO 500
490 CALL VCHAR(18,I*2+2,46+8*Z(I))
500 NEXT I
510 FOR I=1 TO 500 :: AA=8*8 :: NEX
    T I
520 CALL CLEAR
530 FOR I=1 TO 16 :: READ S$(I):: N
    EXT I
540 DATA 187E7EFFFF7E7E18,0107070F0
    F070701,80E0E0F0F0E0E080,FF7E7E
    1800000000
560 DATA 00000000187E7EFF,FFFFFFFF
    FFFFFFF,FFFEFCF8F0E0C080,000103
    070F1F3F7F
580 DATA 80C0E0F0F8FCFEFF,7F3F1F0F0
    7030100,8080808080808080,010101
    0101010101
600 DATA FF00000000000000,000000000
    00000FF,8040201008040201,010204
    0810204080
610 X=40
620 FOR I=1 TO 5
630 P=Z(I)+1
640 IF Z(I)=1 AND Y=0 THEN CALL COL
    OR(I*2,2,1):: CALL COLOR(I*2+1,
    2,1):: GO TO 690
650 IF Z(I)>1 AND Z(I)=Y THEN CALL
    COLOR(I*2,2,1):: CALL COLOR(I*2
    +1,2,1):: GO TO 690
660 IF Z(I)=1 AND Y>0 THEN P=16
670 IF Z(I)=7 AND Z(I)<>Y THEN P=14
680 CALL COLOR(I*2,P,1):: CALL COLO
    R(I*2+1,P,1)
690 FOR J=1 TO 16 :: CALL CHAR(X,S$
    (J)):: X=X+1 :: NEXT J
700 NEXT I
710 G2=32 :: H1=12 :: F1=16
720 CALL CHAR(37,"FF818181818181FF"
    ):: CALL VCHAR(H1,F1,37):: J=39
730 QW=1 :: UU=1
740 FOR I=1 TO 8 :: CALL VCHAR(24,I
    *2+4,J+I):: NEXT I
750 FOR I=9 TO 16 :: CALL VCHAR(1,I
    *2-12,J+I):: NEXT I
760 X=1
770 CALL JOYST(UU,F2,H2)
780 CALL KEY(UU,RV,SV)
790 IF (H1=1 AND H2=4)OR(H1=24 AND
    H2=-4)OR(F1=2 AND F2=-4)OR(F1=3
    0 AND F2=4)THEN F2=0 :: H2=0 ::
    GO TO 900
800 IF F2=0 AND H2=0 AND SV=0 THEN
    900
810 H3=H1-H2/4 :: F3=F1+F2/4
820 CALL GCHAR(H3,F3,G3):: CALL GCH
    AR(H1,F1,G1)
830 IF RV+QW=19 AND(H1=24 OR H1=1)T
    HEN 890
840 IF H3=24 AND G3<>32 THEN H3=23
    :: CALL VCHAR(H1,F1,32):: CALL
    VCHAR(H3,F3,G3):: G2=32 :: GO
    TO 870
850 IF H3=1 AND(G3<>32)THEN H3=2 ::
     CALL VCHAR(H1,F1,32):: CALL VC
    HAR(H3,F3,G3):: G2=32 :: GO TO
    870
860 IF (H3<>24 AND H3<>1)OR(H3=24 A
    ND G3=32)OR(H3=1 AND G3=32)THEN
     CALL VCHAR(H3,F3,G1):: CALL VC
     HAR(H1,F1,G2):: G2=G3
870 IF RV+QW=19 THEN CALL VCHAR(H3,
    F3,G1):: G2=G3 :: CALL SOUND(10
    0,110,2):: GOTO 890
880 IF RV+QW=17 THEN CALL VCHAR(H3,
    F3,37):: G2=32 :: CALL SOUND(10
    0,220,2)
890 H1=H3 :: F1=F3
900 QW=-QW
910 IF QW=1 THEN UU=1 ELSE UU=2
920 FOR AA=1 TO 4 :: NEXT AA
930 X=X+1 :: IF X=10 THEN 940 ELSE
    770
940 J=J+16
950 IF J>103 THEN J=39
960 GO TO 740
```

ATARI FONTBYTER

Orson Scott Card

It's hard to tell, when you're using "Fontbyter," whether this is a utility or a game. You can easily create graphics displays many times the size of the screen and save them to disk, using the ROM character set – or character sets you have designed yourself. And because Fontbyter allows you to use two "hidden" character modes, ANTIC modes 4 and 5, you get all the high-resolution color of Graphics 7 with the convenience and memory usage of Graphics 0.

Once you have a character set designed and a picture drawn on the screen using "Fontbyter," changing an 8-by-8-pixel character block takes only one POKE. This allows easy, almost instant animation; your programs can be shorter than they would be if you tried to get the same effect with Graphics 7; and you have more memory available to you because the screen displays take up less room.

The problem is creating the actual display. In ANTIC 4, you have 24 lines of 40 characters; in ANTIC 5, 12 lines of 40 characters. Laying out the screen display and writing the DATA statements can be a long, tedious, painful process. You have to remember what each character looks like and make sure that the characters are in the right order in the DATA statements you create. And when you want to change a display, you have to go back and find the right DATA statement and alter it.

Fontbyter lets you create and edit in ANTIC 4 or 5 right on the screen. You don't have to write down the number of the character and POKE it into memory; you only have to press a key or combination of keys, and your character will be displayed exactly where you want it on the screen. Simple commands allow you to fill large areas with a single character, insert or delete lines, scroll around the screen to view large areas quickly, or change the colors on the screen. And Fontbyter will scroll horizontally and vertically, so you can use the screen as a window onto a very large display – up to 4K.

Best of all, you can save your screen to disk at any point and return to continue editing it. Using a simple subroutine, you can then load your screen into memory in your own program. The first eight bytes of every file Fontbyter creates contain the mode number, the display width, the display height, and the five colors of the screen display.

Starting Fontbyter

Character set. When you RUN Fontbyter, the program accesses your disk and shows you a directory of all the files with the filetype ".SET". Fontbyter assumes that these are all character sets. The program then asks you to choose which one you want to use. Or, if you wish to use the built-in ROM character set, enter the character "@" as the filename.

There is only one custom character set included with Fontbyter, but by using a character editor you can create as many different sets as you want.

If the character set you ask for is not on the disk in drive 1, the program will prompt you to either insert the correct disk or ask for a different set. Also, whenever Fontbyter asks you for a filename, you don't need to enter more than the eight-character name – Fontbyter always supplies the device name "D1:" and the extender ".SET" or ".SCR". If you use an illegal name, Fontbyter will ask you to try again.

Screen files. When you have chosen your character set, Fontbyter displays a directory of all the files with the filetype ".SCR". Fontbyter assumes that these files contain screen displays created and saved by Fontbyter. If no directory is displayed, it means that there are no files with the filetype ".SCR" on the disk.

At the end of the directory, you will be told the number of sectors left on the disk. Be sure that the disk you use for saving screens has enough room for the screen you intend to save. A maximum-size display is almost 4K, which will create a file of 33 sectors. Disks can fill up pretty fast at that rate.

Save file. The program asks you what name your saved screen file should have. When you are through editing and want to save your finished screen, this is the filename that Fontbyter will use to create the save file. You can use a filename that you used before, but saving the new file will erase

the old one.
Load file. The program asks you if you want to edit a screen that was previously saved. If you do, you will be asked the name of the file you want to load from.

Notice that this system allows you to load from a file and then save your edited version back to the same file, erasing the old version; or you can choose to save the file under a different filename, so that both versions will exist. There is an added safeguard, too. When you save the screen display, it is first saved under the name "D1.TEMPFILE.SCR". Then Fontbyter asks you if you want to save it under the name you chose at the beginning of the program. If you change your mind about the save filename then, you can exit Fontbyter and use DOS to change "D: TEMPFILE.SCR" to whatever name you want.

Load file parameters. If your load file is found, Fontbyter immediately opens it and reads the first three bytes. Then it reminds you of the ANTIC mode, width (in characters), and height (in lines) of the file as it was saved. If you don't want to change those parameters, you can proceed directly to the final check; if you do want to change them, Fontbyter will ask you to choose the mode, width, and height of the file as if you were creating a new screen.

ANTIC mode. Fontbyter asks you to choose which ANTIC mode you want. The only choices are 2 (Graphics 0), 4, or 5. Mode 4 has shorter, squarer characters, and fits 24 lines on a screen. Mode 5 has tall, thin characters and fits only 12 lines on a screen. This means that a display file a hundred lines from top to bottom will give you more than eight distinct screen displays in ANTIC 5, but only just over four distinct displays in ANTIC 4. ANTIC 2 (Graphics 0) is included, even though it is not a four-color mode, so that you can use Fontbyter to create displays using the built-in ROM character set.

If you own an XL model (600XL, 800XL, 1200XL, 1400XL, or 1450XLD), ANTIC 4 and 5 correspond to Graphics 12 and 13.

Display width. The minimum width of a line is 40 characters. If you enter a number less than 40, Fontbyter will change it to 40. The maximum width depends on the mode. The limiting factor here is that all screen displays must fit within 4K. Because of this, the wider a screen display you choose, the fewer vertical lines you can have. You cannot have a line so wide that it would not allow the minimum number of lines. Since you will not be allowed any fewer than 24 screen lines in ANTIC 2 or 4, you naturally can't have as wide a screen as in mode 5, which has a minimum of 12 lines per screen.

Display height. The minimum height, in number of lines, is 12 lines for ANTIC 5 and 24 lines for ANTIC 2 and 4. The maximum height depends on the line width you chose. If you ask for more lines than the allowable maximum, Fontbyter will change the figure to the maximum.

Final check. Fontbyter clears the screen and then displays what your choices were: the character set, the file in which to save your screen, the file (if any) to load from, the mode, the width (in characters), and the height (in lines). If you want to make any changes, press OPTION. If you are satisfied with your choices, press START.

Fontbyter will display a wait message for a few moments, and then the screen will go completely blank. This is so that the setup operations will run faster. When Fontbyter is ready to go on – and it won't be long – either the load screen you asked for will appear or a cursor will appear in the upper-left-hand corner of a blank screen. The cursor is whatever the ESCAPE character looks like in the character set you chose.

Also, part of the character set will be displayed on the bottom four lines of the screen. The characters are arranged in the same order as the computer keyboard, so that you can easily figure out which key to press in order to display a particular character.

Editing Features

To use the keyboard. The character set is divided into three groups: regular, shifted, and control. You can change from one to another using the CAPS/LOWR key. To get the regular character group, press CAPS/LOWR. To get the shifted character group, press SHIFT and CAPS/LOWR at the same time. To get the control character group, press CONTROL and CAPS/LOWR at the same time. As soon as you make the change, the character keyboard display at the bottom of the screen will change to show you the characters now available.

Instead of the usual computer keyboard system of locking only the alphabetic keys into shifted and control functions, Fontbyter shifts the entire keyboard. After you press SHIFT and CAPS/LOWR, you can press any key on the keyboard and get the shifted character – without pressing SHIFT again. The same applies to CONTROL with CAPS/LOWR.

Some keys, of course, don't have a shifted or control value (ESC, DEL, and RETURN, for instance), and others usually display only the inverse of another character (SHIFT-TAB, for instance). Since these don't display a separate character, pressing them only produces the same character that you would get if you pressed the space bar – a blank. In addition, if your character set redefines the space bar character, that character will fill your display when it first comes up, and will appear on the screen whenever you enter a

EXPLORE A NEW DIMENSION IN SOFTWARE

Action!

INTRODUCING ACTION! — Now the fastest 8-bit language

Another first from OSS! ACTION! is a brand new language designed to run on 6502-based computers, including Atari, Apple II, and Commodore 64. A powerful, structured language, ACTION! can draw out a new, higher dimension of performance from these machines, with speeds never seen before. ACTION! combines some of the best features of such languages as Pascal, C, and Algol, and offers speeds over 100 times faster than BASIC interpreters.

ACTION! is ideal for games, music processing, real-time control, and many other applications. But if what you're really looking for is raw speed in compiled code, ACTION! is just for you. There's more . . . ACTION! comes with a 128-column screen editor which rivals word processing programs, as well as a monitor mode which allows you to choose between on-line activities. ACTION!'s unique one-pass compiler will accept code from memory, disk, or cassette, and ACTION! has the ability to include source library files.

ACTION! is provided in cartridge form only. Introductory price for ATARI Version $99.00
Call or write for availability of Apple II and Commodore 64 Versions.

A Strong Software Family
Other major systems software products from OSS include:

BASIC A+	the only logical upgrade to Atari BASIC with extra features for games and business programs....$80.00
C/65	the first native mode "small c" compiler for Atari and Apple computers....$80.00
MAC/65	the finest and fastest complete 6502 macro assembler/editor package you can buy....$80.00
BUG/65	a powerful, self-relocatable debugger. FREE with MAC/65....$34.95

And More...

OS/A+, the first and finest operating system for BOTH Atari and Apple II computers, is NOW included FREE as a part of every OSS systems software package. OS/A+ features a keyboard-driven, easy-to-use command processor, several simple resident commands, and logical and readable requests for even the most sophisticated utility commands. Versions of OS/A+ for some higher capacity drives are available at extra cost.

NOTE: Unless otherwise noted, all OSS products require 48K and at least one disk drive.

ASK YOUR DEALER, or call or write for our brochure.

OSS Optimized Systems Software, Inc. 10379 Lansdale Avenue • Cupertino • California • 95014 • (408) 446-3099 nonprinting character.

The keys do not produce their normal clicking sound, except for the command keys, which are described next.

Command keys. No matter which character group you are using, there are some key combinations that Fontbyter interprets as commands. Pressing INSERT and SHIFT together will insert a blank line on the screen. Pressing DELETE and SHIFT together will delete a line. Pressing CONTROL and an ARROW key together will cause the cursor to move.

Remember, to print the *character* represented by the CONTROL-ARROW combination, press only the ARROW key while the control group is locked in. To move the cursor, press CONTROL and ARROW at the same time, regardless of which group is locked in.

Inverse video (Atari logo) key. This key is a *toggle*. Pressing it switches between inverse and regular video. In ANTIC 2 (Graphics 0), this will cause all the characters you enter to be reversed, as the computer normally does. In ANTIC 4 and 5, however, this will cause Color 3 to take its value from color register 4 (memory location 711 instead of 710). It will affect, therefore, only one of the colors, and if a character does not contain any dots of Color 3, inverse mode won't have any effect at all.

CONTROL-ESC. This key combination is a toggle. It will switch between Still and Auto-Advance modes. In Still Mode, pressing noncommand keys will display a new character in the same place on the screen. In Auto-Advance Mode, pressing noncommand keys will display a new character and then advance the cursor to the next position to the right, unless doing so would take the cursor beyond the edge of the display.

To move the cursor. Either move the joystick in the direction you want to move, or press the appropriate CONTROL-ARROW key combination. Only the joystick allows diagonal movement.

When the cursor reaches the edge of the screen, the display will begin to scroll until it reaches the limits of display height and width you specified during start-up. If you are at the edge of the display, the cursor simply won't move any farther that direction.

Fast-fill function. Sometimes you will have large areas or lines to fill with the same character. Instead of entering the character by typing it in each space where it is to appear, you can use the joystick and fire button. First maneuver the cursor until it is on top of the character you want to copy, or move it to the place where you want to begin the fast-fill operation and enter the character from the keyboard. Then press down the joystick button and hold it down while you use the joystick to move the cursor. From then on, until you let up on the button, wherever you move the cursor using the joystick, a trail made up of that character will be left behind.

You can also use this function to erase areas of the screen fairly quickly. Just move the cursor to a blank, press down the button, and the cursor will leave blanks behind it wherever you make it go.

Clear screen function. To erase the entire display, press CONTROL-SHIFT-CLEAR.

Delete line function. To delete an entire line of your screen, move your cursor to the line you want to delete and press SHIFT-DELETE. The line will vanish, and the entire display below that line will move upward one line on the screen. Whether the very bottom of your display is visible on the screen or not, a line of blanks will be inserted as the last line in your display.

Insert line function. To insert a blank line in your display, move the cursor to the position where you want the new line. Then press SHIFT-INSERT. The line that the cursor was on will move down, as will all the other lines below it in the display, and the cursor will now be on a blank line. At the bottom of the display, whether it is visible on the screen or not, the last line of your display will be deleted completely.

With both the delete and insert line functions, the line that disappears is irrecoverably lost. To get it back, you will have to enter all the characters just as you did before. So take care when using these two functions.

By using the delete and insert functions in succession, you can quickly blank large areas of the screen, a line at a time. Simply move to the top of the area you want to blank out, and press SHIFT-DELETE as often as it takes to erase all the lines you wanted to get rid of. Then press SHIFT-INSERT until the desired number of blank lines appears.

You can also use these functions to move the entire picture up or down in the display. For instance, suppose you loaded a display that had been created and saved with only 24 lines, and you want to add another 24-line picture above it. At the beginning of the editing session, simply specify 48 lines as the height of the display. Fontbyter will put the 24 new blank lines at the end of the display. To move the old picture down into that blank area, start at the top of the screen and press SHIFT-INSERT 24 times.

Three joystick modes. We've already gone over the use of the joystick in Cursor Mode. The joystick can also be toggled into two other modes. If you press the START button while in Cursor Mode, the joystick will change to Scroll Mode. If you press the START button in Scroll Mode, the joystick will shift to Color Mode. And pressing the START button in Color Mode will shift you back to Cursor Mode again.

ComputAbility™

The Discount Software Company That Pays You A Dividend! !

ATARI

ATARI 800 48K	CALL
ATARI 800XL 64K	CALL
ATARI 810 DISK DRIVE	CALL
ATARI 1050 DISK DRIVE	CALL
ATARI 850 INTERFACE	CALL
ATARI 1010 RECORDER	77.00
HOME MANAGER KIT	64.95
ATARI 1027 PRINTER	289.00

Call for **PERCOM** Disk Drive Prices.

MOSAIC

64K Ram/400	149.00
64K Ram + Cable Kit /400/800	169.00
48K Ram Kit	94.00
16/32 Expander	64.95
32K Ram	77.95
Mosaic Adaptor	49.95

B-KEY 400 .... 39.95
AMDEK COLOR I 289.00

Suncom Joysensor...34.95
30 day money back guarantee

PRINTERS
PROWRITER 8510 AP ... CALL
Seikosha
W/Parallel Port....... 289.00

SUPER SPECIALS

FROGGER	ZAXXON	REPTON	PREPPIE II	CHOPLIFTER
$22.95 DISK/TAPE	$26.95 DISK/TAPE	$26.95 DISK	$22.95 DISK/TAPE	$21.49 DISK

PRICES EFFECTIVE NOW THROUGH SEPTEMBER 30, 1983

FREE* SOFTWARE AND ACCESSORIES

ATARI

Conversational Languages - T	43.95
Programming 2 & 3 - T	21.95
Music Composer - C	32.95
My First Alphabet - D	26.95
Touch Typing - T	19.95
Home Filing Manager - D	37.95
Mailing List - T	19.95
Asteroids - C	26.95
Caverns Of Mars - D	28.95
Computer Chess - C	26.95
Missile Command - C	26.95
Super Breakout - C	26.95
Star Raiders - C	32.95
Assembly Editor - C	44.95
Basic - C	39.95
Macroassembler - D	65.95
Microsoft Basic - D	65.95
Pilot (Home Package) - C	58.95
Invitation To Programming I - T	18.95
Speed Reading - T	55.95
Basketball - C	26.95
Graph-It - T	15.95
Juggles House - D/T	22.95
Pilot (Educator) - C	97.95
Video Easel - C	26.95
Defender - C	32.95
Galaxian - C	32.95
Qix - C	32.95
Dig Dug - C	32.95
ET - C	37.95
Timewise - D	23.95
Atariwriter - C	74.95
Donkey Kong - C	37.95
Ms. Pac-Man - C	39.95
Tennis - C	35.95
Eastern Front - C	32.95
Donkey Kong Jr. - C	39.95
Pengo - C	35.95
Logo - C	79.95
Robitron - C	35.95
Pole Position - C	39.95
Microsoft Basic II - C	65.95
Paint - D	33.95
Caverns of Mars - C	32.95

FIRST STAR

Astro Chase - D/T	23.95
Bristles - D/T	23.95
Flip Flop - D/T	23.95
Rent Wars - D/T	23.95

BIG FIVE

Miner 2049'er - Cart	39.95
Scraper Caper - C	39.95

COMPUTABILITY DIVIDEND COUPON PREMIUMS

3 M Blank Disk - Box of 10	27.00 or 10 Coupons
Elephant Disks - Box of 10	22.00 or 8 Coupons
Flip 'n File Disk Holder	20.95 or 6 Coupons
Flip 'n File Cartridge Holder	20.95 or 6 Coupons
Monitor Stand	27.95 or 10 Coupons
VU - Case Disk Holder W/Lock	24.00 or 10 Coupons
Heavy Plastic Dust Cover 800 or 810	9.95 or 3 Coupons
First Book of Atari Software 1983	17.95 or 6 Coupons
Kids & The Atari	17.95 or 6 Coupons
The Atari Assembler	12.95 or 5 Coupons
Atari Games & Recreation	14.95 or 6 Coupons
Atari Pilot for Beginners	14.95 or 6 Coupons
The Visicalc Book Atari	14.95 or 6 Coupons
Atari Basic	12.95 or 5 Coupons
Atari Sound & Graphics	9.95 or 3 Coupons
Starfighter	12.99 or 5 Coupons
Slik Stik	9.95 or 3 Coupons
Extension Cable - 6 ft.	6.95 or 3 Coupons
Lefty Adaptor	9.95 or 3 Coupons
Stik Stand	6.99 or 3 Coupons
Suncom Tac II	19.95 or 8 Coupons
Suncom Game Switch	6.95 or 3 Coupons
TI Adaptor	12.95 or 5 Coupons
Kraft Switch Hitter Joystick	17.95 or 7 Coupons
Kraft Joystick	15.95 or 6 Coupons
Maxell 2 Pack Disks	8.95 or 3 Coupons
Inside Atari Basic	12.95 or 5 Coupons
Compute's First Book of Atari	12.95 or 5 Coupons
Compute's Second Book of Atari	12.95 or 5 Coupons
De Re Atari	17.95 or 6 Coupons
Wico Joystick	24.00 or 10 Coupons
$24.00 Worth of Software	10 Coupons

*COUPON PROGRAM

The purchase of each program (with the exception of Super Specials, Atari and APX) will earn you 1 COMPUTABILITY DIVIDEND COUPON. Save 10 coupons and redeem them for your choice of any program we sell for $24.00 or less. Less than 10 coupons may be redeemed for premium items as indicated. You pay only a $2.50 shipping & handling charge.

DATASOFT

Pooyan - D/T	23.95
Moon Shuttle - D/T	23.95
Sands of Egypt - D	23.95
Text Wizard - D	37.95
Spell Wizard - D	37.95
Basic Compiler - D	59.95
Teletalk - D	37.95
Canyon Climber - D/T	20.95

SIRIUS

Bandits - D	27.95
Way Out - D	31.95
Repton - D	31.95
Twerps - D	27.95
Blade of Blackpoole - D	31.95
Type Attack - D	31.95
Beer Run - D	27.95
Critical Mass - D	31.95

ADVENTURE INTERNATIONAL

Preppie - D/T	23.95
S.A.G.A. Adventures - D	31.95
Sea Dragon - D/T	27.95
Stratos - D/T	27.95
Bugoff - D/T	23.95
Preppie II - D/C	27.95
Stone of Sisyphus - D	31.95
Eliminator - D/T	20.95
Rally Speedway - Cart	37.95
Ultra Disassembler - D	39.95
Diskey - D	39.95
Triad - D/T	27.95

SYNAPSE

Dimension X - D/T	27.95
Shadow World - D/T	27.95
Rainbow Walker - D/T	27.95
Drelbs - D/T	27.95
Shamus II - D/T	27.95
Blue Max - D/T	27.95
Fort Apocalypse - D/T	27.95
Necromancer - D/T	27.95
Pharoah's Curse - D/T	27.95
Survivor - D/T	27.95

PARKER BROS.

Q-Bert
Frogger
Astro Chase
Popeye
Tutankhamen
Risk
Chess

Please call for Prices

MISCELLANEOUS

Night Mission - D/T	23.95
Jumpman - D/T	31.95
Juice - D/T	23.95
Castle-Wolfenstein - D	23.95
Financial Wizard - D	44.95
PM Animator - D	27.95
Teletari - D	31.95
Starbowl Football - D/T	25.50
Ali Baba - D	26.50
Ultima II - D	44.95
Crisis Mountain - D	27.95
Star League Baseball - D/T	25.50
Home Accountant - D	54.95
Mountain King - Cart	31.95

D - Disk T - Cassette C - Cartridge

ATARI is a trademark of ATARI, Inc.

We Carry Hundreds of Items for ATARI 400/800, Ask for Our FREE CATALOG.

Mastercard/VISA Order Toll Free

MasterCard **800-558-0003** VISA

No surcharge for credit cards

In Wisc. Call **414/351-2007**

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

ORDERING INFORMATION: To order by mail, send money order, certified check, or personal check (allow 14 days to clear) to COMPUTABILITY. Include $2.00 shipping on all software orders. Include 3% shipping on all hardware orders, minimum $2.50. Mastercard and Visa please include card number and expiration date. WI residents please add 5% sales tax. Canadian, APO & FPO software orders include 5% shipping minimum $5.00. All other foreign software, please add 15% shipping, minimum $7.00 (US Funds only). Prices subject to change without notice.

ORDER HOURS
Mon-Fri 12-9PM (CST)
Sat 12-5 PM (CST)

1. Scroll Mode. This mode enables you to scroll the TV screen window around the entire display by moving the joystick in the appropriate direction. When you move, the cursor character will disappear. When you return to Cursor Mode, the cursor will come back to the middle of the screen.

2. Color Mode. In this mode, the joystick controls the color registers as follows:

- Forward and back: Color register 1 (Memory location 708)
- Left and right: Color register 2 (709)
- Forward and back with joystick button depressed: Color register 3 (710)
- Left and right with joystick button depressed: Background color register (712)
- Forward and back with SELECT depressed: Inverse color register (711)

Summary Of Command Keys

START. Cycle from Cursor Mode to Scroll Mode to Color Mode and back to Cursor Mode.

SELECT. Save the current display without interrupting the edit. In Color Mode, moving the joystick forward and back with SELECT pressed will change the inverse color.

OPTION. Save the current display and stop the editing session.

CONTROL-ARROW. Move the cursor.

SHIFT-INSERT. Insert a blank line where the cursor is, and delete the bottom line of the display.

SHIFT-DELETE. Delete the cursor line, and add a blank line at the bottom of the display.

Atari logo key. Toggle back and forth between inverse and regular characters.

SHIFT-CAPS/LOWR. Select the shifted character group.

CONTROL-CAPS/LOWR. Select the control character group.

CONTROL-ESC. Toggle between Still and Auto-Advance modes.

CONTROL-SHIFT-CLEAR. Erase the entire display.

As you press the joystick forward or to the right in Color Mode, that particular color will get brighter and brighter until it reaches maximum brightness; then it will jump to the next color at its darkest value and get brighter and brighter with that color. Pushing leftward or backward cycles through the colors from bright to dark. There are 16 colors, each with eight levels of brightness. You can cycle through the colors endlessly in either direction.

When you start editing with a new display, the colors are the system default colors. When you load a previously saved display, however, you start with the colors saved with that display. You can change the colors however you like, and whatever the colors are when you save your display, those values will be saved with it.

Ending And Saving

There are two ways to save a screen.

1. You can press the SELECT button when the joystick is in Cursor Mode, and the display will be saved as "D1:TEMPFILE.SCR". The screen is not changed, and you can resume editing as soon as the joystick or keyboard respond again.

2. You can press the OPTION button. Fontbyter will save the entire display in a file named "D1:TEMPFILE.SCR". The screen then clears, and Fontbyter asks if you want to save the display in the save file you asked for at the beginning of the edit. If you answer yes, "TEMPFILE" is renamed with the save filename you chose at the beginning. If a file with the same name already exists on the disk, it will be erased at this time.

If you are merely saving a half-done file to make sure some catastrophe doesn't lose it for you, then "TEMPFILE.SCR" should be security enough – if the system crashes, you'll know that the screen as you last saved it is in that file.

You will then be asked if you want to return to edit the same screen. If you say yes, your saved screen will quickly be reloaded into memory, and the program will reinitialize. If you say no, you will be asked whether you want to quit or start Fontbyter over again. If you choose the quit option and change your mind, don't worry. Just give the direct command RUN, and Fontbyter will begin again with the setup prompts.

Using Fontbyter Screens In Your Programs

Just because Fontbyter scrolls doesn't mean you have to make one continuous scrolling display. You can create many different screen displays in one file, "stacking" them vertically, and then use page flipping in your own program to move instantly from one to another. The advantage of using Fontbyter is that while you are creating them, you can scroll from one to the other to compare them and make sure that any animation effects you are trying for are working properly.

The diagrams will show you the variety of display configurations you can choose.

Here are subroutines you can include in your own programs to use the displays you have created with Fontbyter.

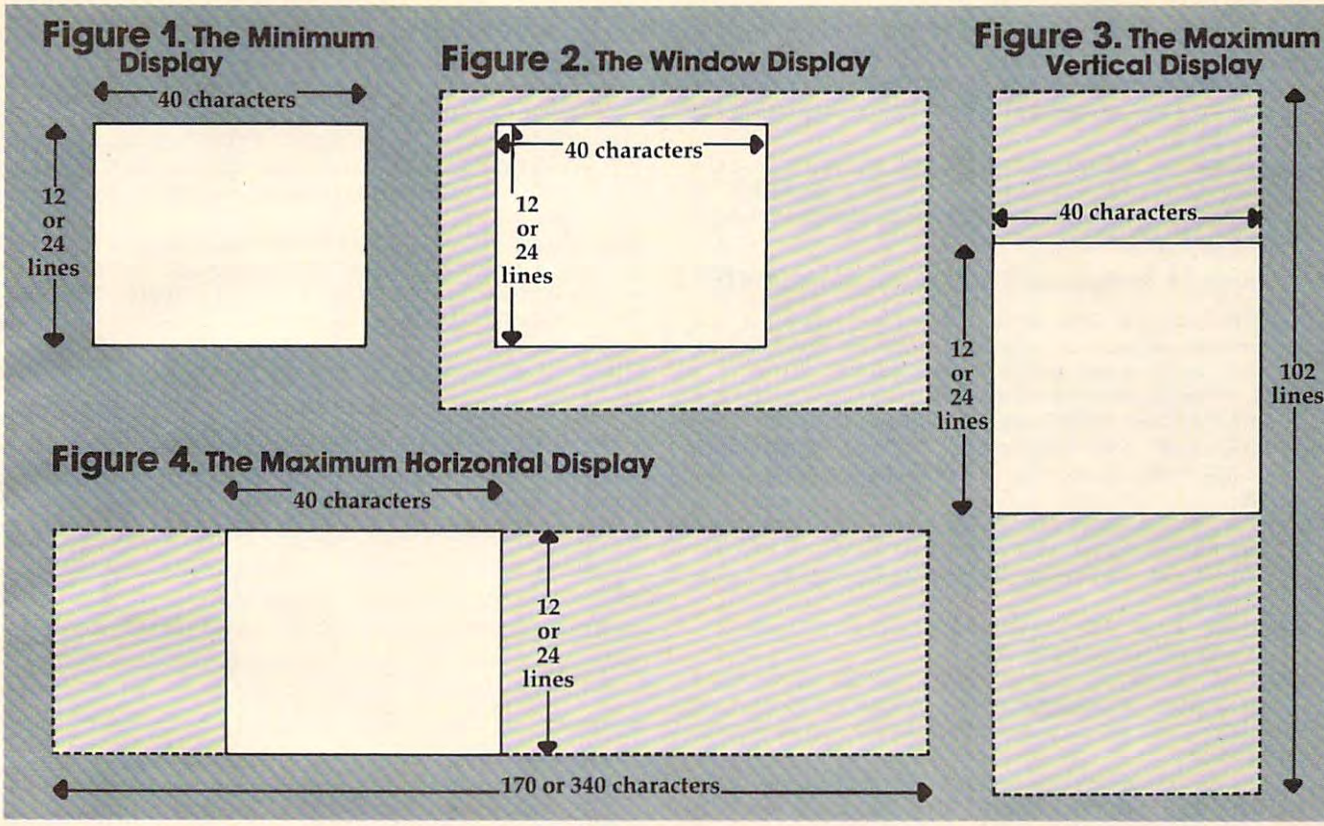


Figure 1. The Minimum Display

Figure 2. The Window Display

Figure 3. The Maximum Vertical Display

Figure 4. The Maximum Horizontal Display

Loading files. To use Fontbyter displays, your program will need to load a character set and the display file. Subroutine 1 loads slowly, entirely from BASIC. Subroutine 2 loads quickly, using a machine language routine that accesses an operating system fast-load program.

Subroutine 1. Slow Load

```
10 REM Slow load (character set)
100 OPEN #1,4,0,"D1:CHARACT.SET":FOR I=0 TO 1023:GET #1,N:POKE CH,N:NEXT I:CLOSE #1:RETURN
190 REM Slow load (display file)
200 OPEN #1,4,0,"D1:DRAWING.SCR":GET #1,MD:GET #1,WD:GET #1,LN:IF MD>5 THEN MD=MD-10:WD=WD+256
205 FOR I=708 TO 712:GET #1,N:POKE I,N:NEXT I:FOR I=0 TO WD*LN-1:GET #1,N:POKE SC+I,N:NEXT I:CLOSE #1:RETURN
```

Subroutine 2. Fast Load

```
10 REM Set up variables
20 X=16:ICCOM=834:ICBADR=836:ICBLEN=840:REM See text for meaning of variables SP and CHBAS
90 REM Fast load (display file)
100 OPEN #1,4,0,"D1:DRAWING.SCR":GET #1,MD:GET #1,WD:GET #1,LN:IF MD>5 THEN WD=WD+256:MD=MD-10
110 SZ=WD*LN:FOR I=708 TO 712:GET #1,N:POKE I,N:NEXT I
120 POKE ICBADR+X+1,SP:POKE ICBADR+X,0:POKE ICBLEN+X+1,1+INT(SZ/256):POKE ICBLEN+X,0
130 POKE ICCOM+X,7:I=USR(ADR("hhh█LV█"),X):CLOSE #1:RETURN
190 REM Fast load (character set)
200 OPEN #1,4,0,"D1:CHARACT.SET":POKE ICBADR+X+1,CHBAS:POKE ICBADR+X,0:POKE ICBLEN+X+1,4:POKE ICBLEN+X,0
210 POKE ICCOM+X,7:C=USR(ADR("hhh█LV█"),X):CLOSE #1:POKE 756,CHBAS:RETURN
```

Display list setup. Subroutine 3 sets up an ANTIC 2 or 4 display list that can be horizontally or vertically scrolled. Subroutine 4 sets up an ANTIC 5 display list that can be horizontally or vertically scrolled. Subroutines 5 and 6 set up display lists that *cannot* be horizontally scrolled – use these only to load displays that were created with the minimum line width.

On XL models, the simple display list can be set up with the BASIC statements Graphics 12 + 16 and Graphics 13 + 16, making the non-scrolling display list subroutine unnecessary. For horizontally scrolling displays, however, these subroutines are still needed.

Subroutine 3.
Horizontal Scroll Display List, ANTIC 2 or 4

```
10 REM Lines 20 and 30 are just a demonstration.  Change the value of SC and see what happens!
20 DL=PEEK(88)+256*PEEK(89):SC=DL:MODE=4:WIDE=40:GOSUB 100
30 FOR I=0 TO 1000:NEXT I:SC=0:MODE=2:GOSUB 100:FOR I=0 TO 1000:NEXT I:GOTO 20
90 REM This ANTIC 2 or 4 display list can be horizontally scrolled. Just set the values of SC,DL,MODE, and WIDE.
```

```
100 FOR I=Ø TO 2:POKE DL+I,112:NEXT
    I:N=Ø:M=MODE+64
110 FOR I=DL+3 TO DL+72 STEP 3:C=SC+
    N:POKE I,M:POKE I+2,INT(C/256):P
    OKE I+1,C-256*PEEK(I+2):N=N+WIDE
    :NEXT I
120 POKE I,65:DLHI=INT(DL/256):DLLO=
    DL-DLHI*256:POKE I+1,DLLO:POKE I
    +2,DLHI:POKE 561,DLHI:POKE 560,D
    LLO:RETURN
```

Subroutine 4. Horizontal Scroll Display List, ANTIC 5

```
10 REM Lines 2Ø and 3Ø are just a de
   monstration.  Change the value of
    SC and see what happens!
2Ø DL=PEEK(88)+256*PEEK(89):SC=PEEK(
   1Ø6)*256:MODE=5:WIDE=4Ø:GOSUB 1ØØ
3Ø FOR I=Ø TO 1ØØØ:NEXT I:SC=Ø:GOSUB
    1ØØ:FOR I=Ø TO 1ØØØ:NEXT I:GOTO
   2Ø
9Ø REM This ANTIC 5 display list can
    be horizontally scrolled.  Just
   set the values of SC,DL,MODE, and
    WIDE.
1ØØ FOR I=Ø TO 2:POKE DL+I,112:NEXT
    I:N=Ø:M=MODE+64
11Ø FOR I=DL+3 TO DL+36 STEP 3:C=SC+
    N:POKE I,M:POKE I+2,INT(C/256):P
    OKE I+1,C-256*PEEK(I+2):N=N+WIDE
    :NEXT I
12Ø POKE I,65:DLHI=INT(DL/256):DLLO=
    DL-DLHI*256:POKE I+1,DLLO:POKE I
    +2,DLHI:POKE 561,DLHI:POKE 56Ø,D
    LLO:RETURN
```

Subroutine 5. Regular Display List, ANTIC 2 or 4

```
1Ø REM The actual subroutine is line
   s 1ØØ-12Ø.  You set the value of
   DL,SC,MODE,and WIDE.
2Ø DL=PEEK(88)+256*PEEK(89):MODE=2:W
   IDE=4Ø
3Ø SC=Ø:MODE=2+2*(MODE=2):GOSUB 1ØØ
4Ø TRAP 3Ø:ON PEEK(753)<>3 GOTO 4Ø:S
   C=SC+48Ø:SP=INT(SC/256):POKE DL5,
   SP:POKE DL4,SC-256*SP
5Ø FOR I=Ø TO 3Ø:NEXT I:GOTO 4Ø
9Ø REM This ANTIC 2 and 4 display li
   st can be page flipped from BASIC
   .  POKE the screen address into D
   L4 and DL5.
1ØØ FOR I=Ø TO 2:POKE DL+I,112:NEXT
    I:DL4=DL+4:DL5=DL+5
11Ø POKE DL+3,64+MODE:POKE DL5,INT(S
    C/256):POKE DL4,SC-256*PEEK(DL5)
    :FOR I=DL+6 TO DL+28:POKE I,MODE
    :NEXT I
12Ø POKE I,65:DLHI=INT(DL/256):DLLO=
    DL-DLHI*256:POKE I+1,DLLO:POKE I
    +2,DLHI:POKE 561,DLHI:POKE 56Ø,D
    LLO:RETURN
```

Subroutine 6. Regular Display List, ANTIC 5

```
1Ø REM The actual subroutine is line
   s 1ØØ-12Ø.  You set the value of
   DL,SC,MODE,and WIDE.
2Ø DL=PEEK(88)+256*PEEK(89):MODE=5:W
   IDE=4Ø:GOSUB 1ØØ
3Ø SC=Ø
4Ø TRAP 3Ø:ON PEEK(753)<>3 GOTO 4Ø:S
   C=SC+48Ø:SP=INT(SC/256):POKE DL5,
   SP:POKE DL4,SC-256*SP
5Ø FOR I=Ø TO 3Ø:NEXT I:GOTO 4Ø
9Ø REM This ANTIC 5 display list can
    be page flipped from BASIC.  Jus
   t POKE the screen address into DL
   4 and DL5.
1ØØ FOR I=Ø TO 2:POKE DL+I,112:NEXT
    I:DL4=DL+4:DL5=DL+5
11Ø POKE DL+3,64+MODE:POKE DL5,INT(S
    C/256):POKE DL4,SC-256*PEEK(DL5)
    :FOR I=DL+6 TO DL+16:POKE I,MODE
    :NEXT I
12Ø POKE I,65:DLHI=INT(DL/256):DLLO=
    DL-DLHI*256:POKE I+1,DLLO:POKE I
    +2,DLHI:POKE 561,DLHI:POKE 56Ø,D
    LLO:RETURN
```

These routines use the following variables:

TOP is the page number of the top of memory. The Atari will not touch anything located above the top of memory – but anything below it is fair game. The display list, character set, screen memory, and machine language routines should all be placed above SP. So the load routines find out where the top of memory is and move it down enough pages to leave room for all the protected program areas. SC is the absolute address of the top of memory (SP*256); it is also the start of screen memory, so that it is POKEd into both the display list and location 106.

How much room should you leave? The character set takes 1K (four pages) and must start on a 1K boundary. Screen memory will never take more than 4K (16 pages), and should start on a 4K boundary, since ANTIC has problems when screen memory crosses that line. If your display is less than 2K, you can probably skip back from the top of memory a mere 4K (16 pages, or PEEK(106)-16), place screen memory at the new top of memory, and put the display list, machine language routines, and character set above it. If your display list is 3K or more, you should probably skip back 6K (24 pages, or PEEK(106)-24), place the character set at the new top of memory, followed by the display list, machine language subroutines, and then screen memory beginning at the 4K boundary line, 16 pages before the old top of memory. This routine assumes that arrangement.

SP is the page number of the start of screen memory, and SC is the absolute address of the start of screen memory (SP*256).

DL is the start of the display list. For page flipping, DL3 is DL+3, and DL4 is DL+4. These will contain the low byte and high byte of screen memory, and POKEing new values into these locations will flip screen memory.

CHBAS is the page number of the character set, and CH is the absolute address (CHBAS* 256).

MODE is the ANTIC mode number – either 2, 4, or 5. Adding 64 to MODE each time it is POKEd in tells the computer to look for a new screen memory address in the next two bytes in the display list.

WIDE is the width, in characters, of the entire horizontal line, not just the 40-character portion visible on screen at any one time. Thus, every MODE instruction is followed by a two-byte address, C, which tells it where to find the start of the next horizontal line.

POKEing 560 and 561 with 0 and DL/256 is what actually makes the display list start working. Until that moment, the display list is just a series of numbers in memory. But once 560 and 561 contain the address of the start of your display list, the TV screen is under your program's control.

ICBADR, ICBLEN, and ICCOM are the addresses of key locations in the IOCB handler. ICCOM must contain the number of the operation to be performed (7 to load, 11 to save). ICBADR must contain the low byte of the starting address of the area in memory to be saved from or loaded to (ICBADR + 1 will contain the high byte). ICBLEN must contain the low byte of the length of the file to load (ICBLEN + 1 will contain the high byte). The variable X represents the offset into the IOCB area. If you OPEN #1, then X = 16. If you OPEN #2, then X = 32. And so on, in multiples of 16. You might not get good results using OPEN #0 or OPEN #6 – those are reserved for system use.

With screen files created by Fontbyter, remember that the first eight bytes always contain the following information:

- ANTIC mode number (plus 10, if width is greater than 255 characters)
- width, or number of characters per line (low byte only, if width is greater than 255 characters)
- display height, or number of lines in the entire display
- colors to POKE into locations 708 through 712

To calculate the number of bytes in the whole screen display (SZ), multiply the height by the width. The number of bytes in the file is that number plus eight.

Typing The Program

The bulk of the program is written in BASIC. The shortest machine language routines are included as string constants. The longer routines, however, DISPLAY, EXPAND, and DELETE, and two data files, MENU.DAT and CHARDATA.DAT, are listed after the main program. These should be entered using the BASIC loader program provided and saved on disk with exactly the filename specified. Fontbyter will look for these files and load them into strings or particular areas or memory during the run of the program.

Since Fontbyter works most efficiently with a disk drive, the program as written assumes a disk drive. However, a patient cassette user can remove all the routines related to choosing and testing filenames, and simply assign the value "C:" to all filename variables. All machine language routines could be added as DATA statements. You may also want to add prompts to tell the user what file the program is asking for. The biggest problem arises with load files during initialization, when the program tests the saved screen file once, then loads it again later. If you decide not to revise the program, make sure that you rewind the cassette containing the screen file after that initial test, so the file will be complete when it is loaded by the screen load subroutine.

Program 1: Fontbyter

```
5 DIM F$(20),FSAVE$(20),FLOAD$(20),F
  L$(40),FLL$(20),DELETE$(124),EXPAN
  D$(124),CLEAR$(33),C(255)
10 GRAPHICS 0:X=16:ICCOM=834:ICBADR=
   836:ICBLEN=840
15 COL1=708:COL2=709:COL3=710:COL4=7
   11:COL5=712:SHIF=64:SCON=PEEK(559
   ):POKE 16,112:GOTO 440
20 OPEN #1,4,0,FL$:GET #1,MD:GET #1,
   WD:GET #1,LN:IF MD>5 THEN WD=WD+2
   56:MD=MD-10
25 SZ=WD*LN:FOR I=COL1 TO COL5:GET #
   1,N:POKE I,N:NEXT I
30 SC=SP*256:POKE ICBADR+X+1,SP:POKE
    ICBADR+X,0:POKE ICBLEN+X+1,1+INT
   (SZ/256):POKE ICBLEN+X,0
35 POKE ICCOM+X,7:I=USR(ADR("hhhLV
   "),X):CLOSE #1:RETURN
40 OPEN #1,8,0,"D1:TEMPFILE.SCR":WD=
   WIDE:MD=MODE:IF WIDE>255 THEN WD=
   WIDE-256:MD=MODE+10
45 PUT #1,MD:PUT #1,WD:PUT #1,LINE:F
   OR I=COL1 TO COL5:PUT #1,PEEK(I):
   NEXT I
50 POKE ICBADR+X+1,SP:POKE ICBADR+X,
   0:POKE ICBLEN+X+1,1+INT((LINE*WID
   E)/256):POKE ICBLEN+X,0
55 POKE ICCOM+X,11:I=USR(ADR("hhhLV
   "),X):CLOSE #1:RETURN
60 IF ((LINE*WIDE-PIX)<WIDE) THEN RE
   TURN
65 LOWAD=SC+WIDE*INT(PIX/WIDE)-1:HIA
   DD=LOWAD+WIDE:POKE 206,INT(HIADD/
   256):POKE 205,HIADD-PEEK(206)*256
70 POKE 204,INT(LOWAD/256):POKE 203,
   LOWAD-PEEK(204)*256:POKE SC+PIX,O
   LD
75 POKE 207,INT((LINE*WIDE-PIX)/WIDE
   ):POKE 208,WLO:POKE 209,WHI
80 C=USR(ADR(DELETE$))
85 OLD=PEEK(SC+PIX):POKE SC+PIX,91:R
   ETURN
90 IF ((LINE*WIDE-PIX)<WIDE) THEN RE
   TURN
95 HIADD=SC+WIDE*(LINE-1)-1:LOWAD=HI
   ADD-WIDE:POKE 206,INT(HIADD/256):
   POKE 205,HIADD-PEEK(206)*256
100 POKE 204,INT(LOWAD/256):POKE 203
    ,LOWAD-PEEK(204)*256:POKE SC+PIX
    ,OLD
105 POKE 207,INT((LINE*WIDE-PIX)/WID
    E):POKE 208,WLO:POKE 209,WHI
110 C=USR(ADR(EXPAND$))
115 OLD=0:POKE SC+PIX,91:RETURN
120 OLD=PEEK(SC+PIX):POKE SC+PIX,91:
    POKE 559,SCON:POKE 16,112
```

```
125 MV=Ø:V=Ø:H=Ø:OPT=PEEK(53279):DI=
    PEEK(632):T=PEEK(644):E=Ø
130 IF OPT=6 THEN GOSUB 870:GOSUB 26
    Ø:GOTO 125
135 IF DI<15 THEN GOSUB 155:GOTO 125
140 IF PEEK(753)=3 THEN GOSUB 22Ø:ON
     MV GOSUB 165:GOTO 125
145 ON OPT=3 GOTO 7Ø5:IF OPT=5 THEN
    GOSUB 4Ø:POKE SC+PIX,91:GOTO 125
15Ø GOTO 125
155 V=WIDE*((DI=9 OR DI=13 OR DI=5)-
    (DI=1Ø OR DI=14 OR DI=6)):POKE 7
    7,Ø
16Ø H=(DI=6 OR DI=7 OR DI=5)-(DI=1Ø
    OR DI=11 OR DI=9)
165 UD=INT(PIX/WIDE):IF UD-(V<Ø)<Ø O
    R UD+(V>Ø)=LINE THEN V=Ø
17Ø LR=PIX-WIDE*UD:IF LR+H<Ø OR LR+H
    >WIDE-1 THEN H=Ø
175 IF H=Ø AND V=Ø THEN 215
18Ø WH=Ø:WV=Ø:W=PEEK(DL4)+256*PEEK(D
    L5)-SC
185 U=INT(W/WIDE):IF V<>Ø THEN WV=(U
    D-U-(V<Ø)<Ø)+2*(UD-U+(V>Ø)>8+12*
    (MODE<>5))
19Ø IF H<>Ø THEN L=W-U*WIDE:WH=(LR+H
    -L<Ø)+2*(LR+H-L>39)
195 IF WH>Ø OR WV>Ø THEN POKE DL+114
    ,WH:POKE DL+115,WV:C=USR(DISPLAY)
2ØØ POKE SC+PIX,OLD:PIX=PIX+H+V:POKE
     53279,1
2Ø5 IF T=1 THEN OLD=PEEK(SC+PIX):POK
    E SC+PIX,91:GOTO 215
21Ø POKE SC+PIX,OLD
215 RETURN
22Ø GOSUB 785:ON (C=134)+(C=135)+(C=
    142)+(C=143)+2*(C=116)+3*(C=119)
    +4*(C=246) GOTO 25Ø,6Ø,9Ø,645
225 IF C=156 THEN AV=1*(AV=Ø):GOTO 9
    2Ø
23Ø IF N=6Ø THEN SHIF=4+C-64:POKE 53
    279,4:GOSUB 93Ø:RETURN
235 IF N=39 THEN VERS=128*(VERS=Ø):G
    OTO 92Ø
24Ø OLD=C(N+SHIF)+VERS:POKE SC+PIX,O
    LD:ON AV GOTO 245:RETURN
245 C=135
25Ø V=WIDE*((C=143)-(C=142)):H=(C=13
    5)-(C=134):MV=1:RETURN
255 GOSUB 92Ø:POKE SC+PIX,91:RETURN
26Ø GOSUB 92Ø
265 DI=PEEK(632):T=PEEK(644):DI=DI+5
    *(DI=7):DI=DI-1Ø:OPT=PEEK(53279)
    :IV=(OPT=5):IF OPT=6 THEN 255
27Ø IF DI<1 OR DI>4 THEN 265
275 ON (4*T)+DI GOSUB 28Ø,285,29Ø,29
    5,3ØØ,3Ø5,31Ø,32Ø:GOTO 265
28Ø POKE COL5,PEEK(COL5)-2+256*(PEEK
    (COL5)<2):RETURN
285 POKE COL5,PEEK(COL5)+2-256*(PEEK
    (COL5)>253):RETURN
29Ø POKE COL3,PEEK(COL3)-2+256*(PEEK
    (COL3)<2):RETURN
295 POKE COL3,PEEK(COL3)+2-256*(PEEK
    (COL3)>253):RETURN
3ØØ POKE COL2,PEEK(COL2)-2+256*(PEEK
    (COL2)<2):RETURN
3Ø5 POKE COL2,PEEK(COL2)+2-256*(PEEK
    (COL2)>253):RETURN
31Ø IF IV THEN POKE COL4,PEEK(COL4)-
    2+256*(PEEK(COL4)<2):RETURN
315 POKE COL1,PEEK(COL1)-2+256*(PEEK
    (COL1)<2):RETURN
32Ø IF IV THEN POKE COL4,PEEK(COL4)+
    2-256*(PEEK(COL4)>253):RETURN
325 POKE COL1,PEEK(COL1)+2-256*(PEEK
    (COL1)>253):RETURN
33Ø FLL$=FL$:FOR I=1 TO LEN(FL$):N=A
    SC(FL$(I,I)):ON N=58 GOSUB 37Ø:N
    EXT I:FL$=FLL$
335 FLL$=FL$:FOR I=1 TO LEN(FL$):N=A
    SC(FL$(I,I)):ON N=46 GOSUB 375:N
    EXT I:FL$=FLL$
34Ø IF LEN(FL$)>8 THEN FL$=FL$(1,8)
345 IF LEN(FL$)<1 THEN 39Ø
35Ø N=ASC(FL$(1,1)):IF N>9Ø OR N<65
    THEN 385
355 IF LEN(FL$)<2 THEN GOTO 365
36Ø FOR I=2 TO LEN(FL$):N=ASC(FL$(I,
    I)):ON (N>9Ø OR N<65) AND (N>57
    OR N<48) GOTO 38Ø:NEXT I
365 FLL$="D1:":FLL$(4)=FL$:N=Ø:RETURN
37Ø FLL$=FL$(I+1,LEN(FL$)):RETURN
375 FLL$=FL$(1,I-1):RETURN
38Ø POP :? "{CLEAR}":? "Illegal char
    acters in ";FL$:GOTO 39Ø
385 ? "{CLEAR}":? FL$;" must start w
    ith a capital":? "letter.":GOTO
    39Ø
39Ø ? "Let's try that name again.":N
    =1:RETURN
395 TRAP 4ØØ:OPEN #1,4,Ø,FL$:N=Ø:CLO
    SE #1:RETURN
4ØØ ? :? FL$;" isn't on disk in":? "
    drive 1":? "Insert disk with ";F
    L$;"and":? "press RETURN.":CLOSE
     #1
4Ø5 ? "Or to try another file name,
    press anyother key."
41Ø ON PEEK(753)<>3 GOTO 41Ø:GOSUB 7
    85:ON N=12 GOTO 395:N=1:RETURN
415 TRAP 435:OPEN #1,4,Ø,FL$:? FL$;"
     is already on disk.":? "Unless
    you change the name, the old"
42Ø ? "file will be lost.  To change
     the namepress RETURN":? "Or pre
    ss any other key to continue.":C
    LOSE #1
425 ON PEEK(753)<>3 GOTO 425:GOSUB 7
    85:ON N=12 GOTO 43Ø:N=Ø:RETURN
43Ø N=1:RETURN
435 CLOSE #1:N=Ø:RETURN
44Ø ? "{13 SPACES}Fontbyter":? :? :?
    :GOSUB 9Ø5
445 GOSUB 85Ø:? "What is the name of
     your character{4 SPACES}set? (E
    nter '@' for ROM set)":POKE 764,
    255:INPUT F$
45Ø IF F$="@" THEN 465
455 FL$=F$:GOSUB 33Ø:ON N GOTO 445:F
    $=FLL$:F$(LEN(FLL$)+1)=".SET"
46Ø FL$=F$:GOSUB 395:ON N GOTO 445
465 GOSUB 84Ø:? :? "What file should
     hold your finished{3 SPACES}scr
    een? (Eight characters)":POKE 76
    4,255:INPUT FSAVE$
47Ø FL$=FSAVE$:GOSUB 33Ø:ON N GOTO 4
    65:FSAVE$=FLL$:FSAVE$(LEN(FLL$)+
    1)=".SCR"
475 FL$=FSAVE$:GOSUB 415:ON N GOTO 4
    65
48Ø FLOAD$="":? :? "Would you like t
    o edit a screen you{3 SPACES}hav
    e already saved? (Y or N) "
485 GOSUB 785:ON N=35 GOTO 535:ON N=
    43 GOTO 49Ø:GOTO 485
49Ø ? :? "What is the name of the sa
    ved screen  file? ":POKE 764,255
```

```
:INPUT FLOAD$
495 FL$=FLOAD$:GOSUB 330:ON N=0 GOTO
 500:GOTO 480
500 FLOAD$=FLL$:FLOAD$(LEN(FLL$)+1)=
".SCR"
505 FL$=FLOAD$:GOSUB 395:ON N GOTO 4
80:OPEN #1,4,0,FLOAD$:GET #1,MD:
GET #1,WD:GET #1,LN:CLOSE #1:FLO
AD=1
510 IF MD>5 THEN MD=MD-10:WD=WD+256
515 ? :? FLOAD$;" was saved as:":? "
Mode ";MD;",":? "with ";LN;" lin
es":? "of ";WD;" characters per
line."
520 ? "If you wish to change these p
arameterspress RETURN.":? "To le
ave them unchanged press any
{5 SPACES}other key."
525 ON PEEK(753)<>3 GOTO 525:GOSUB 7
85:IF N=12 THEN 540
530 MODE=MD:WIDE=WD:LINE=LN:GOTO 585
535 FLOAD=0
540 ? :? "What Antic mode will you w
ork in?":? "(Antic 2, 4, OR 5) "
:POKE 764,255
545 GOSUB 785:ON N<>30 AND N<>24 AND
 N<>29 GOTO 545
550 MODE=C(N)-16
555 ? :? "How wide a line?":? "  (Mi
nimum 40 characters":? "
{3 SPACES}maximum ";170+170*(MOD
E=5);" characters)"
560 POKE 764,255:TRAP 560:INPUT WIDE
:WIDE=INT(WIDE):ON WIDE<40 OR WI
DE>170 GOSUB 790
565 ? :? "How many lines do you want
 to edit?{5 SPACES}(Minimum ";12
+12*(MODE<>5);:? "{3 SPACES}Maxi
mum ";INT(4096/WIDE);")"
570 TRAP 570:INPUT LINE
575 LINE=INT(LINE):IF LINE>INT(4096/
WIDE) THEN LINE=INT(4096/WIDE)
580 IF LINE<12+12*(MODE=4) THEN LINE
=12+12*(MODE=4)
585 ? "{CLEAR}":? "You have chosen:"
:? "Character set--";F$:? "Save
file--";FSAVE$:? "Load file--";F
LOAD$
590 SZ=LINE*WIDE-1:? "Mode ";MODE:?
LINE;" lines of ";WIDE;" charact
ers"
595 ? "If this is right, press START
{9 SPACES}To make changes, press
OPTION"
600 ON (PEEK(53279)=6)+(2*(PEEK(5327
9)=3)) GOTO 605,440:GOTO 600
605 A=PEEK(106):TOP=A-24:CHBAS=TOP:C
H=CHBAS*256:SP=TOP+8:SC=SP*256:P
OKE 106,TOP:OLDCHBAS=224:GRAPHIC
S 0
610 ? "Just a minute while I get mys
elf{6 SPACES}together . . ."
615 IF F$="@" THEN CHBAS=224:CH=CHBA
S*256:GOTO 630
620 OPEN #1,4,0,F$:POKE ICBADR+X+1,C
HBAS:POKE ICBADR+X,0:POKE ICBLEN
+X+1,4:POKE ICBLEN+X,0
625 POKE ICCOM+X,7:C=USR(ADR("hhh*LV
d"),X):CLOSE #1
630 POKE 559,0:GOSUB 640:GOSUB 655:G
OSUB 810:GOSUB 635:ON FLOAD GOSU
B 650:GOSUB 925:GOTO 120
635 POKE 756,CHBAS:RETURN
640 OPEN #1,4,0,"D1:CLEAR.SUB":FOR I
=1 TO 33:GET #1,N:CLEAR$(I,I)=CH
R$(N):NEXT I:CLOSE #1
645 C=USR(ADR(CLEAR$),SP,X):RETURN
650 T=SZ:FL$=FLOAD$:GOSUB 20:SZ=T:RE
TURN
655 DL=256*(TOP+4):DL4=DL+4:DL5=DL+5
:FOR I=0 TO 2:POKE DL+I,112:NEXT
 I:PIX=0:N=0
660 FOR I=DL+3 TO DL+27+36*(MODE<>5)
 STEP 3:C=SC+N*WIDE:POKE I,64+MO
DE:POKE I+2,INT(C/256)
665 POKE I+1,C-256*PEEK(I+2):N=N+1:N
EXT I
670 N=0:MENU=256*(TOP+5)+64:DLMEN=DL
+32+36*(MODE<>5):POKE DLMEN-2,MO
DE+64:POKE DLMEN,INT(MENU/256)
675 POKE DLMEN-1,MENU-256*PEEK(DLMEN
):FOR I=DLMEN+1 TO DLMEN+3:POKE
I,MODE:NEXT I
680 POKE I,65:POKE I+1,0:POKE I+2,DL
/256:OPEN #1,4,0,"D:DISPLAY.SUB"
685 DISPLAY=DL+128:TRAP 690:FOR I=0
TO 186:GET #1,N:POKE DISPLAY+I,N
:NEXT I:GOTO 695
690 POP
695 WHI=INT(WIDE/256):WLO=WIDE-256*W
HI:POKE DL+112,WLO:POKE DL+113,W
HI
700 POKE 560,0:POKE 561,DL/256:CLOSE
 #1:RETURN
705 POKE SC+PIX,OLD:GOSUB 40:POKE 75
6,OLDCHBAS:GRAPHICS 0:POKE 764,2
55
710 ? "Screen is saved as D1:TEMPFIL
E.SCR":? :? "Do you want to save
 the screen as":? FSAVE$;"? (Y o
r N)"
715 GOSUB 785:ON N<>43 AND N<>35 GOT
O 715:IF N=43 THEN GOSUB 765:GOT
O 725
720 FSAVE=0
725 ? :? "Do you want to quit? (Y or
 N)":POKE 764,255
730 GOSUB 785:ON N<>43 AND N<>35 GOT
O 730:ON N=35 GOTO 735:ON N=43 G
OTO 760
735 ? :? "To return to edit the same
 screen,{4 SPACES}press OPTION":
? :? "To start FONTBYTER over, p
ress START"
740 OPT=PEEK(53279):ON ((OPT=6)+(2*(
OPT=3))) GOTO 745,750:GOTO 740
745 POKE 106,A:GRAPHICS 0:GOTO 10
750 POKE 106,TOP:GOSUB 635:FL$="D1:T
EMPFILE.SCR":IF FSAVE=1 THEN FL$
=FSAVE$
755 GOSUB 20:GOSUB 655:GOTO 120
760 POKE 106,A:POKE 764,255:GRAPHICS
 0:END
765 FSAVE=1:TRAP 770:OPEN #2,4,0,FSA
VE$:CLOSE #2:XIO 36,#2,0,0,FSAVE
$:XIO 33,#2,0,0,FSAVE$:GOTO 775
770 CLOSE #2
775 FL$="D1:TEMPFILE.SCR,":FLL$=FSAV
E$(4,LEN(FSAVE$)):FL$(17)=FLL$
780 XIO 32,#1,0,0,FL$:RETURN
785 C=PEEK(764):N=C-64*INT(C/64):RET
URN
790 IF WIDE<40 THEN WIDE=40:RETURN
795 IF WIDE>170 AND MODE<>5 THEN WID
E=170:RETURN
800 IF WIDE<340 THEN RETURN
805 WIDE=340:RETURN
810 TRAP 815:OPEN #1,4,0,"D:DELETE.S
```

```
     UB":FOR I=1 TO 124:GET #1,N:DELE
     TE$(I,I)=CHR$(N):NEXT I:GOTO 820
815  POP
820  CLOSE #1:WHI=INT(WIDE/256):WLO=W
     IDE-256*WHI
825  TRAP 830:OPEN #1,4,0,"D:EXPAND.S
     UB":FOR I=1 TO 124:GET #1,N:EXPA
     ND$(I,I)=CHR$(N):NEXT I:GOTO 835
830  POP
835  CLOSE #1:RETURN
840  TRAP 865:XIO 36,#1,0,0,"D:*.SCR"
845  ? :? "Currently saved screen fil
     es:":FLL$="SCR":GOTO 860
850  TRAP 865:XIO 35,#1,0,0,"D:*.SET"
855  ? :? "Currently available charac
     ter sets:":FLL$="SET"
860  FL$="D1:*.":FL$(LEN(FL$)+1)=FLL$
     :OPEN #1,6,0,FL$:FOR I=0 TO 50:I
     NPUT #1,FLL$:? FLL$:NEXT I
865  CLOSE #1:RETURN
870  GOSUB 920:POKE SC+PIX,OLD:GOTO 8
     95
875  WV=2*((DI=5)+(DI=13)+(DI=9))+(DI
     =10)+(DI=6)+(DI=14):WH=2*(DI<8 A
     ND DI>4)+(DI<12 AND DI>8)
880  W=(PEEK(DL4)+256*PEEK(DL5))-SC:U
     =INT(W/WIDE):WV=WV-(U=0 AND WV=1
     )-2*((U+7+12*(MODE<>5)=LINE-2) A
     ND WV=2)
885  L=W-(U*WIDE):WH=WH-(L=0 AND WH=1
     )-2*((L+40)=WIDE AND WH=2)
890  POKE DL+114,WH:POKE DL+115,WV:C=
     USR(DISPLAY)
895  IF PEEK(53279)<>6 THEN DI=PEEK(6
     32):ON DI<>15 GOTO 875:GOTO 895
900  PIX=PEEK(DL4)+256*PEEK(DL5)+(6+6
     *(MODE<>5))*WIDE+20:OLD=PEEK(PIX
     ):PIX=PIX-SC:RETURN
905  OPEN #4,4,0,"D:CHARDATA.DAT"
910  FOR I=0 TO 255:GET #4,N:C(I)=N:N
     EXT I
915  CLOSE #4:RETURN
920  FOR I=0 TO 10:POKE 53279,4:NEXT
     I:RETURN
925  OPEN #1,4,0,"D:MENU.DAT":FOR I=4
      TO 483:GET #1,N:POKE MENU+I,N:N
     EXT I:CLOSE #1
930  MENSH=MENU+160*INT(SHIF/64):POKE
      DLMEN,INT(MENSH/256):POKE DLMEN
     -1,MENSH-256*PEEK(DLMEN):RETURN
```

Program 2: DISPLAY. SUB
Machine Language Scrolling Subroutine

```
900  OPEN #1,8,0,"D1:DISPLAY.SUB"
910  FOR I=1 TO 186:READ N:PUT #1,N:N
     EXT I:CLOSE #1:? I:END
1000 DATA 104,173,49,2,133,206,133,2
     13
1008 DATA 173,48,2,105,3,133,205,105
1016 DATA 109,133,212,162,0,161,205,
     41
1024 DATA 191,133,207,230,205,161,20
     5,133
1032 DATA 203,160,1,177,205,133,204,
     200
1040 DATA 177,212,240,34,201,2,208,1
     6
1048 DATA 24,165,203,105,1,133,203,1
     65
1056 DATA 204,105,0,133,204,24,144,1
     4
1064 DATA 56,165,203,233,1,133,203,1
     65
1072 DATA 204,233,0,133,204,24,160,3
1080 DATA 177,212,240,42,201,2,208,1
     9
1088 DATA 24,165,203,160,0,113,212,1
     33
1096 DATA 203,200,165,204,113,212,13
     3,204
1104 DATA 24,144,19,56,165,203,160,0
1112 DATA 241,212,133,203,165,204,20
     0,241
1120 DATA 212,133,204,24,144,0,160,8
1128 DATA 165,207,201,5,240,2,160,20
1136 DATA 162,0,165,203,129,205,230,
     205
1144 DATA 165,204,129,205,132,207,24
     ,165
1152 DATA 203,160,0,113,212,133,203,
     165
1160 DATA 204,200,113,212,133,204,23
     0,205
1168 DATA 230,205,164,207,136,208,21
     9,165
1176 DATA 203,129,205,230,205,165,20
     4,129
1184 DATA 205,96
```

Program 3: EXPAND. SUB
Machine Language Line Insert Subroutine

```
900  OPEN #1,8,0,"D1:EXPAND.SUB"
910  FOR I=1 TO 122:READ N:PUT #1,N:N
     EXT I:CLOSE #1:? I:END
1000 DATA 104,166,207,169,0,165,209,
     240
1008 DATA 29,160,255,177,203,145,205
     ,136
1016 DATA 208,249,230,204,230,206,16
     4,208
1024 DATA 177,203,145,205,136,208,24
     9,198
1032 DATA 204,198,206,24,144,9,164,2
     08
1040 DATA 177,203,145,205,136,208,24
     9,202
1048 DATA 240,29,56,165,205,229,208,
     133
1056 DATA 205,165,206,229,209,133,20
     6,56
1064 DATA 165,203,229,208,133,203,16
     5,204
1072 DATA 229,209,133,204,24,144,182
     ,165
1080 DATA 209,240,27,160,255,169,0,1
     45
1088 DATA 203,136,208,251,230,206,23
     0,204
1096 DATA 164,208,145,203,136,208,25
     1,198
1104 DATA 206,198,204,24,144,11,164,
     208
1112 DATA 240,7,169,0,145,203,136,20
     8
1120 DATA 251,96
```

Program 4: DELETE. SUB
Machine Language Line Delete Subroutine

```
900  OPEN #1,8,0,"D1:DELETE.SUB"
910  FOR I=1 TO 122:READ N:PUT #1,N:N
     EXT I:CLOSE #1:? I:END
1000 DATA 104,166,207,169,0,165,209,
     240
1008 DATA 29,160,255,177,205,145,203
     ,136
```

```
1016 DATA 208,249,230,204,230,206,16
     4,208
1024 DATA 177,205,145,203,136,208,24
     9,198
1032 DATA 204,198,206,24,144,9,164,2
     08
1040 DATA 177,205,145,203,136,208,24
     9,202
1048 DATA 240,29,24,165,205,101,208,
     133
1056 DATA 205,165,206,101,209,133,20
     6,24
1064 DATA 165,203,101,208,133,203,16
     5,204
1072 DATA 101,209,133,204,24,144,182
     ,165
1080 DATA 209,240,27,160,255,169,0,1
     45
1088 DATA 205,136,208,251,230,206,23
     0,204
1096 DATA 164,208,145,205,136,208,25
     1,198
1104 DATA 206,198,204,24,144,11,164,
     208
1112 DATA 240,7,169,0,145,205,136,20
     8
1120 DATA 251,96
```

Program 5: MENU. DAT

```
900 OPEN #1,8,0,"D1:MENU.DAT"
910 FOR I=1 TO 482:READ N:PUT #1,N:N
    EXT I:CLOSE #1:? I:END
1000 DATA 0,0,91,0,17,0,18,0
1008 DATA 19,0,20,0,21,0,22,0
1016 DATA 23,0,24,0,25,0,16,0
1024 DATA 28,0,30,0,126,0,0,0
1032 DATA 0,0,0,0,0,0,0,0
1040 DATA 0,0,0,127,0,113,0,119
1048 DATA 0,101,0,114,0,116,0,121
1056 DATA 0,117,0,105,0,111,0,112
1064 DATA 0,13,0,29,0,0,0,0
1072 DATA 0,0,0,0,0,0,0,0
1080 DATA 0,0,0,0,0,0,97,0
1088 DATA 115,0,100,0,102,0,103,0
1096 DATA 104,0,106,0,107,0,108,0
1104 DATA 27,0,11,0,10,0,0,0
1112 DATA 0,0,0,0,0,0,0,0
1120 DATA 0,0,0,0,0,0,0,122
1128 DATA 0,120,0,99,0,118,0,98
1136 DATA 0,110,0,109,0,12,0,14
1144 DATA 0,15,0,0,0,0,0,0
1152 DATA 0,0,0,0,0,0,0,0
1160 DATA 0,0,0,0,1,0,2,0
1168 DATA 3,0,4,0,5,0,6,0
1176 DATA 7,0,32,0,8,0,9,0
1184 DATA 125,0,0,0,0,0,0,0
1192 DATA 0,0,0,0,0,0,0,0
1200 DATA 0,0,0,0,0,49,0,55
1208 DATA 0,37,0,50,0,52,0,57
1216 DATA 0,53,0,41,0,47,0,48
1224 DATA 0,63,0,124,0,0,0,0
1232 DATA 0,0,0,0,0,0,0,0
1240 DATA 0,0,0,0,0,0,33,0
1248 DATA 51,0,36,0,38,0,39,0
1256 DATA 40,0,42,0,43,0,44,0
1264 DATA 26,0,60,0,62,0,0,0
1272 DATA 0,0,0,0,0,0,0,0
1280 DATA 0,0,0,0,0,0,0,58
1288 DATA 0,56,0,35,0,54,0,34
1296 DATA 0,46,0,45,0,59,0,61
1304 DATA 0,31,0,0,0,0,0,0
1312 DATA 0,0,0,0,0,0,0,0
1320 DATA 0,0,0,0,0,0,0,0
1328 DATA 0,0,0,0,0,0,0,0
1336 DATA 0,0,0,0,0,0,0,0
1344 DATA 0,0,0,0,0,0,0,0
1352 DATA 0,0,0,0,0,0,0,0
1360 DATA 0,0,0,0,0,81,0,87
1368 DATA 0,69,0,82,0,84,0,89
1376 DATA 0,85,0,73,0,79,0,80
1384 DATA 0,92,0,93,0,0,0,0
1392 DATA 0,0,0,0,0,0,0,0
1400 DATA 0,0,0,0,0,0,65,0
1408 DATA 83,0,68,0,70,0,71,0
1416 DATA 72,0,74,0,75,0,76,0
1424 DATA 123,0,94,0,95,0,0,0
1432 DATA 0,0,0,0,0,0,0,0
1440 DATA 0,0,0,0,0,0,0,90
1448 DATA 0,88,0,67,0,86,0,66
1456 DATA 0,78,0,77,0,64,0,96
1464 DATA 0,0,0,0,0,0,0,0
1472 DATA 0,0,0,0,0,0,0,0
1480 DATA 0,0
```

Program 6: CHARDATA. DAT

```
900 OPEN #1,8,0,"D1:CHARDATA.DAT"
910 FOR I=1 TO 256:READ N:PUT #1,N:N
    EXT I:CLOSE #1:? I:END
1000 DATA 108,106,27,0,0,107,11,10
1008 DATA 111,0,112,117,0,105,13,29
1016 DATA 118,0,99,0,0,98,120,122
1024 DATA 20,0,19,22,91,21,18,17
1032 DATA 12,0,14,110,0,109,15,0
1040 DATA 114,0,101,121,127,116,119,
     113
1048 DATA 25,0,16,23,126,24,28,30
1056 DATA 102,104,100,0,0,103,115,97
1064 DATA 44,42,26,0,0,43,60,62
1072 DATA 47,0,48,53,0,41,63,124
1080 DATA 54,0,35,0,0,34,56,58
1088 DATA 4,0,3,6,0,5,2,1
1096 DATA 59,0,61,46,0,45,31,0
1104 DATA 50,0,37,57,0,52,55,49
1112 DATA 8,0,9,7,0,32,125,0
1120 DATA 38,40,36,0,0,39,51,33
1128 DATA 76,74,123,0,0,75,94,95
1136 DATA 79,0,80,85,0,73,92,93
1144 DATA 86,0,67,0,0,66,88,90
1152 DATA 0,0,0,0,0,0,0,0
1160 DATA 64,0,96,78,0,77,0,0
1168 DATA 82,0,69,89,0,84,87,81
1176 DATA 0,0,0,0,0,0,0,0
1184 DATA 70,72,68,0,0,71,83,65
1192 DATA 0,0,0,0,0,0,0,0
1200 DATA 0,0,0,0,0,0,0,0
1208 DATA 0,0,0,0,0,0,0,0
1216 DATA 0,0,0,0,0,0,0,0
1224 DATA 0,0,0,0,0,0,0,0
1232 DATA 0,0,0,0,0,0,0,0
1240 DATA 0,0,0,0,0,0,0,0
1248 DATA 0,0,0,0,0,0,0,0
```

Program 7: CLEAR. SUB

Machine Language Screen Clear Subroutine

```
900 OPEN #1,8,0,"D1:CLEAR.SUB"
910 FOR I=1 TO 33:READ N:PUT #1,N:NE
    XT I:CLOSE #1:? I:END
1000 DATA 104,104,104,133,208,104,10
     4,101
1008 DATA 208,133,209,169,0,133,207,
     160
1016 DATA 255,145,207,136,208,251,14
     5,207
1024 DATA 230,208,165,208,197,209,20
     8,235
1032 DATA 96
```

©

Timex/Sinclair
Making Change

Michael B. Williams

This game is an excellent educational tool for children and is based on a previously published COMPUTE! *article. The author also includes conversion tips for T/S users who want to translate programs originally written for other computers.*

Converting a program written for one computer to another computer can get difficult if the program contains machine-dependent features (graphics commands, for instance) or a lot of POKEs. If a program has many such features, it would probably be easier to rewrite it from scratch, once you understand it.

With modification, many programs published in COMPUTE! and other computer magazines can be converted to run on the Timex/Sinclair. One of these is "Making Change," by Myron Miller (COMPUTE!, February 1983), a program written to help children learn how to count money, divide money, and make change.

Program Conversion

When you transfer any program, your conversion should make up for deficiencies in one area by compensating for them in another area. If your computer has limited graphics, liven it up with sound, and vice versa. The Sinclair has no sound, no color, and limited graphics. But that does not mean we cannot make the program interesting for the user.

First of all, I decided which version of "Making Change" to convert. I chose the Atari version and went to work. I made multiple statements into individual lines and added STR$ when printing numbers so there would be no pause before they were displayed. Congratulating myself on my persistence, I soon found I wasn't done yet – I had the huge task of debugging ahead. I eliminated one bug only to find several more.

Finally, I concluded it would be easier to rewrite the program using the listing as a guide. In doing so, I just went about program conversion in a different way. Instead of interpreting the listing line by line, I first tried to understand what it did as a whole. The result was a bug-free program that made it fun to learn about money.

Each problem is a question about money: how to count it, how to divide it – in short, how to make change. It addresses the child by name and asks him or her directly. If the child answers incorrectly, the program encourages the child to try once more, and the question is repeated. By the third attempt, if the child has not correctly solved the problem, he or she is given the answer.

A correct answer causes the program to call the machine language routine in line 1. This routine is very important to the program, but you can modify it as you wish.

Regardless of whether or not the question was answered correctly, the child is given the option of receiving his or her score, continuing, or stopping the program. In this way the child's progress may be evaluated at any stage of the program.

Special Program Notes

Program 1 POKEs a machine language routine into the REM statement in line 1. After typing in Program 1, proofread it carefully. Type RUN and ENTER. Test it by using RAND USR 16514. The screen should fill up very fast with inverse spaces. List Program 1 and note that the REM statement in line 1 has been altered. It now contains a machine language subroutine.

Now, delete lines 10-60 and type in Program 2 with line 1 still in memory. When typing in Program 2, use the following instructions where graphics characters appear:

60: Graphics, SHIFT-S, and SHIFT-D
5010: Underlined letters are inverse video
5115: Graphics and SHIFT-D

Program 1: Machine Language Loader

```
1 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
10 LET ZZ=16514
20 LET A$="2A0C40017618237EB92604C6807704
   10F5C9"
30 POKE ZZ,16*CODE A$+CODE A$(2)-476
40 LET ZZ=ZZ+1
50 LET A$=A$(3 TO )
60 GOTO 30
```

Program 2: Making Change

```
50 SLOW
60 PRINT "██████████████████████████████
   █           MAKING CHANGE           █
   ██████████████████████████████"
70 PRINT ,,"     TIMEX/SINCLAIR VERSION
   "
80 PRINT ,,"HELLO. PLEASE TELL ME YOUR N
   AME."
90 INPUT N$
100 GOSUB 2000
1000 REM PROBLEMS
1010 REM
1020 LET TP=TP+1
1030 LET CT=INT (100*RND)+1
1040 LET TR=0
1050 LET PT=NOT PT
1060 LET QU=INT (CT/25)
1070 LET DI=INT ((CT-QU*25)/10)
1080 LET NI=INT ((CT-QU*25-DI*10)/5)
1090 LET PE=INT CT-QU*25-DI*10-NI*5
1100 GOTO PT*1000+3000
2000 REM VARIABLES
2010 LET TP=0
2020 LET TC=0
2030 LET TW=0
2040 LET ME=0
2050 LET ML=0
2060 LET FI=0
2070 LET PT=0
2080 LET SCORE=5000
2100 DIM R$(5,10)
2110 FOR X=1 TO 5
2120 LET R$(X)="FANTASTIC.GREAT.    RIGH
     T.    TERRIFIC. VERY GOOD."(X*10-9
     TO X*10)
2130 NEXT X
2999 RETURN
3000 REM COUNT CHANGE
3010 CLS
3020 PRINT TP;") ";N$;", IF I HAVE..."
3040 PRINT ,,TAB 5;QU;" QUARTER"+("S" AN
     D QU<>1)
3050 PRINT ,,TAB 5;DI;" DIME"+("S" AND D
     I<>1)
3060 PRINT ,,TAB 5;NI;" NICKEL"+("S" AND
      NI<>1);", AND "
3070 PRINT ,,TAB 5;PE;" PENN"+("Y" AND P
     E=1)+("IES" AND PE<>1);", THEN"
3080 PRINT ,,"HOW MUCH CHANGE DO I HAVE?
      ";
3110 INPUT CH
3120 PRINT CH
3130 LET TR=TR+1
3140 IF CH=CT THEN GOTO 3500
3150 PRINT ,,"NOPE. THAT'S "+STR$ ABS (C
     T-CH)+" CENT"+("S" AND ABS (CH-CT)<
     >1)+" TOO "+("MUCH." AND CH>CT)+("L
     ITTLE." AND CH<CT)
3160 LET ML=ML+1
3165 IT TR=1 THEN LET TW=TW+1
3170 IF TR=3 THEN GOTO 3210
3180 PRINT ,,"PRESS ENTER TO TRY AGAIN."
3190 IF INKEY$="" THEN GOTO 3190
3200 GOTO 3000
3210 PRINT ,,"THE ANSWER IS "+STR$ CT+",
      "+N$+"."
3220 GOTO SCORE
3500 LET ZZ=USR 16514
3510 PRINT
3520 PRINT N$+", "+"THATS "+R$(INT (5*RN
     D)+1)
3530 LET ZZ=USR 16514
3540 IF TR=1 THEN LET TC=TC+1
3550 LET ME=ME+1
3560 GOTO SCORE
4000 REM GIVE CHANGE
4010 CLS
4020 PRINT STR$ TP+") "+N$+", IF I HAVE
     "+STR$ CT+" CENT"+("5" AND CT<>1)+"
     ,"
4030 PRINT ,,"HOW MANY QUARTERS DO I HAV
     E? ";
4040 INPUT Q
4050 PRINT Q,,TAB 9;"DIMES? ";
4060 INPUT D
4070 PRINT D,,TAB 9;"NICKELS? ";
4080 INPUT N
4090 PRINT N,,TAB 9;"PENNIES? ";
4100 INPUT P
4110 PRINT P
4115 LET TR=TR+1
4120 IF Q=INT Q AND N=INT N AND D=INT D
     AND P=INT P THEN GOTO 4200
4130 PRINT ,,"THATS NOT FAIR, USING DECI
     MALS. YOU MUST PAY A FINE, "+N$+"."
4140 LET FI=FI+1
4150 GOTO 4220
4200 IF 25*Q+10*D+5*N+P=CT THEN GOTO 440
     0
4205 PRINT ,,"NOT QUITE--THAT MAKES "+ST
     R$ (25*Q+10*D+5*N+P)+" CENTS."
4210 LET ML=ML+1
4215 IF TR=1 THEN LET TW=TW+1
4220 IF TR=3 THEN GOTO 4300
4230 PRINT ,,"PRESS ENTER TO TRY ONCE MO
     RE."
4240 IF INKEY$="" THEN GOTO 4240
4250 GOTO 4000
4300 PRINT ,,"A THOUSAND TIMES NO, "+N$+
     "."
4310 PRINT ,,"I WOULD HAVE "+STR$ QU;" Q
     UARTER"+("S" AND QU<>1);TAB13;STR$
     DI+" DIME"+("S" AND DI<>1)
4320 PRINT TAB 13;STR$ NI+" NICKEL"+("S"
      AND NI<>1);", AND ";TAB 13;STR$ PE
     +" PENN"+("Y" AND PE=1)+("IES" AND
     PE<>1);"."
4330 GOTO SCORE
4400 IF Q=QU AND D=DI AND N=NI AND P=PE
     THEN GOTO 4500
4410 PRINT ,,"TRUE, BUT YOU COULD HAVE U
     SED   FEWER COINS, "+N$+"."
4420 GOTO 4210
4500 LET ZZ=USR 16514
4510 PRINT
4520 PRINT "HEY..."+R$(INT (5*RND)+1)
4530 LET ZZ=USR 16514
4540 LET ME=ME+1
```

```
4550 IF TR=1 THEN LET TC=TC+1
5000 REM SCORE
5010 PRINT ,,"PRESS C TO CONTINUE, S FOR
      YOUR SCORE, OR ENTER TO STOP."
5020 LET I$=INKEY$
5030 IF I$="" THEN GOTO 5020
5040 IF I$="C" THEN GOTO 1000
5050 IF I$<>"S" AND I$<>CHR$ 118 THEN GO
      TO 5020
5100 CLS
5110 PRINT N$+"S "+("FINAL " AND I$=CHR$
      118)+"SCORE:"
5115 PRINT "████████████████████████████
      ████"
5120 PRINT ,,"NUMBER OF PROBLEMS: "+STR$
      TP
5130 PRINT ,,"TOTAL CORRECT: "+STR$ TC
5140 PRINT ,,"TOTAL WRONG: "+STR$ TW
5150 PRINT ,,"PERCENT CORRECT: "+STR$ IN
     T ((TC/TP)*100)
5160 PRINT ,,,,"MONEY EARNED: "+STR$ ME
5170 PRINT ,,"MONEY LOST: "+STR$ ML
5180 PRINT ,,"FINES: "+STR$ FI
5190 LET X=ME-ML-FI
5200 PRINT ,,("I OWE YOU" AND X>0)+("YOU
      OWE ME" AND X<0)+" "+STR$ ABS X+"
     CENT"+("S" AND ABS X<>1);"."
5210 IF I$=CHR$ 118 THEN GOTO 5250
5220 PRINT ,,"PRESS ENTER TO CONTINUE "
5230 IF INKEY$="" THEN GOTO 5230
5240 GOTO 1000
5250 PRINT ,,"BYE, "+N$+". I HAD FUN."
9997 STOP
9998 SAVE "MAKING CHANGE"
9999 RUN
```

©

Relative Files For VIC-20 And Commodore 64
Part I

Jim Butterfield, Associate Editor

You can use relative files with your VIC or 64 computer and 1540/1541 disk drive. If you have the appropriate IEEE interface, you can do the same job on the 4040, 8050, or other recent Commodore disk units. It takes a little more work, and careful programming. But it can be done.

All the examples given here will work on all PETs and CBMs. On 4.0 BASIC, there are easier ways, but this will work.

Binary Numbers: High And Low

We'll be talking about some numbers packed into ASCII characters. In the expression CHR$(N), we can't use a value of N greater than 255. Sometimes we will want to send larger numbers. For example, if we want to select record number 1000 of a relative file, we'll need to split it into two parts. The "high" part would be the number divided by 256; the low part would be the remainder. So a value of 1000 would split up into a high part of 3 and a low part of 232, since 1000 divided by 256 gives 3 with 232 remainder.

When we send a number this way, we almost always send the low part first. So to send 1000, we'll eventually send to the disk:

CHR$(232);CHR$(3).

In Part II, we'll indicate a number that is split in this manner with the terms "High" and "Low."

Creating A Relative File

Decide how long you want a record to be. For example, you might have a file that will contain a name, a set of initials, and a date. You could allow 15 characters for the name; two characters for the initials; and seven characters for the date. Additionally, you'd need two extra characters as "separators" between the three data fields, giving a total of up to 26 characters in a record. You can go as high as 254, but no higher.

When we create a relative file, we must give the record length. After it is created, we don't need to specify the length: the disk will remember.

Let's open a relative file using direct statements. You can do this in a program, of course; but you may find it interesting to see things happening. First, however, we'll set up a program to allow us to check for errors on the command channel. Enter this program:

```
100 INPUT#15,E,E$,E1,E2
110 PRINT E;E$;E1,E2
```

Now type, as a direct command: OPEN 15,8,15. This will open the command channel for us. Anytime we want to look at a disk error condition, we can type GOTO 100, and the error will be printed.

We're ready to open our relative file. Type:

```
OPEN 1,8,2,"RANDFIL,L,"+CHR$(26)
```

That does the job. The name of the new file is RANDFIL. The L stands for length: don't forget to put a comma both before and after. Finally, the CHR$(26) gives the length of our record. We don't need to use all 26 characters, but we must not exceed this value when we write a record.

Positioning To A Specific Record

We've created the file, but we have not written any records yet. It's a good idea to bring enough records into existence to fill more than one disk sector, which takes up 254 bytes. In the case of 26-character records, this means that we should create at least ten records.

We could do this with ten PRINT#1 statements, but I'd like to show you another way. Let's position directly to record number 10 and write something there. Automatically, all missing records (in this case, 1 to 9) will come into existence. So we'd better learn how to position a relative file.

Now, we send our "position" command down the command channel. To identify to the disk which file we want to position, we use the secondary address. For our relative file in progress, that would be 2. That's important: secondary address, not logical file number. Now, another thing about the disk: it likes to see you add 96 to the secondary address, so we should send 98.

We have said that we want to go to record number 10. We must split this number up into high and low byte: we get 0 high and 10 low. Finally, we want to choose the start of the record, or position 1. Let's put it all together and type in:

```
PRINT#15,"P"+CHR$(96+2)+CHR$(10)+CHR$
  (0)+CHR$(1)
```

You'll see the disk error light go on – we'll account for this in a moment. To review: P for position; 96+2 for secondary address 2; 10 and 0 for record number 10; and 1 for the start of the record.

Why did the error light go on? Because there is no record number ten – yet. You may type GOTO 100 and look at the error notice: you'll see RECORD NOT PRESENT, which makes sense. The moment we write something, we'll bring this missing record into existence. Let's do that:

```
PRINT#1,"DOAKES"+CHR$(13)+"J"+CHR$(13)+
  "AUG15";
```

We have just written record number ten. Note that we are using a Return character to separate the fields (name, initial, date), and note that the PRINT statement ends with a semicolon. This seems puzzling: it doesn't work that way on sequential files. Let's give the golden rule for writing relative files:

Rule: One PRINT statement writes one and only one record to a relative file.

So the semicolon at the end does not change anything: we've written a complete record. And the Return characters in the middle do not change anything: we've written only one record.

Let's tie up this file for the moment. Close it with:

```
CLOSE 1
```

No need to close the command channel.

After Creation

Let's write a program to read and write this little file that we have created. Here we go:

```
100 OPEN 4,8,5,"RANDFIL"
```

I've changed the logical file number and secondary address just to prove that it doesn't matter. Note that we don't need to specify the length, once the file exists.

```
110 OPEN 15,8,15
```

Now for the main user interface. We'll ask for a record number, and quit if the user types zero:

```
200 INPUT"RECORD NUMBER";R
210 IF R<1 GOTO 600
```

Let's position to the record:

```
220 R0=INT(R/256):R1=R-R0*256
```

R0 is the high part of the number, and R1 is low. Now we can position:

```
230 PRINT#15,"P"+CHR$(101)+CHR$(R1)+CHR$
    (R0)+CHR$(1)
```

We remember that 101 is 96 plus 5. Let's look for an error:

```
240 INPUT#15,E,E$,E1,E1
250 IF E<>0 THEN PRINT E$:GOTO 200
```

We've positioned at a valid record. Now let's ask if we want to read or write:

```
300 INPUT"R OR W";C$
310 IF C$="R" GOTO 400
320 IF C$="W" GOTO 500
330 GOTO 200
```

Now for the reading part. We are already positioned, but first we must learn another golden rule, this one for reading records:

Rule: Variable ST signals end of record with value 64.

This, too, is different from sequential files, where ST signals end-of-file. Now we can read however many fields are in the record, since we'll detect the end of record in ST:

```
400 F=0
410 F=F+1:INPUT#4,F$
420 PRINT "FIELD";F;":";F$
430 IF ST=0 GOTO 410
440 GOTO 200
```

Thus, we keep reading until we have gathered all the fields within the record. Now, for the writing part. We've been here before:

```
500 FOR F=1 TO 3
510 PRINT"FIELD";F;
520 INPUT F$(F)
530 NEXT F
540 PRINT#4,F$(1)+CHR$(13)+F$(2)+CHR$(13)+
    F$(3);
550 GOTO 200
```

That's it except for quitting. We must remember to close our file:

```
600 CLOSE 4
```

Try playing with this program. You might like to try for nonexistent records, or writing records that are too large to fit. You'll quickly see how it all works. Note the curiosity: the character "pi," or CHR$(255), is stored in unused records.

It's not too hard and can be very useful. With relative records, you can go directly to any chosen part of your file. You can read or write as desired.

It's another tool for effective use of your computer.

Maximum Memory=Maximum Application!

Maximum Memory allows you to use more powerful programs for:

● EDUCATION ● ENTERTAINMENT ● MAIL LISTS
● BUSINESS APPLICATIONS ● FINANCIAL RECORDS

APROPOS TECHNOLOGY

RAMAX™

You get up to a full 32k RAM use with your VIC-20!

A perfect investment to give your family and yourself more enjoyment and use from your home computer! The ease of operation, the neat appearance, and the real POWER it adds to your VIC at this low price makes it a MUST for every VIC home!

SPECIAL
Only $124.95

Price includes shipping and handling within Continental USA. Foreign orders please add $15.00. Calif. Residents add 6% sales tax.

10 DAY MONEY-BACK GUARANTEE.
If not satisfied, simply return in original condition for your money back.

NEW! RAMAX JR.™

Already own an 8k Expander? Get the NEW **RAMAX JR.™**! Identical to the RAMAX™ except with 19k instead of 27k. Our instructions will show you how to use your 8k as BLK 3 with JR. to get the full compliment of memory!

SPECIAL $109.95 Shipping Included

To equal the total memory of RAMAX™ you would have to buy Commodore's 16k Memory Expansion, PLUS their 8k Expansion, PLUS their 3k Expansion for a total list price of over $200.00! THEN you would need a "mother board" which would run the total up to OVER $400! With RAMAX™ you buy just ONE piece...at ABOUT HALF THE PRICE!

RAMAX™ Features and Specifications:

● Adds up to a full 27k bytes of additional RAM to the standard VIC-20's internal RAM.

● Built-in switch allows User selection of any combination of 5 areas of RAM memory*:
BLK 1 (8k:Adr.8192-16383)
BLK 2 (8k:Adr.16383-24575)
BLK 3 (8k:Adr.24576-32767)
BLK 5 (Adr.40960-49152, allows/disallows 8k ROM games)
RAM (3k:Adr.1024-4095)
RESET (Resets computer without power off/on)

● Built-in electrical Fuse to protect equipment.

● Totally self-contained. No external power supply needed.

● Two (2) extension connectors allow ANY additional cartridges and/or devices designed for the VIC expansion port.

● Very low power consumption (.175 amp usual).

● High reliability gold-plated connectors are designed for long life.

● Complete Operating Manual

● 6 month parts and labor warranty to original purchaser.

● Factory service.

*Many VIC-20 cartridges and programs require certain configurations of the memory (i.e. certain games will only run on the unexpanded VIC while others require the upper portion of the expanded memory). With RAMAX® you have switches that turn-on and turn-off portions of the memory to provide the right area of memory - all without plugging or unplugging. It's so easy!

Contact Your Local Dealer.
Or Send Check or Money Order For the Total
Calif. residents add 6% tax.

Phone orders: Call **(805) 482-3604**
Foreign orders, add $15.00
All items shipped from stock.
DEALER INQUIRIES WELCOME

WE SERVICE WHAT WE SELL!

VIC-20 & Commodore-64 are registered trademarks of Commodore International.

350 N. Lantana Ave., Suite 821
Camarillo, CA 93010

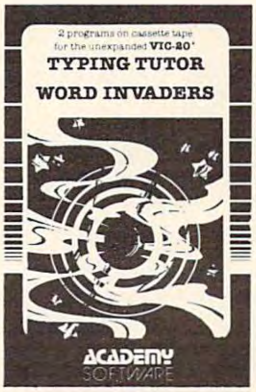
New Product!
APROSPAND-64™ Gives your Commodore 64 full expandability! This superbly designed expansion module plugs into the 64 & gives you 4 switchable (singly or in any combination) expansion connectors - plus fuse protection - plus a reset button! only **$69.95**
Shipping Included

APROPOS TECHNOLOGY

FOR THE VIC 20™

JOIN THE COMPUTER REVOLUTION WITH A MASTERY OF THE KEYBOARD!

In the age of the computer, *everyone* from the school child to the Chairman of the Board should be at home at the computer keyboard. Soon there will be a computer terminal on every desk and in every home. Learn how to use it right ...and have some fun at the same time!



Rated THE BEST educational program for the VIC 20™ by Creative Computing Magazine

TYPING TUTOR PLUS WORD INVADERS — $21.95

(2 programs on one cassette tape for the unexpanded VIC 20™)

Typing Tutor plus Word Invaders makes learning the keyboard easy and fun! Typing Tutor teaches the keyboard in easy steps. Word Invaders makes typing practice an entertaining game. Highly praised by customers:

"Typing Tutor is great!", "Fantastic", "Excellent", High quality", "A source of great joy and learning for our children", "Even my little sister likes it", "Word Invaders is sensational!"

Customer comment says it all . . .

". . . and it was everything you advertised it would be. In three weeks, my 13 year old son, who had never typed before, was typing 35 w.p.m. I had improved my typing speed 15 w.p.m. and my husband was able to keep up with his college typing class by practicing at home."

FOR THE COMMODORE 64™

SPECIAL VERSION OF TYPING TUTOR PLUS WORD INVADERS NOW AVAILABLE FOR THE COMMODORE 64™...

$21.95 *(Tape)*
$24.95 *(Disk)*

All of the features of the VIC 20™ Version and more

SPECIAL!

SPRITE DESIGNER

by Dr. Lee T. Hill

$16.95 *(Tape)* **$21.95** *(Disk)*

Create and then transform sprites automatically. We have the other sprite making programs, but this is the one *we use* to make sprites. The automatic transformations are great!

VISA

Shipping and handling $1.00 per order. California residents add 6% sales tax. VISA and Mastercard orders must include full name as shown on card, card number, and expiration date. Free catalog sent with order and on request.

MasterCard

ACADEMY SOFTWARE

P.O. Box 9403, San Rafael, CA 94912 (415) 499-0850

Programmers: Write to our New Program Manager concerning any exceptional VIC 20TM or Commodore 64TM game or other program you have developed.

Sprite Editor For TI

Larry Long

Here's a way to get maximum use of sprites on the TI-99/4A – and a program that generates listings for your sprite creations.

A very powerful yet often unused feature of the TI-99/4A is its ability to display and control sprites. With the 99/4A and the Extended BASIC Module, it is possible to generate 28 sprites for display and independent simultaneous movement. Program 1 should convince any doubters that this can be done. Although a lot of colored letters floating around the screen are a bit pointless, if we can modify and control the sprites, we will have a most useful feature.

Sprites can be designed by drawing on a piece of graph paper and then converting the on/off pixels to a hexadecimal number. If the two largest sizes of sprites are used, the hexadecimal number describing the shape of the sprite would be 64 characters long. A solution is a sprite editor that will allow us to draw the pattern we want on the screen and then have the computer create the program we need to make that sprite pattern. Program 2 will do exactly that, and more. It will allow us to edit the sprite pattern. Then, when we press the L key, it will display a complete listing that would, if copied on paper and then entered into the computer, provide a sprite and the necessary routine to control its movement.

Your Options

When you run the program, the first display screen will be a design grid with a box-shaped cursor. The area under the cursor will initially be white (signifying an "off" pixel). Press 1 to change the color beneath the cursor to black (representing an "on" pixel) or to move the cursor about the grid with the arrow keys. To turn off a particular pixel, press 0 and the background color will be returned to white. When you have completed your design, press the P key to see it displayed as a sprite.

At this point, you are given several options. You can magnify your newly constructed sprite (M key), change its color (C key), change its background color (B key), or set it in motion (E, S, D, X keys). If you are not pleased with the sprite's shape, you can modify it by striking the T key or (if the changes required are quite drastic) simply press the A key to start with a fresh grid. On the other hand, if you are satisfied with your sprite and its color and directional parameters, press the L key to create the BASIC statements needed to achieve these effects.

If using the sprite editor is your only concern, then skip the rest of this article and go straight to Program 2 and enjoy this easy access to sprites.

How The Editor Works

To understand what makes the editor work, let's take a general overview of the program:

Lines	
100-260	Set up screen display.
270-460	Are the main loop of the designing portion of the program.
470-680	Evaluate the design, put its values in an array, read the values in the array, convert them to hexadecimal numbers, and then build a 64-character string to describe the sprite pattern.
690-770	Put the sprite on the screen and display new program instructions.
780-930	Main loop of the implementation portion of the program.
940-980	Change size of sprite.
1000-1150	Display a listing of the sprite program.
1160-1220	Change the color of the sprite and screen.

A cursor is needed to indicate where you are located on the design grid. I chose to use a sprite (line 220) because I could move it around freely without disturbing the display under it. Repositioning the cursor is accomplished in line 380 with a CALL LOCATE. The arrow keys reposition the cursor, and the ENTER key changes the area under the cursor.

What makes "Sprite Editor" so valuable is its ability to generate the hexadecimal pattern for the sprite. The loop from line 500 through line 560 determines the character in each position of the design grid and stores that value in the array B (R,C). Line 570 provides a string with all of the possible hexadecimal digits placed in ascending order. Line 580 sets M$ to be "null." The loop from line 590 to line 630 evaluates the array elements and converts each row in the left half of the design grid to a pair of hexadecimal digits and concatenates them to M$. Line 620 is probably the most significant line in this loop, as it provides the hexadecimal numbers. It causes the computer to look at a particular digit (element) in HEX$ determined by the values calculated for HIGH and LOW. Lines 630-680 perform the same operation as 590-630, only for the right half of the design grid.

Line 690 assigns the hexadecimal numbers to ASCII characters 104, 105, 106, and 107. It is necessary to specify only the first character number in the CALL CHAR statement. When this feature is used, it is required that you start with a character that is evenly divisible by 4. Line 730 actually displays the sprite.

Lines 740-770 provide instructions for the implementation portion of the program. Lines 780-830 check for specific key presses and provide appropriate branching to list the program; end the program; start from the beginning; change the background color; modify the existing sprite; change sprite size; or change sprite color. Lines 840-920 check for arrow key presses and then increment or decrement sprite speed.

Lines 940-980 change sprite size. Lines 1000-1150 display a program listing that would generate a sprite like the one designed by the Sprite Editor. One problem with listing the program is displaying the quote character. The computer interprets it to mean that you want to end the PRINT statement. The solution is to redefine an unused character (I chose the lowercase "n") to look like the quote character.

Finally, lines 1160-1220 allow you to change the color of the sprite and screen.

Program 1: Sprite Generation

```
100 CALL MAGNIFY(2)::FOR X=1 TO 28::
    CALL SPRITE(#X,64+X,X/2,96,128,I
    NT(RND*100)-50,INT(RND*100)-50):
    :NEXT X::GOTO 100
```

Program 2: Sprite Editor

```
100 REM SPRITE EDITOR
110 DIM B(16,16):: SC=1
130 C1=7
140 CALL CHAR(100,"")
150 CALL CHAR(101,"FFFFFFFFFFFFFFFF
    ")
160 CALL CHAR(102,"FFFFC3C3C3C3FFFF
    ")
170 CALL COLOR(9,2,16)
180 CALL CLEAR
190 DISPLAY AT(1,10):"SPRITE EDITOR"
200 FOR R=1 TO 16 :: CALL HCHAR(4+R
    ,2,100,16):: NEXT R
210 CALL MAGNIFY(1)
212 IF K=84 THEN GOTO 217
215 CALL SCREEN(8)
217 CALL DELSPRITE(ALL)
220 CALL SPRITE(#28,102,14,32,8)
225 CALL HCHAR(21,1,32,31):: CALL H
    CHAR(22,1,32,31)
230 DISPLAY AT(22,2):"E=UP X=DOWN S
    =LEFT D=RIGHT"
240 DISPLAY AT(23,2):"PRESS 1 - PIX
    EL ON ,0 - OFF"
250 DISPLAY AT(24,2):"PRESS P TO DI
    SPLAY SPRITE"
260 R=1 :: C=1
270 CALL KEY(0,K,S)
271 IF S=0 THEN 270
272 IF K=48 THEN KHAR=100
274 IF K=49 THEN KHAR=101
280 IF K=83 THEN C=C-1 :: GOTO 320
290 IF K=68 THEN C=C+1 :: GOTO 320
300 IF K=69 THEN R=R-1 :: GOTO 320
310 IF K=88 THEN R=R+1 :: GOTO 320
312 IF K=80 THEN 470
320 IF C<1 THEN C=16
330 IF C>16 THEN C=1
340 IF R<1 THEN R=16
350 IF R>16 THEN R=1
380 CALL LOCATE(#28,(8*R)+25,8*C+1)
420 CALL HCHAR(4+R,1+C,KHAR)
430 CALL SOUND(20,200,5)
460 GOTO 270
470 CALL DELSPRITE(ALL)
480 CALL HCHAR(21,1,32,128)
490 DISPLAY AT(22,2):"PLEASE WAIT W
    HILE I THINK."
500 FOR R=1 TO 16
510 FOR C=1 TO 16
520 CALL GCHAR(4+R,1+C,GC)
530 GC=GC-100
540 B(R,C)=GC
550 NEXT C
560 NEXT R
570 HEX$="0123456789ABCDEF"
580 M$=""
590 FOR R=1 TO 16
600 LOW=B(R,5)*8+B(R,6)*4+B(R,7)*2+
    B(R,8)+1
610 HIGH=B(R,1)*8+B(R,2)*4+B(R,3)*2
    +B(R,4)+1
620 M$=M$&SEG$(HEX$,HIGH,1)&SEG$(HE
    X$,LOW,1)
630 NEXT R
640 FOR R=1 TO 16
650 LOW=B(R,13)*8+B(R,14)*4+B(R,15)
    *2+B(R,16)+1
660 HIGH=B(R,9)*8+B(R,10)*4+B(R,11)
    *2+B(R,12)+1
670 M$=M$&SEG$(HEX$,HIGH,1)&SEG$(HE
    X$,LOW,1)
680 NEXT R
690 CALL CHAR(104,M$)
700 CALL MAGNIFY(3)
710 MM=3
720 M=4
730 CALL SPRITE(#1,104,C1,50,170,0,
    0)
740 DISPLAY AT(21,2):"C COLOR  M MA
    GNIFY  T EDIT"
750 DISPLAY AT(22,2):"A ERASE  Q QU
    IT  B BACKGRD"
```

```
760 DISPLAY AT(23,2):"E=UP X=DOWN S
    =LEFT D=RIGHT"
770 DISPLAY AT(24,8):"L LISTS PROGR
    AM"
780 CALL KEY(0,K,S)
790 IF K=76 THEN GOTO 1000
800 IF K=81 THEN GOTO 990
810 IF K=65 THEN GOTO 100
812 IF K=66 THEN GOSUB 1200
815 IF K=84 THEN GOTO 210
820 IF K=77 THEN GOTO 940
830 IF K=67 THEN GOTO 1160
840 IF K=83 THEN H=H-2
850 IF K=68 THEN H=H+2
860 IF K=69 THEN V=V-2
870 IF K=88 THEN V=V+2
880 IF V>120 THEN V=120
890 IF V<-120 THEN V=-120
900 IF H>120 THEN H=120
910 IF H<-120 THEN H=-120
920 CALL MOTION(#1,V,H)
930 GOTO 780
940 CALL MAGNIFY(M)
950 MM=M
960 IF M=3 THEN M=4 ELSE M=3
970 FOR D=1 TO 20 :: NEXT D
980 GOTO 780
990 STOP
1000 REM PROGRAM LISTER
1010 CALL CHAR(110,"002424")
1020 CALL CLEAR
1030 PRINT "{6 SPACES}PROGRAM LISTI
     NG"
1035 CALL DELSPRITE(ALL)
1040 PRINT
1050 PRINT ">100 CALL CHAR(104,n";:
     : FOR W=1 TO 64 :: PRINT SEG$(
     M$,W,1);:: NEXT W :: PRINT "n)"
1055 PRINT ">105 CALL SCREEN(";SC;"
     )"
1060 PRINT ">110 CALL MAGNIFY(";MM;
     ")"
1070 PRINT ">120 CALL SPRITE(#1,104
     ,";C1;",150,150,";V;",";H;")"
1080 PRINT ">130 CALL KEY(0,K,S)"
1090 PRINT ">140 IF K=68 THEN H=H+2
     "
1100 PRINT ">150 IF K=83 THEN H=H-2
     "
1110 PRINT ">160 IF K=88 THEN V=V+2
     "
1120 PRINT ">170 IF K=69 THEN V=V-2
     "
1130 PRINT ">180 CALL MOTION(#1,V,H
     )"
1140 PRINT ">190 GOTO 130"
1150 PRINT :: PRINT :: PRINT :: PRI
     NT :: PRINT
1155 DISPLAY AT(21,3):"A - ERASE
     {3 SPACES}Q - QUIT"
1156 CALL KEY(0,K,ST):: IF ST=0 THE
     N 1156
1157 IF K=81 THEN GOTO 990
1158 IF K=65 THEN GOTO 100
1159 GOTO 1156
1160 C1=C1+1 :: IF C1>16 THEN 1180
1170 CALL COLOR(#1,C1):: GOTO 780
1180 C1=2 :: CALL COLOR(#1,C1):: GO
     TO 780
1200 REM SCREEN COLOR CHANGE
1210 SC=SC+1 :: IF SC=17 THEN SC=2
1220 CALL SCREEN(SC):: RETURN
```

Atari Menu Buttons

Joseph D. Korman

This utility streamlines the menu selection process by using the OPTION, SELECT, and START keys. The resulting program can be stored on tape or disk and can then be used as the beginning of new programs requiring menu selections.

After catching the programming bug and purchasing an Atari 800, I began to write custom programs for home use. These included checking account, household inventory, telephone book, and the like. In all the programs, the menus ended with an input statement requesting the code for the desired choice. For example:

D. ENTER DEPOSIT
C. ENTER CHECKS
L. LIST CHECKS
S. SAVE REVISED DATA
ENTER NEXT FUNCTION:?

After input of the variable, the program would run a series of IF tests to determine the choice and proceed to the indicated line number for execution. Although the programs worked well, I felt that something was missing to streamline the selection process. I found the answer in James Brunn's article in *COMPUTE!'s First Book of Atari*. The article included information about using the OPTION / SELECT / START buttons on the 800 keyboard.

The menu create utility is actually a skeleton of a program designed to let the user move the cursor to each of the menu options by pressing the OPTION key. Once the cursor is at the line of the desired option; the SELECT key is used to move the program execution to the appropriate line. After the skeleton is loaded, the titles and option names should be changed to reflect the requirements of the new program. After this is done, the programmer need only enter the logic of the options and commands to return to the menu after their execution.

The following program provides ten options starting on line five (5) and printing on each odd line. The user may add more selections to this column and may add a second column to the right side of the screen. If this is done, some changes in the cursor movement logic will be required. This will allow the user to make truly custom menus for the Atari 400/800 programs.

The -] is printed in reverse mode and moves down each time the OPTION button is pressed.

The menu created by this program looks like this on the screen:

"TITLE OF MENU"

-] ITEM 1
ITEM 2
ITEM 3
ITEM 4
ITEM 5
ITEM 6
ITEM 7
ITEM 8
ITEM 9
ITEM 10

Each time the OPTION button is pressed, the arrow moves down the menu one position. Holding the OPTION button causes the arrow to continuously move from top to bottom and jump back to the top. The operator releases the button when the arrow is adjacent to the desired option. The SELECT button is then pressed to execute that part of the program.

Menu Buttons

```
1 REM MENU CREATE UTILITY
10 GRAPHICS 0:SETCOLOR 2,2,8:SETCOLO
   R 1,2,0
11 POKE 752,1
20 POSITION 12,1:? "TITLE OF MENU"
21 POSITION 2,3:? "{3 SPACES}ITEM 1"
22 POSITION 2,5:? "{3 SPACES}ITEM 2"
23 POSITION 2,7:? "{3 SPACES}ITEM 3"
24 POSITION 2,9:? "{3 SPACES}ITEM 4"
25 POSITION 2,11:? "{3 SPACES}ITEM 5
   "
26 POSITION 2,13:? "{3 SPACES}ITEM 6
   "
27 POSITION 2,15:? "{3 SPACES}ITEM 7
   "
28 POSITION 2,17:? "{3 SPACES}ITEM 8
   "
29 POSITION 2,19:? "{3 SPACES}ITEM 9
   "
30 POSITION 2,21:? "{3 SPACES}ITEM 1
   0"
31 REM POSITION ENTRIES ON ALL LINES
    TO INCREASE THE NUMBER OF SELECT
   IONS
35 POSITION 2,3:L=3:? "->"
40 IF PEEK(53279)=3 THEN 50
41 IF PEEK(53279)=5 THEN 60
42 IF PEEK(53279)=6 THEN RUN
43 GOTO 40
50 REM MENU SCROLL
51 POSITION 2,L
52 ? "  "
```

```
53 L=L+2:REM USE L+1 IF ENTRIES ARE
   ON ALL LINES
54 IF L=23 THEN L=3
55 POSITION 2,L:REM FOR TWO COLUMN M
   ENU POSITION 21,L AND ADD LOGIC T
   O RETURN TO LEFT COLUMN FROM BOT
   RT
56 ? "->":FOR T=1 TO 40:NEXT T:REM U
   SE HIGHER NUMBER TO SLOW ->
57 GOTO 40
60 REM SELECT OPTION
61 IF L=3 THEN 100
62 IF L=5 THEN 200
63 IF L=7 THEN 300
64 IF L=9 THEN 400
65 IF L=11 THEN 500
66 IF L=13 THEN 600
67 IF L=15 THEN 700
68 IF L=17 THEN 800
69 IF L=19 THEN 900
70 IF L=21 THEN 1000
71 IF L=23 THEN L=3:GOTO 61
72 REM ADJUST THE ABOVE LOGIC FOR SI
   NGLE LINE SELECTIONS AND DUAL COL
   UMN MENUS
100 GRAPHICS 0:SETCOLOR 2,1,2:SETCOL
    OR 1,1,8
110 POSITION 12,1:? "ITEM NUMBER 1"
115 REM PUT LOGIC FOR THE SELECTION
    HERE
116 REM DON'T FORGET LOGIC TO RETURN
     TO THE MAIN MENU AFTER THE SELE
    CTION IS COMPLETED
117 REM CONTINUE FOR ALL OTHER SELEC
    TIONS
120 FOR T=1 TO 500:NEXT T:RUN
200 GRAPHICS 0:SETCOLOR 2,8,2:SETCOL
    OR 1,8,8
210 POSITION 12,1:? "ITEM NUMBER 2"
220 FOR T=1 TO 500:NEXT T:RUN
300 GRAPHICS 0:SETCOLOR 2,8,8:SETCOL
    OR 1,8,2
310 POSITION 12,1:? "ITEM NUMBER 3"
320 FOR T=1 TO 500:NEXT T:RUN
400 GRAPHICS 0:SETCOLOR 2,4,8:SETCOL
    OR 2,4,2
410 POSITION 12,1:? "ITEM NUMBER 4"
420 FOR T=1 TO 500:NEXT T:RUN
500 GRAPHICS 0:SETCOLOR 2,11,8:SETCO
    LOR 2,11,2
510 POSITION 12,1:? "ITEM NUMBER 5"
520 FOR T=1 TO 500:NEXT T:RUN
600 GRAPHICS 0:SETCOLOR 2,1,2:SETCOL
    OR 1,1,8
610 POSITION 12,1:? "ITEM NUMBER 6"
620 FOR T=1 TO 500:NEXT T:RUN
700 GRAPHICS 0:SETCOLOR 2,8,2:SETCOL
    OR 1,8,8
710 POSITION 12,1:? "ITEM NUMBER 7"
715 IF L=23 THEN L=3:GOTO 61
720 FOR T=1 TO 500:NEXT T:RUN
800 GRAPHICS 0:SETCOLOR 2,8,8:SETCOL
    OR 1,8,2
810 POSITION 12,1:? "ITEM NUMBER 8"
820 FOR T=1 TO 500:NEXT T:RUN
900 GRAPHICS 0:SETCOLOR 2,4,8:SETCOL
    OR 2,4,2
910 POSITION 12,1:? "ITEM NUMBER 9"
920 FOR T=1 TO 500:NEXT T:RUN
1000 GRAPHICS 0:SETCOLOR 2,11,8:SETC
     OLOR 2,11,2
1010 POSITION 12,1:? "ITEM NUMBER 10
     "
1020 FOR T=1 TO 500:NEXT T:RUN
```

©

All About The Hardware Interrupt

Peter Marcotty

Using the hardware interrupt vector is not something that you can learn by reading a user's manual. This article defines it and discusses how to use it in your machine language programs.

An interrupt is a hardware event. Every 60th of a second, a clock inside the computer causes a change in voltage on one of the pins of the 6502 chip (6510 if you have a 64). This change tells the 6502 to stop (interrupt) whatever it is doing, remember how to get back to it, and go to the machine language program pointed to by the hardware interrupt vector (an address inside the computer which *points to* the address of a machine language program that normally "services" the interruption).

Usually the vector sends the computer to a program that updates the screen, looks at the keyboard, and changes the value of TI$. (This is the "servicing.") No matter what you are doing in BASIC or machine language, the interrupt will happen 60 times a second unless you specifically turn it off.

Perhaps the most effective use of the interrupt is that you can wedge a routine of your own into the process, before it goes off to its regular housekeeping chores. Simply point the interrupt vector to the beginning of your routine, do whatever you want to do, and then send the computer to where it usually goes.

In order for us to change the interrupt vector, we must stop the hardware interrupt action altogether. If it tried to jump to the location pointed to by the interrupt vector, and we had changed only one byte of the two-byte vector (remember, interrupts can happen at any time), we'd get some very undesirable results.

Implementing The Interrupt

It will be helpful if you refer to the program for your machine while reading this section.

The first line of your program should be the SEI command. SEI stands for SEt Interrupt mask, and it will stop the computer from interrupting until you let it. After an SEI, you have about 0.009 seconds to change the interrupt vector before the computer gets impatient and crashes. Fortunately, this is plenty of time for our purposes. The next four lines take the address of our program (both the low and high byte) and put them in the hardware interrupt vector. Next we have a CLI (CLear Interrupt mask) which tells the computer it can start performing interrupts again. Finally, we have an RTS command which returns us to BASIC.

The program does not finish running with the RTS command; in fact, it's only just beginning. Since the hardware interrupt vector now points to our own routine, every 60th of a second our main program will be run, almost without any delay in whatever else we might be doing.

At the end of the routine that does the actual work, we cannot return from wherever we were called with a simple RTS. The screen has yet to be updated, and the keyboard hasn't been checked to see if any keys are down. We must JMP to the location where the vector usually points. That's where the servicing routine resides. The locations of the hardware interrupt vector for various computers are given in the table.

The sample program should help you understand how your interrupt routines must be set up.

To turn off your interrupt-driven program, you can change the pointer back to its original value, or on the VIC and 64, simply hit RUN/STOP and RESTORE.

The example programs simply take a look at the contents of the memory location that shows what key is currently being pressed and puts it in the top left corner of the screen.

Two programs are given for each machine. The first can be typed in with an assembler, and the second is a hexadecimal dump to be entered with a monitor. Both have exactly the same effects. To RUN the programs on a PET, type SYS 826; on a VIC or 64, SYS 828. The programs are located in the second cassette buffer, a 192-byte-area of memory that is usually safe for small machine language programs.

Note that interrupt-driven programs will interfere with the normal operation of LOAD and SAVE commands.

Interrupt Memory Locations

The hardware interrupt on the 64 and VIC works in exactly the same way as on the PET, although memory locations will be different.

This table shows all the differences:

	Location of Hardware Interrupt Vector		Points to
Upgrade PETs	144-145	($90-$91)	$E62E
4.0 PETs	144-145	($90-$91)	$E455
64	788-789	($314-$315)	$EA31
VIC	788-789	($315-$315)	$EABF

Program 4: Hardware Interrupt Routine – VIC Version

```
2
4:      Ø33C                        .OPT P4,OO
6:      Ø33C                        *=   $Ø33C
                            ;VIC VERSION
1Ø:     Ø33C 78                     SEI          ;DISABLE INTERRUPTS
2Ø:     Ø33D A9 49                  LDA  #$49    ;LOAD LOW BYTE OF ROUTINE IN LINE 8Ø
3Ø:     Ø33F 8D 14 Ø3               STA  $314    ;STORE LO BYTE OF INTERRUPT VECTOR
4Ø:     Ø342 A9 Ø3                  LDA  #$Ø3    ;LOAD HI BYTE OF ROUTINE IN LINE 8Ø
5Ø:     Ø344 8D 15 Ø3               STA  $315    ;STORE HI BYTE OF INTERRUPT VECTOR
6Ø:     Ø347 58                     CLI          ;REENABLE INTERRUPT
7Ø:     Ø348 6Ø                     RTS          ;RETURN
8Ø:     Ø349 A5 CB                  LDA  2Ø3     ;LOAD CURRENT KEY PRESSED
9Ø:     Ø34B 8D ØØ 1E               STA  $1EØØ   ;STORE IT ON THE SCREEN
95:     Ø34E A9 ØØ                  LDA  #Ø
97:     Ø35Ø 8D ØØ 96               STA  $96ØØ   ;SET COLOR TO BLACK
1ØØØ:   Ø353 4C BF EA               JMP  $EABF

C*
     PC  IRQ  SR AC XR YR SP
.;  B78Ø E455 2C 34 3A 9D FA
.
.:  Ø33C 78 A9 49 8D 14 Ø3 A9 Ø3
.:  Ø344 8D 15 Ø3 58 6Ø A5 CB 8D
.:  Ø34C ØØ 1E A9 ØØ 8D ØØ 96 4C
.:  Ø354 BF EA 49 56 2E 36 34 2E
```

Interrupt Applications

Eric Brandon, Editorial Programmer

Interrupts can be used in many different applications, but the two most common are within utilities and games.

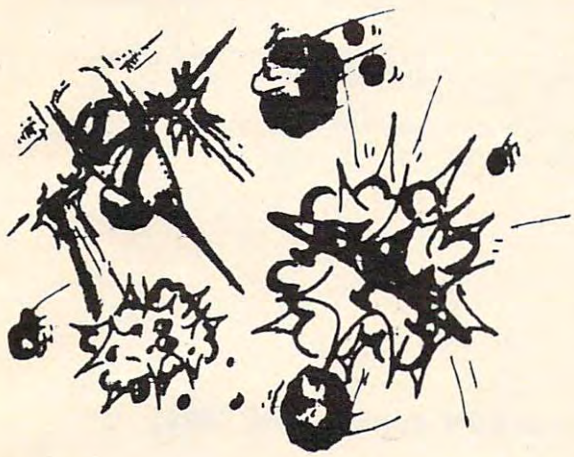
Because an interrupt-driven program is in the "background" of whatever the user is doing, it is ideal for applications where we want to do something concurrent with the normal operation of the computer. Good examples of this are found in "Marquee" (COMPUTE!, February 1981), which displays a message across the top of the screen as a sort of electronic "string around your finger," and "Realtime Clock" (COMPUTE!, January 1982), which displays the time in a corner of the screen to remind you to stop playing Alien Zap and go to bed.

Other uses for interrupt-driven utilities are programs which constantly check which keys are pressed and act accordingly. My favorite from this class is "Keyprint" (COMPUTE!, November/December 1980). Whenever you hit the & and the shift simultaneously, the computer freezes and sends whatever is on the screen to the printer.

In games, interrupts can be used for convenience or smoothness. Suppose you want to write a space game which has a moving starfield in the background. You could worry about writing a program which simultaneously moves your spaceship, the mutant ants, *and* the starfield around, or you could use interrupts. It is a simple matter to write a routine which moves some stars around and to point the interrupt vector to it. Now, you can write your game safe in the knowledge that whatever is going on in your program, those stars will keep floating by.

Best of all, when something holds up your main program for a second or two, such as a sound effect or an explosion, the background won't freeze up but will keep moving, making your game look "smoother" and more professional. ©

D.E.S.-SOFT ™
a division of
DES-Data Equipment Supply Corp.



ZARCON

By Steven Prentiss

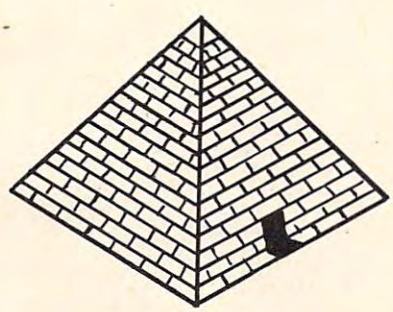
Destined to be one of the best arcade style games on commodore computers.Can you in your Zarcon Fighter shoot all the alien formations while flying through an asteriod field!

5 Levels of excitement
Joystick required
Vic-20 cassette 9.95
Commodore 64 cassette 9.95

SHROOMS

Subterranean Encounter
By Michael Koberstein

Inner Earth has invaded the surface world. The mutant spiders and the moles have taken humans captive and changed them into mussrooms. To rescue them you must enter the caverns. You are their only hope. Good Luck!
Vic-20 & 8K Expander,-Joystick-cass 9.95



SPHINX

3-D ADVENTURE

By Cory Christensen

You the Adventurous Explorer are looking for the lost treasure of the Sphinx. It is rumored to be in the Ancient Pyramid of RA. Beware of deadly traps and hidden passage ways.
Commodore 64-cassette 9.95

You are the commander of squadron of laser ships. It is your duty to defend the cities of earth against incoming alien attack. Spectacular Hi-Res graphics and machine code for super fast action.
Joystick Vic-20 cassette 9.95

SEE YOUR LOCAL DEALER TO SEE OUR FINE PRODUCTS

Dealer inquiries invited ***Software Distribution Available*** **Programs wanted**

(714) 778-5455 **DES** **Data Equipment Supply Corp. 8315 Firestone Blvd., Downey, CA 90241** **(213) 923-9361**

VIC-20™, COMMODORE™, COMMODORE 64™, and CBM™ are trademarks of Commodore Business Machines, Inc.

Program 1: Hardware Interrupt Routine – 4.0 BASIC Version (4032, 8032)

```
2
4:      Ø33A                        .OPT P4,OO
6:      Ø33A                        *=   $Ø33A
                          ;PET 4.Ø VERSION
1Ø:     Ø33A 78                     SEI            ;DISABLE INTERRUPTS
2Ø:     Ø33B A9 45                  LDA  #$45      ;LOAD LOW BYTE OF ROUTINE IN LINE 8Ø
3Ø:     Ø33D 85 9Ø                  STA  $9Ø       ;STORE LO BYTE OF INTERRUPT VECTOR
4Ø:     Ø33F A9 Ø3                  LDA  #$Ø3      ;LOAD HI BYTE OF ROUTINE IN LINE 8Ø
5Ø:     Ø341 85 91                  STA  $91       ;STORE HI BYTE OF INTERRUPT VECTOR
6Ø:     Ø343 58                     CLI            ;REENABLE INTERRUPT
7Ø:     Ø344 6Ø                     RTS            ;RETURN
8Ø:     Ø345 A5 97                  LDA  151       ;LOAD CURRENT KEY PRESSED
9Ø:     Ø347 8D ØØ 8Ø               STA  $8ØØØ     ;STORE IT ON THE SCREEN
1ØØØ:   Ø34A 4C 55 E4               JMP  $E455

C*
     PC  IRQ  SR AC XR YR SP
.;  B78Ø E455 2C 34 3A 9D F8
.
.:  Ø33A 78 A9 45 85 9Ø A9 Ø3 85
.:  Ø342 91 58 6Ø A5 97 8D ØØ 8Ø
.:  Ø34A 4C 55 E4 3Ø 2C 3Ø 3Ø 3F
```

Program 2: Hardware Interrupt Routine – Upgrade ROM Version (3016, 3032)

```
2
4:      Ø33A                        .OPT P4,OO
6:      Ø33A                        *=   $Ø33A
                          ;PET UPGRADE (2.Ø) VERSION
1Ø:     Ø33A 78                     SEI            ;DISABLE INTERRUPTS
2Ø:     Ø33B A9 45                  LDA  #$45      ;LOAD LOW BYTE OF ROUTINE IN LINE 8Ø
3Ø:     Ø33D 85 9Ø                  STA  $9Ø       ;STORE LO BYTE OF INTERRUPT VECTOR
4Ø:     Ø33F A9 Ø3                  LDA  #$Ø3      ;LOAD HI BYTE OF ROUTINE IN LINE 8Ø
5Ø:     Ø341 85 91                  STA  $91       ;STORE HI BYTE OF INTERRUPT VECTOR
6Ø:     Ø343 58                     CLI            ;REENABLE INTERRUPT
7Ø:     Ø344 6Ø                     RTS            ;RETURN
8Ø:     Ø345 A5 97                  LDA  151       ;LOAD CURRENT KEY PRESSED
9Ø:     Ø347 8D ØØ 8Ø               STA  $8ØØØ     ;STORE IT ON THE SCREEN
1ØØØ:   Ø34A 4C 2E E6               JMP  $E62E

C*
     PC  IRQ  SR AC XR YR SP
.;  B78Ø E455 2C 34 3A 9D FA
.
.:  Ø33A 78 A9 45 85 9Ø A9 Ø3 84
.:  Ø342 91 58 6Ø A5 97 8D ØØ 8Ø
.:  Ø34A 4C 2E E6 3Ø 2C 3Ø 3Ø 3F
```

Program 3: Hardware Interrupt Routine – 64 Version

```
2
4:      Ø33C                        .OPT P4,OO
6:      Ø33C                        *=   $Ø33C
                          ;64 VERSION
1Ø:     Ø33C 78                     SEI            ;DISABLE INTERRUPTS
2Ø:     Ø33D A9 49                  LDA  #$49      ;LOAD LOW BYTE OF ROUTINE IN LINE 8Ø
3Ø:     Ø33F 8D 14 Ø3               STA  $314      ;STORE LO BYTE OF INTERRUPT VECTOR
4Ø:     Ø342 A9 Ø3                  LDA  #$Ø3      ;LOAD HI BYTE OF ROUTINE IN LINE 8Ø
5Ø:     Ø344 8D 15 Ø3               STA  $315      ;STORE HI BYTE OF INTERRUPT VECTOR
6Ø:     Ø347 58                     CLI            ;REENABLE INTERRUPT
7Ø:     Ø348 6Ø                     RTS            ;RETURN
8Ø:     Ø349 A5 CB                  LDA  2Ø3       ;LOAD CURRENT KEY PRESSED
9Ø:     Ø34B 8D ØØ Ø4               STA  $Ø4ØØ     ;STORE IT ON THE SCREEN
95:     Ø34E A9 Ø1                  LDA  #1
97:     Ø35Ø 8D ØØ D8               STA  $D8ØØ     ;SET COLOR TO WHITE
1ØØØ:   Ø353 4C 31 EA               JMP  $EA31

C*
     PC  IRQ  SR AC XR YR SP
.;  B78Ø E455 2C 34 3A 9D FA
.
.:  Ø33C 78 A9 49 8D 14 Ø3 A9 Ø3
.:  Ø344 8D 15 Ø3 58 6Ø A5 CB 8D
.:  Ø34C ØØ Ø4 A9 Ø1 8D ØØ D8 4C
.:  Ø354 31 EA 49 56 2E 34 2E 3Ø
```

SJB DISTRIBUTORS. THE MOST COMPETITIVE PRICES ON COMMODORE.

commodore

NEW COMMODORE PRODUCTS

The Executive 64	Call
CBM C128-80	$ 795
CBM BX700	2990
B Series Software	Call
CBM 1520 Plotter	169
CBM 1526 Printer	349

SOFTWARE FOR CBM 64

BUSINESS

Word Processing (WordPro 3+)	$ 69
Quick Brown Fox	56
Writers Assistant	99
Spell Master	75
Calc Result	125
Busicalc II	95
Spread Sheet Assistant	99
Data Manager	70
M File (merge with WordPro)	89
Info Mast	139
64 Mailing List	28
The Manager	50
Home Accountant (continental)	75
Finance Assistant	45
Stock (investment analysis)	80
Agricultural Management	Call
General Ledger, A/R, A/P, P/R, Inv	Call

RECREATION

Assembler Package (cassette or disk, compiled, includes editor, loader, disassembler)	39
Sprite Master (access)	30
Neutral Zone (access)	35
Space Belt	19
Pet Emulator	30
Coco II (build your own games)	40
Vic Tree (programmers utilities)	75
Micro-Term (save to printer,disk)	39
Hesmon	35
Synthesound	45
Gothmogs Lair	30
Road Toad	15
Commodore Games	Call

INTERFACES & ACCESSORIES

80 Column Expander	$159
VIC 1600 Modem	95
VIC 1650 (auto answer, auto dial)	150
VIC 1525 Graphic Printer	225
VIC 1530 Datasette Recorder	65
VIC 1541 Disk Drive	249
VIC Switch (connect 8 64's or Vics to printer, dd)	149
PET-IEEE cable	33
IEEE-IEEE cable (2m)	49
Parallel Interface (Epson, Okidata, IDS, NEC)	70
Programmers Reference Guide	18
Verbatim Diskettes (10 per box)	26
Hes Modem	75
ADA 1450	149
ADA 1800 (new)	129
Numeric Keypad	65

VIC PRODUCTS & ACCESSORIES

8K RAM Memory Expansion Cartridge	$ 40
16K RAM	70
24K RAM	105
VIC 3 Slot Expander	27
VIC 6 Slot Expander	70
Cassette Interface	30
Gorf (64 also)	30
Omega Race	30
Arcade Joystick - Heavy duty w/2 firing buttons! Great for the VIC or 64	25
Auto Clock	125

MONITORS - GREAT RESOLUTION (64 OR VIC)

CBM 1701 Color Monitor	$ 249
Amdek Color Plus	299
Panasonic TR-120 (w/speaker)	155
Panasonic CT-160	279
BMC (green screen)	95
Transtar 20 (high resolution green phosphor)	129
Video/Audio Cable	15

PRINTERS - LETTER QUALITY

CBM 6400, 40 cps	$1450
Diablo 620, 25 cps	949
Transtar 140 (serial)	1395
Transtar 130, 16 cps (auto load, wp features!)	769
NEC 3500 Series	1600
NEC 7700 Series	2350

PRINTERS - DOT MATRIX

CBM 8023, 150 cps/graphics	$ 545
CBM 4023 Printer	395
Epson FX Printer, 160 cps	549
Epson MX-80 FT w/graftrax	Call
Epson FX-100	859
Okidata 82A, 120 cps (serial and parallel)	429
NEC 8023A (parallel)	429
Okidata 92	559
Star Gemini, 10	329
Star Gemini, 15	499
Transtar 315 (hi-res, color)	575

COMMODORE BUSINESS SERIES

SuperPet (5 languages, 2 processors)	$1059
CBM 8032 Computer, 80 Column	595
CBM Memory Expansion, 64K	259
CBM 8050, 1 mg. Dual Drive	995
CBM 8250, 2 mg. Dual Drive	1295
CBM D9060, 5 mg. Hard Disk	1995
CBM D9090, 7.5 mg. Hard Disk	2250
CBM 2031, 170K Single Drive (New)	395
DC Hayes Smart Modem	220

BUSINESS SOFTWARE-8032

WordPro 4+ or 5+	$ 309
InfoPro	219
Administrator	489
VisiCalc (expanded)	199
BPI A/R, G/L, Job Cost, Inventory, Payroll	ea.325

MasterCard, Visa, Money Order, Bank Check

COD (add $5) accepted.
Add 3% surcharge for credit cards.
In stock items shipped within 48 hours.
F.O.B, Dallas, Texas
All products shipped with manufacturer's warranty.
Prices are subject to change without notice.

TO ORDER
CALL TOLL FREE
800-527-4893
800-442-1048
(Within Texas)
Business Hours
Mon.- Fri. 8 to 6, Sat. 10-2

Write for free catalog.

GAME OF THE MONTH

Adventu-Writer (make your own adventure games) Join the Adventu-Writer Club	$ 49

PRODUCT OF THE MONTH

INTERPOD (intelligent IEEE RS232, serial interface for VIC or C64)	$ 179

SJB DISTRIBUTORS INC.
10520 Plano Road, Suite 206
Dallas, Texas 75238
(214) 343-1328

Cracking The Kernal

Peter Marcotty

What is the 64 Kernal? How is it available and how do you use it? This article answers these questions and summarizes each of the Kernal's routines – a real machine language programmer's aid.

What if you want to write a machine language (ML) program for the Commodore 64 that uses the disk drive? Or what if you would like to have your ML program print out to the printer? Where do you begin?

First of all, when you're writing ML programs, it is often helpful to use the routines that are already part of the computer's operating system. But sometimes these routines are buried in ROM among countless other things and they can seem impossible to find. For Commodore 64 users, the Kernal simplifies the search. The Kernal *is* the 64 operating system and contains a collection of extremely useful subroutines that are often quite easy to use.

The wonderful thing about these routines is the incredibly simple way to communicate with them and the powerful results of such brief programming. Often all that is necessary to utilize the subroutine is to load the accumulator (LDA) with one number. Occasionally, a routine will call for another preparatory subroutine to be called first, but these setup routines are just as easy to use.

Using the Kernal involves just these three simple steps: 1) setting up, 2) calling the routine, and 3) handling any errors.

User Callable Kernal Routines

Name	Address Hex	Address Decimal	Function
ACPTR	$FFA5	65445	Input byte from serial port.
CHKIN	$FFC6	65478	Open channel for input.
CHKOUT	$FFC9	65481	Open channel for output.
CHRIN	$FFCF	65487	Input character from channel.
CHROUT	$FFD2	65490	Output character to channel.
CIOUT	$FFA8	65448	Output byte to serial port.
CINT	$FF81	65409	Initialize screen editor.
CLALL	$FFE7	65511	Close all channels and files.
CLOSE	$FFC3	65475	Close a specified logical file.
CLRCHN	$FFCC	65484	Close input and output channels.
GETIN	$FFE4	65508	Get character from keyboard buffer.
IOBASE	$FFF3	65523	Return base address of I/O devices.
IOINIT	$FF84	65412	Initialize input/output.
LISTEN	$FFB1	65457	Command devices on serial bus to LISTEN.
LOAD	$FFD5	65493	Load RAM from a device.
MEMBOT	$FF9C	65436	Read/set bottom of memory.
MEMTOP	$FF99	65433	Read/set top of memory.
OPEN	$FFC0	65472	Open a logical file.
PLOT	$FFF0	65520	Read/set X,Y cursor position.
RAMTAS	$FF87	65415	Initialize RAM, reset tape buffer.
RDTIM	$FFDE	65502	Read realtime clock.
READST	$FFB7	65463	Read I/O status word.
RESTOR	$FF8A	65418	Restore I/O default vectors.
SAVE	$FFD8	65496	Save RAM to device.
SCNKEY	$FF9F	65439	Scan keyboard.
SCREEN	$FFED	65517	Return X,Y organization of screen.
SECOND	$FF93	65427	Send secondary address after LISTEN.
SETLFS	$FFBA	65466	Set logical, first, and second address.
SETMSG	$FF90	65424	Control Kernal messages.
SETNAM	$FFBD	65469	Set filename.
SETTIM	$FFDB	65499	Set realtime clock.
SETTMO	$FFA2	65442	Set time-out on serial bus.
STOP	$FFE1	65505	Check for STOP key.
TALK	$FFB4	65460	Command serial bus device to TALK.
TKSA	$FF96	65430	Send secondary address after TALK.
UDTIM	$FFEA	65514	Increment realtime clock.
UNLSN	$FFAE	65454	Command serial bus to UNLISTEN.
UNTLK	$FFAB	65451	Command serial bus to UNTALK.
VECTOR	$FF8D	65421	Read/set vectored I/O.

Here is a brief summary of each routine with examples:

ACPTR is used to get data off the serial bus. TALK and TKSA must be called first.

```
; Get a byte from the serial bus.
JSR  ACPTR
STA $0800
;This example only shows the end result; call TALK
   and TKSA first.
```

Cassettes are slow...

If you own a Commodore 64™ or VIC 20™ computer, you already know how long it can take to load or save a program. How much time are you wasting just waiting for READY to appear on the screen? Probably a lot, and that's why you need

THE SIGNAL™ from ZAXIS.

THE SIGNAL automatically keeps track of cassette operations and signals you with a pleasant "beep" when both a program header is found and when a Load or Save is completed. You no longer need to stare at the screen for what seems like endless minutes—instead you can go on to other work and when you hear THE SIGNAL, you know that things are READY. THE SIGNAL also provides a reassuring power-on beep, and can be activated under program control.

$29.95

THE SIGNAL plugs right into the back of your VIC 20 or Commodore 64 computer, and your cassette cable plugs into THE SIGNAL. That's all it takes to start making your computer operations more efficient. After you've used THE SIGNAL, you won't know how you got along without it!

THE SIGNAL is available from your favorite computer dealer, or order direct: $29.95 plus $3.00 for UPS shipping and handling (CA residents add 6.5% sales tax). We accept VISA, MasterCard, check or money order. Do not send cash. Sorry, no CODs. *Dealer Inquiries Welcome*

Commodore 64 and VIC 20 are trademarks of Commodore Business Machines, Inc.

ZAXIS™
P.O. Box 666
San Carlos, CA 94070
(415) 592-4334

Commodore 64™ -VIC-20™
software from
ADVANCED ARCADEWARE

4 Game Pak

HEAD ON	A daring and dangerous death race.
ALIEN INVASION	4 players. Creatures descend.
ATOMIC MAN	Super hero rescues a city.
SERPENTS	2 players. Snakes entangle each other.

suggested list price $59.95 **NOW $29.99**

Specify Commodore 64 or VIC-20, Disk-Pak or Cassette Pak
To order: Mail address along with check or money order to:

ADVANCED ARCADEWARE

P.O. Box 845
Thomasville, NC 27360
(919) 431-3231

Add $2.00 for shipping
Allow 2 weeks for delivery

Accounts Payable & Receivable Program

Business or Home owner, keep track of your accounts payable and receivables

- 300 entries per disk
- 75 entries per cassette
- Printer option
- Automatic Sort

Requires 8K or 16K Expander

Disk - $35.95
Cassette - $29.95

COMPU SENSE

TO ORDER:
P.O. BOX 768
WICHITA, KS 67201
(316) 263-1095

AMERICAN EXPRESS ® VISA MasterCard

Handling charges $3.00
C.O.D. (Add $2.00)
Personal checks allow 3 week delivery
VIC-20* is a registered trademark of Commodore
Prices subject to change

"SMART TERMINAL"
TELECOMPUTING POWER FOR VIC - 20 COMMODORE 64

"Smart-Term" does more than convert your VIC or 64 to a null terminal, it gives you features other programs don't offer, such as:

— A PET to ascii conversion test mode
— Transmits all ascii control characters
— Transmits 122 ascii codes
— Receives 92 ascii codes
— Has repeat key feature
— Allows you to enter and save four permanent messages (up to 80 characters each) for one key, transmission of code, password, names, messages, etc.
— Easy to read, smooth scrolling characters
— Fully menu driven for reliability and ease of use
— Transmits and receives in seperate character colors (selectable)
— Many other features

One of the best telecommunications programs available, "Smart-Term" converts your VIC or 64 into a *SMART TERMINAL*. Requires minimum 3K expansion.

TO ORDER: Specify VIC-20 or 64;
TAPE: $16.95 or
DISK: $18.95
(u.s. funds) + $1.50 P&H

TO ORDER SEND CHECK OR MONEY ORDER. CHECK NEED THREE WEEKS TO CLEAR. NO C.O.D.'s TO U.S. CANADIAN ORDERS ADD 25% FOR CDN. FUNDS.

BYTE — RYTE
P.O. BOX 205, STATION CART.
MONTREAL, QUEBEC
H4K 2J5 CANADA

DEALER INQUIRIES INVITED

VIC-20 and Commodore 64 are registered trademarks of Commodore Business Machines Inc.

64K MEMORY FOR THE *VIC 20?

YES, IT'S HERE!

LETCO

ANNOUNCES

THE 64KV MEMORY EXPANSION MODULE W/24K OF NORMAL EXPANSION + 40K ADD'L FOR PROGRAM OR DATA STORAGE. THIS 40K IS PAGED IN OR OUT IN 8K BLOCKS WITH A SINGLE POKE INSTR. THE 64KV USES STATE OF THE ART DYNAMIC RAMS WITH TRANSPARENT REFRESH. ALL 8K BLOCKS ARE SWITCH SELECTABLE AND THE ENTIRE UNIT DRAWS ONLY 250 MA. PLUGS DIRECTLY INTO EXP PORT. ALL THIS AT A PRICE YOU CAN AFFORD

ONLY $179.95

(SHIPPING AND HANDLING INCLUDED)

90 DAY LIMITED WARRANTY

OHIO RESIDENTS ADD SALES TAX

LETCO
7310 WELLS RD.
PLAIN CITY, OH. 43064 DEPT. CM
1-614-873-4410

NAME ______

ADDRESS ______

CITY, STATE ______

ZIP CODE ______

WE ACCEPT VISA OR MASTERCARD. PLEASE INCLUDE CARD NO. EXP. DATE AND SIGN

*VIC 20 IS A REG. TM. OF C.B.M. INC.

CHKIN is used to define any OPENed file as an input file. OPEN must be called first.

```
; Define logical file #2 as an input channel.
LDX #2
JSR  CHKIN
; The X register designates which file #.
```

CHKOUT. Just like CHKIN, but it defines the file for output. OPEN must be called first.

```
; Define logical file #4 as an output file.
LDX #4
JSR  CHKOUT
; Once again the X register defines the file #.
```

CHRIN will get a character from the current input device. Calling OPEN and CHKIN can change the input device.

```
 ; Store a typed string to the screen.
       LDY #$00
LOOP   JSR CHRIN
       STA $0800,Y
       INY
       CMP #$0D
       BNE LOOP
       RTS
 ; This example is like an INPUT statement. Try
   running it.
```

CHROUT. Load the accumulator with your number and call. OPEN and CHKOUT will change the output device.

```
; Duplicate the command of CMD 4:PRINT "A";
LDX #4
JSR  CHKOUT
LDA #'A
JSR  CHROUT
RTS
; The letter A is printed to the screen; call OPEN first
for the printer.
```

CIOUT will send data to the serial bus. LISTEN and SECOND must be called first. Call UNLSN to finish up neatly.

```
; Send the letter X to the serial bus.
LDA  #'X
JSR  CIOUT
RTS
; The accumulator is used to transfer the data.
```

CINT resets the 6567 video controller chip and the screen editor.

```
; Reset the 6567 chip and the 6566 VIC chip.
JSR CINT
RTS
; Basically, just like pressing the STOP and RESTORE
  keys.
```

CLALL really does what its name implies – it closes all files and resets all channels.

```
; Close all files.
JSR CLALL
RTS
; The CLRCHN routine is called automatically.
```

CLOSE. This routine will CLOSE any logical file that has been OPENed.

```
Close logical file #2.
LDA #2
JSR  CLOSE
; The accumulator designates the file #.
```

CLRCHN resets all channels and I/O registers – the input to keyboard and the output to screen.

```
; Restore default values to I/O devices.
JSR CLRCHN
RTS
; The accumulator and the X register are altered.
```

GETIN will get one piece of data from the input device. OPEN and CHKIN can be used to change the input device.

```
 ; Wait for a key to be pressed.
WAIT JSR GETIN
     CMP #0
     BEQ WAIT
 ; If the serial bus is used, then all registers are altered.
```

IOBASE returns the low and high bytes of the starting address of the I/O devices in the X and Y registers.

```
; Set the Data Direction Register of the user port to 0
  (input).
JSR  IOBASE
STX  POINT
STY  POINT+1
LDY  #2
LDA  #0
STA  (POINT),Y
; POINT is a zero-page address used to access the DDR
  indirectly.
```

IOINIT initializes all I/O devices and routines. It is part of the system's powering-up routine.

```
; Initialize all I/O devices.
JSR  IOINIT
RTS
; All registers are altered.
```

LISTEN will command any device on the serial bus to receive data.

```
; Command device #8 to listen.
LDA  #8
JSR
LISTEN
; The accumulator designates the device #.
```

LOAD. The computer will perform either the LOAD or the VERIFY command. If the accumulator is a 1, then LOAD; if 0, then verify.

```
; Load a program into memory.
LDA  #$08
LDX  #$02
LDY  #$00
JSR  SETLFS
LDA  #$04
LDX  #L,NAME
LDY  #H,NAME
JSR  SETNAM
LDA  #$00
LDY  #$20
JSR  LOAD
RTS
```

```
  NAME .BY 'FILE'
  ; Program 'FILE' will be loaded into memory starting at
   8192 decimal, X being the low byte and Y being the
   high byte for the load.
```

MEMBOT. If the carry bit is set, then the low byte and the high byte of RAM are returned in the X and Y registers. If the carry bit is clear, the bottom of RAM is set to the X and Y registers.

```
; Move bottom of memory up one page.
SEC
JSR   MEMBOT
INY
CLC
JSR   MEMBOT
RTS
; The accumulator is left alone.
```

MEMTOP. Same principle as MEMBOT, except the top of RAM is affected.

```
; Protect 1K of memory from BASIC.
SEC
JSR   MEMTOP
DEY
CLC
JSR   MEMTOP
; The accumulator is left alone.
```

OPEN. After SETLFS and SETNAM have been called, you can OPEN a logical file.

```
 ; Duplicate the command OPEN 15,8,15,'I/O'
 LDA  #3
 LDX  #L,NAME
 LDY  #H,NAME
 JSR  SETNAM
 LDA  #15
 LDX  #8
 LDY  #15
 JSR  SETLFS
 JSR  OPEN
 RTS
NAME .BY 'I/O'
 ; OPEN opens the current name file with the current LFS.
```

PLOT. If the carry bit of the accumulator is set, then the cursor X,Y is returned in the Y and X registers. If the carry bit is clear, then the cursor is moved to X,Y as determined by the Y and X registers.

```
; Move cursor to row 12, column 20 (12,20).
LDX  #12
LDY  #20
CLC
JSR   PLOT
; The cursor is now in the middle of the screen.
```

RAMTAS is used to test RAM, reset the top and bottom of memory pointers, clear $0000 to $0101 and $0200 to $03FF, and set the screen memory to $0400.

```
; Do RAM test.
JSR   RAMTAS
RTS
; All registers are altered.
```

RDTIM. Locations 160-162 are transferred, in order, to the Y and X registers and the accumulator.

```
; Store system clock to screen.
JSR   RDTIM
STA  1026
STX  1025
STY  1024
; The system clock can be translated as hours/minutes/
seconds.
```

READST. When called, READST returns the status of the I/O devices. Any error code can be translated as operator error.

```
; Check for read error.
JSR   READST
CMP  #16
BEQ  ERROR
; In this case, if the accumulator is 16, a read error
  occurred.
```

SCREEN returns the number of columns and rows the screen has in the X and Y registers.

```
; Determine the screen size.
JSR   SCREEN
STX   MAXCOL
STY   MAXROW
RTS
; SCREEN allows further compatibility between the 64,
  the VIC-20, and future versions of the 64.
```

SECOND. After LISTEN has been called, a SECONDary address may be sent.

```
; Address device #8 with secondary address #15.
LDA  #8
JSR   LISTEN
LDA  #15
JSR   SECOND
; The accumulator designates the address number.
```

SETLFS stands for SET Logical address, File address, and Secondary address. After SETLFS is called, OPEN may be called.

```
; Set logical file #1, device #8, secondary address of 15.
LDA  #1
LDX  #8
LDY  #15
JSR   SETLFS
; If OPEN is called, the command will be OPEN 1,8,15.
```

SETMSG. Depending on the accumulator, either error messages, control messages, or neither is printed.

```
; Turn on control messages.
LDA  #$40
JSR   SETMSG
RTS
; A 128 is for error messages; a zero, for turning both
  off.
```

SETNAM. In order to access the OPEN, LOAD, or SAVE routines, SETNAM must be called first.

```
 ; SETNAM will prepare the disk drive for 'FILE#1'.
  LDA  #6
  LDX  #L,NAME
  LDY  #H,NAME
  JSR   SETNAM
NAME .BY 'FILE#1'
 ; Accumulator is file length, X is low byte, and Y is high
  byte.
```

Using The Kernal From BASIC

Charles Brannon, Program Editor

Surprisingly, the BASIC programmer will find little use for the Commodore 64's powerful Kernal structure. The Kernal is a collection of machine language modules. Kernal routines exist for OPENing files, reading or writing data, checking the keyboard, testing memory, and reading system variables. All these routines are easily available as BASIC commands, such as OPEN, PRINT, INPUT, GET, FRE(0), etc. You, as a BASIC programmer, have a wealth of such powerful and easy-to-use commands.

When you begin to work with machine language, however, you'll discover that there are no built-in "commands" in the 6502 microprocessor for doing all these tasks. The 6502 (the Commodore 64's 6510 processor is functionally identical) deals with very small tasks, no more complicated than the BASIC statement A=2, or POKE 100+X,3. That's why a library of ready-to-use routines such as the Kernal is so valuable.

However, you can replicate almost anything the Kernal does in BASIC. In fact, the BASIC interpreter, which lets you edit and run BASIC programs, is just a large machine language program that itself calls the same Kernal routines.

You can do almost everything machine language and the Kernal does in BASIC, assisted by POKE and PEEK, just more slowly (since BASIC has to be interpreted, command by command, instead of directly executed like machine language). Using the Kernal, it is easy to write very short machine language routines which do things faster and more efficiently than BASIC.

SETTIM is the opposite of RDTIM: it SETs the system clock instead of ReaDing it.

```
; Set system clock to 10 minutes=3600 jiffies.
LDA  #0
LDX  #L,3600
LDY  #H,3600
JSR  SETTIM
; This allows very accurate timing for many things.
```

SETTMO is used only with an IEEE add-on card to access the serial bus.

```
; Disable time-outs on serial bus.
LDA  #0
JSR  SETTMO
; To enable time-outs, set the accumulator to a 128 and
  call SETTMO.
```

STOP will set the Z flag of the accumulator if the STOP key was pressed.

```
  ; Check for STOP key being pressed.
WAIT JSR  STOP
     BNE  WAIT
     RTS
  ; STOP must be called if the STOP key is to remain
    functional.
```

TALK. This routine will command a device on the serial bus to send data.

```
; Command device #8 to TALK.
LDA  #8
JSR  TALK
RTS
; The accumulator designates the file number.
```

TKSA is used to send a secondary address for a TALK device. TALK must be called first.

```
; Signal device #4 to talk with command #7.
LDA  #4
JSR  TALK
LDA  #7
JSR  TKSA
RTS
; This example will tell the printer to print in uppercase.
```

UDTIM. If you are using your own interrupt system, you can update the system clock by calling UDTIM.

```
; Update the system clock.
JSR  UDTIM
RTS
; It is useful to call UDTIM before calling STOP.
```

UNLSN commands all devices on the serial bus to stop receiving data.

```
; Command the serial bus to UNLiSteN.
JSR  UNLSN
RTS
; The serial bus can now be used for other things.
```

UNTLK. All devices previously set to TALK will stop sending data.

```
; Command serial bus to stop sending data.
JSR  UNTLK
RTS
; Sending UNTLK commands all talking devices to get
  off the serial bus.
```

VECTOR. If the carry bit of the accumulator is set, the start of a list of the current contents of the RAM vectors is returned in the X and Y registers. If the carry bit is clear, then the user list pointed to by the X and Y registers is transferred to the system RAM vectors.

```
; Change the input routines to new system.
SEC
JSR  VECTOR
LDA  #L,MYINP
STA  USER+10
LDA  #H,MYINP
```

```
      STA  USER+11
      LDX  #L,USER
      LDY  #H,USER
      CLC
      JSR  VECTOR
      RTS
USER  .DE 26
; The new input list can start anywhere. USER is the
  location for temporary strings, and 35-36 is the utility
  pointer area.
```

Error Codes

If an error occurs during a Kernal routine, then the carry bit of the accumulator is set and the error code is returned in the accumulator.

Number	Meaning
0	Routine ended by the STOP key.
1	Too many files open.
2	File already open.
3	File not open.
4	File not found.
5	Device not present.
6	File is not an input file.
7	File is not an output file.
8	File name is missing.
9	Illegal device number.
240	Top-of-memory change RS-232 buffer allocation. ©

Mastermaze Update For The Atari

David Butler

In the February 1983 issue of COMPUTE!, *there was an excellent, multilevel maze game called "Mastermaze" by Kenneth S. Szajda. Here's a machine language routine for the Atari version that greatly speeds up the maze generator.*

A game written in machine language can run far faster than a BASIC version. However, in the case of this "Mastermaze" update, the speed of the game is not affected, only the maze generator which starts things off.

Before you can even begin to play Mastermaze, your computer must generate from 1 to 32 levels of mazes. This process can take several minutes. By incorporating this new machine language routine, you should find that the maze generating part of the program runs more than ten times faster. Thirty-two levels can be generated in under 90 seconds.

To use this routine to update your version of Mastermaze, first LIST the original version of the program. Delete lines 50, 90, 100, 110, and 111. Then, substitute the lines in this new program listing where they appear in the original. Also add the new lines that did not appear in Mastermaze.

Adding An ML Maze Generator

```
8 DIM A$(37):SW=Ø
26 POKE 752,1:? "{CLEAR}":POSITION 4,1Ø:? "CONSTRUCTING MAZE...WAIT FOR START"
27 RESTORE 1ØØØ:FOR I=1536 TO 169Ø:READ A:POKE I,A:NEXT I:POKE 755,1
8Ø G=USR(1536,1675,A):GOSUB 5ØØ
1ØØØ DATA 1Ø4,1Ø4,133,2Ø8,1Ø4,133,2Ø7,1Ø4,133,2Ø4,1Ø4,133,2Ø3,173,1Ø,21Ø,41,3,133,212
1Ø1Ø DATA 133,213,24,1Ø,168,165,2Ø3,113,2Ø7,133,2Ø5,165,2Ø4,2ØØ,113,2Ø7,133,2Ø6,16Ø,Ø
1Ø2Ø DATA 177,2Ø5,2Ø1,128,2Ø8,4Ø,165,212,24,1Ø5,1,145,2Ø5,1Ø5,3,1Ø,168,165,2Ø3,113
1Ø3Ø DATA 2Ø7,133,2Ø3,2ØØ,165,2Ø4,113,2Ø7,133,2Ø4,169,Ø,168,145,2Ø3,165,2Ø5,133,2Ø3,165
1Ø4Ø DATA 2Ø6,133,2Ø4,24,144,183,23Ø,212,165,212,41,3,133,212,197,213,2Ø8,18Ø,16Ø,Ø
1Ø5Ø DATA 177,2Ø3,133,212,152,145,2Ø3,169,251,24,1Ø1,212,176,24,198,212,165,212,24,1Ø
1Ø6Ø DATA 168,56,165,2Ø3,241,2Ø7,133,2Ø3,2ØØ,165,2Ø4,241,2Ø7,133,2Ø4,24,144,131,96,2
1Ø7Ø DATA Ø,176,255,254,255,8Ø,Ø,1,Ø,216,255,255,255,4Ø,Ø
```

©

COMPUTE! Back Issues

Here are some of the applications, tutorials, and games from available back issues of **COMPUTE!**. Each issue contains much, much more than there's space here to list, but here are some highlights:

May 1981: Named GOSUB/GOTO in Applesoft, Generating Lower Case Text on Apple II, Copy Atari Screens to the Printer, Disk Directory Printer for Atari, Realtime Clock on Atari, PET BASIC Delete Utility, PET Calculated Bar Graphs, Running 40 Column Programs on a CBM 8032.

June 1981: Computer Using Educators (CUE) on Software Pricing, Apple II Hires Character Generator, Ever- expanding Apple Power, Color Burst for Atari, Mixing Atari Graphics Modes 0 and 8, Relocating PET BASIC Programs, An Assembler In BASIC for PET, QuadraPET: Multitasking?

July 1981: Home Heating and Cooling, Animating Integer BASIC Lores Graphics, The Apple Hires Shape Writer, Adding a Voice Track to Atari Programs, Machine Language Atari Joystick Driver, Four Screen Utilities for the PET, Saving Machine Language Programs on PET Tape Headers, Commodore ROM Systems, The Voracious Butterfly on OSI.

August 1981: Minimize Code and Maximize Speed, Apple Disk Motor Control, A Cassette Tape Monitor for the Apple, Easy Reading of the Atari Joystick, Blockade Game for the Atari, Atari Sound Utility, The CBM "Fat 40," Keyword for PET, CBM/PET Loading, Chaining, and Overlaying.

October 1981: Automatic DATA Statements for CBM and Atari, VIC News, Undeletable Lines on Apple, PET, VIC, Budgeting on the Apple, Switching Cleanly from Text to Graphics on Apple, Atari Cassette Boot-tapes, Atari Variable Name Utility, Atari Program Library, Train your PET to Run VIC Programs, Interface a BSR Remote Control System to PET, A General Purpose BCD to Binary Routine, Converting to Fat-40 PET.

December 1981: Saving Fuel $$ (multiple computers: versions for Apple, PET, and Atari), Unscramble Game (multiple computers), Maze Generator (multiple computers), Animating Applesoft Graphics, A Simple Printer Interface for the Apple II, A Simple Atari Wordprocessor, Adding High Speed Vertical Positioning to Atari P/M Graphics, OSI Supercursor, A Look At SuperPET, Supermon for PET/CBM, PET Mine Maze Game.

January 1982: Invest (multiple computers), Developing a Business Algorithm (multiple computers), Apple Addresses, Lowercase with Unmodified Apple, Cryptogram Game for Atari, Superfont: Design Special Character Sets on Atari, PET Repairs for the Amateur, Micromon for PET, Self-modifying Programs in PET BASIC, Tinymon: a VIC Monitor, Vic Color Tips, VIC Memory Map, ZAP: A VIC Game.

May 1982: VIC Meteor Maze Game, Atari Disk Drive Speed Check, Modifying Apple's Floating Point BASIC, Fast Sort For PET/CBM, Extra Atari Colors Through Artifacting, Life Insurance Estimator (multiple computers), PET Screen Input, Getting The Most Out Of VIC's 5000 Bytes.

August 1982: The New Wave Of Personal Computers, Household Budget Manager (multiple computers), Word Games (multiple computers), Color Computer Home Energy Monitor, Intelligent Apple Filing Cabinet, Guess That Animal (multiple computers), PET/CBM Inner BASIC, VIC Communications, Keyprint Compendium, Animation With Atari, VIC Curiosities, Atari Substring Search, PET and VIC Electric Eraser.

September 1982: Apple and Atari and the Sounds of *TRON*, Commodore Automatic Disk Boot, VIC Joysticks, Three Atari GTIA Articles, Color Computer Graphics, The Apple Pilot Language, Sprites and Sound on the Commodore 64, Peripheral Vision Exerciser (multiple computers), Banish INPUT Statements (multiple computers), Charades (multiple computers), PET Pointer Sort, VIC Pause, Mapping Machine Language, Editing Atari BASIC With the Assembler Cartridge, Process Any Apple Disk File.

January 1983: Sound Synthesis And The Personal Computer, Juggler And Thunderbird Games (multiple computers), Music And Sound Programs (multiple computers), Writing Transportable BASIC, Home Energy Calculator (multiple computers), All About Commodore WAIT, Supermon64, Perfect Commodore INPUTs, Atari Autonumber, Copy VIC Disk Files, Commodore 64 Architecture.

February 1983: How The Pros Write Computer Games, 12 Joysticks Compared, Slalom (a game in 3-D for multiple computers), Super Shell Sort For PET, Atari SuperFont Plus, Creating Graphics On The VIC, Joysticks And Sprites On The 64, Bi-Directional VIC Scrolling, Commodore 64 Video: A Guided Tour, The Atari Cruncher, Easy Apple Editing, VIC Custom Characters For Games.

March 1983: An Introduction To Data Storage (multiple computers), Mass Memory Now And In The Future, Games: Closeout, Boggler, Fighter Aces, Letter And Number Play (all for multiple computers), VIC Music, Direct Atari Disk Access, TRS-80 Color Computer Data Base, Apple Subroutine Capture, PET Quickplot, TI Graphics Made Easy, VIC and Atari Memory Management.

April 1983: Selecting The Right Word Processor, Air Defense (multiple computers), Scriptor: An Atari Word Processor, Retirement Planner (multiple computers), TI-99 Match-Em, Dr. Video For Commodore, Atari Filefixer, Video 80: 80 Columns For The Atari, VICword, Magic Commodore BASIC.

May 1983: The New Low Cost Printer/Plotters, Jumping Jack (multiple computers), Deflector (multiple computers), VIC Kaleidoscope, Graphics on the Sinclair/Timex, Bootmaker For VIC, PET and 64, VICSTATION: A "Paperless Office," The Atari Musician, Apple Fast Sort, TI BASIC One-liners.

June 1983: How To Buy The Right Printer, The New, Low-cost Printers, Astrostorm (multiple computers), The Hawkmen Of Dindrin (multiple computers), MusicMaster For The Commodore 64, Commodore Data Searcher, Atari Player/Missile Graphics Simplified, TI Structured BASIC, UnNEW For The VIC and 64, Atari Fast Shuffle, VIC Contractor, Sinclair/Timex Screen Splitter.

Home and Educational COMPUTING! (Fall 1981 and Summer 1981 – count as one back issue): Exploring The Rainbow Machine, VIC As Super Calculator, Custom Characters, Alternate Screens, Automatic Line Numbers, Using The Joystick (Spacewar Game), Fast Tape Locater, Window, VIC Memory Map.

Back issues are $3 each or six for $15. Price includes freight in the US. Outside the US add $1 per magazine ordered for surface postage. $4 per magazine for air mail postage. All back issues subject to availability.

In the Continental US call
TOLL FREE 800-334-0868
(In NC Call 919-275-9809)

Or write to **COMPUTE!** Back Issues, P.O. Box 5406, Greensboro, NC 27403 USA. Prepayment required in US funds. MasterCard, Visa and American Express accepted. North Carolina Residents add 4% sales tax.

COMPUTE!'s Second Book Of Atari

After only three years on the market, the Atari 400/800 microcomputers have become among the most popular personal computers ever made. So it was no surprise when *COMPUTE!'s First Book of Atari*, a collection of the best Atari articles published during 1980-81 in **COMPUTE!** Magazine, also became a "bestseller" with Atari enthusiasts. The first printing sold out in just a few months.

That's why we've followed up with *COMPUTE!'s Second Book of Atari*. Available immediately, the *Second Book of Atari* continues **COMPUTE!**'s tradition for personal computer users.

But the *Second Book of Atari* differs from the *First Book* in one important respect – all the articles are totally new and previously unpublished. The *Second Book of Atari* includes such interesting articles as "Page Flipping," "Fun With Scrolling," "Perfect Pitch," "Player-Missile Drawing Editor," and "TextPlot Makes a Game." Whole chapters are devoted to subjects such as "Advanced Graphics and Game Utilities," "Programming Techniques," and "Beyond BASIC." With 250 pages – more than 25 percent thicker than the *First Book* at the same price – the *Second Book of Atari* is crammed with information and ready-to-type program listings. And the book is spiral-bound to lie flat and is fully indexed for quick reference.

Best of all, *COMPUTE!'s Second Book of Atari*, like **COMPUTE!** Magazine itself, is written and edited to appeal to all computer enthusiasts – beginners and experts alike. Priced at only $12.95.

Available at computer dealers and bookstores nationwide. To order directly call TOLL FREE 800-334-0868. In North Carolina call 919-275-9809. Or send check or money order to **COMPUTE! Books**, P.O. Box 5406, Greensboro, NC 27403.

Add $2.00 shipping and handling. Outside the U.S., add $5.00 for air mail, $2.00 for surface mail. All orders prepaid, U.S. funds only.

COMPUTE!'s Programmer's Reference Guide to the TI-99/4A

Author: C. Regena
Price: $14.95
On Sale: Now

Just about the best way to learn how to program a computer is to sit down with a patient friend who already knows how, and ask questions while you experiment with the computer. Owners of the popular Texas Instruments home computer will find that C. Regena is that kind of friend, and *Programmer's Reference Guide to the TI-99/4A* is that kind of book.

Regena carefully explains every BASIC command and function, and all the techniques needed to program TI graphics, sound, and speech. It's hard to think of a question that she doesn't answer simply and clearly, with hints about ways to write programs that do exactly what you want.

The book also provides dozens and dozens of programs, ranging from very short examples to full-length commercial-quality software. In effect, readers can look over Regena's shoulder as she goes through the programming process step by step, explaining what she's doing as she goes along. Not to mention the fact that the finished programs are valuable in their own right.

Even readers who are familiar with the computer will find this book valuable as a reference, where they can look up information they need and find the answers to particular questions.

Above all, *Programmer's Reference Guide to the TI-99/4A* is a book that lets readers use it however they like. You don't have to start at page one and read through, following someone else's plan for what you should learn first and what can wait until later. Instead, you can explore this book from any point of view, to solve almost any programming problem, and find the answer quickly and easily.

C. Regena is **COMPUTE!** Magazine's regular columnist on the TI-99/4A. She's an experienced and resourceful programmer. Like most of her readers, she taught herself how to program, and she hasn't forgotten what it's like to be a beginner, just starting out with the computer. And with *Programmer's Reference Guide*, TI users now have Regena to help them learn how to make their computer do exactly what they want it to do.

Programmer's Reference Guide to the TI-99/4A is available from **COMPUTE!** Publications, the leading publisher of books and magazines for home, educational, and recreational computing.

Available at computer dealers and bookstores nationwide. To order directly call TOLL FREE 800-334-0868. In North Carolina call 919-275-9809. Or send check or money order to **COMPUTE! Books**, P.O. Box 5406, Greensboro, NC 27403.

Add $2 shipping and handling. Outside the U.S. add $5 for air mail, $2 for surface mail. All orders prepaid, U.S. funds only.

Table of Contents

COMPUTE!'s The Atari BASIC Sourcebook

Authors: Bill Wilkinson, Kathleen O'Brien, and Paul Laughton
Price: $12.95
On Sale: Now

If you program in BASIC, you know about commands like PRINT, GOSUB, IF-THEN, and others.

But did you know that each of these commands is actually a mini-program in itself? Atari BASIC is a collection of machine language routines that tell the computer what to do, how to do it, and what to do next.

Atari BASIC is a powerful and versatile language. Now available from **COMPUTE! Books**, *The Atari BASIC Sourcebook* offers Atari programmers a chance to look inside the language – directly to the source code that *is* Atari BASIC.

Authors Bill Wilkinson, Kathleen O'Brien, and Paul Laughton, the people who wrote Atari BASIC, take you on a tour through the language. They explain how it works and how you can make it work for you.

The Atari BASIC Sourcebook answers these questions (and more):

- When you RUN a BASIC program, what is really going on inside the computer?
- How does the computer know how to handle a FOR-NEXT loop? How does it RETURN from a subroutine?
- Where do ERROR messages come from? How does the computer know what's wrong?
- How does your Atari decide which mathematical operation to perform first?
- Why do some tasks take so long, while others happen almost instantly?
- Why does the computer sometimes lock up when you delete lines from a program?
- How does the computer interpret words and symbols like GOTO, INT, CHR$, *, and =?
- How can a machine language programmer take advantage of the sophisticated routines in Atari BASIC?

Intermediate to advanced Atari programmers will find a wealth of useful and interesting information in *The Atari BASIC Sourcebook.*

Much more than a simple source code listing, this book explains how BASIC works and why. All major routines are examined and explored. The authors go into detail about the internal design, the stack, input/output statements, and much more. When you finish reading this book, you will have an in-depth understanding of how to put Atari BASIC to work for you in ways you never thought possible.

The Atari BASIC Sourcebook is available at many computer stores and bookstores, and can also be ordered directly from **COMPUTE! Books**.

Available at computer dealers and bookstores nationwide. To order directly call TOLL FREE 800-334-0868. In North Carolina call 919-275-9809. Or send check or money order to **COMPUTE! Books**, P.O. Box 5406, Greensboro, NC 27403.

Add $2 shipping and handling. Outside the U.S. add $5 for air mail, $2 for surface mail. All orders prepaid, U.S. funds only.

A Beginner's Guide To Typing In Programs

What Is A Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential*, but without a program, it isn't going anywhere. Most of the programs published in **COMPUTE!** are written in a computer language called BASIC. BASIC is easy to learn and is built into most computers (on some computers, you have to purchase an optional BASIC cartridge).

BASIC Programs

Each month, **COMPUTE!** publishes programs for many machines. To start out, type in only programs written for your machine, e.g., "TI Version" if you have a TI-99/4. Later, when you gain experience with your computer's BASIC, you can try typing in and converting certain programs from one computer to yours.

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one "right way" of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as "O" for the numeral "0", a lowercase "l" for the numeral "1", or an uppercase "B" for the numeral "8". Also, you must enter all punctuation such as colons and commas just as they appear in the magazine. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

Brackets And Special Characters

The exception to this typing rule is when you see the curved bracket, such as "{DOWN}". Anything within a set of brackets is a special character or characters that cannot easily be listed on a printer. When you come across such a special statement, refer to the appropriate key for your computer. For example, if you have an Atari, refer to the "Atari" section in "How to Type COMPUTE!'s Programs."

About DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could "lock up," or "crash." The keyboard, break key, and RESET (or STOP) keys may all seem "dead," and the screen may go blank. Don't panic – no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, so always SAVE a copy of your program before you RUN it. If your computer crashes, you can LOAD the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is RUN. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get To Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program, so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? You can always retype the line, but you at least need to know how to backspace. Do you know how to enter inverse video, lowercase, and control characters? It's all explained in your computer's manuals.

A Quick Review

1) Type in the program a line at a time, in order. Press RETURN or ENTER at the end of each line. Use backspace or the back arrow to correct mistakes.

2) Check the line you've typed against the line in the magazine. You can check the entire program again if you get an error when you RUN the program.

3) Make sure you've entered statements in brackets as the appropriate control key (see "How To Type COMPUTE!'s Programs" elsewhere in the magazine.)

We regret that we are no longer able to respond to individual inquiries about programs, products, or services appearing in **COMPUTE!** *due to increasing publication activity. On those infrequent occasions when a published program contains a typo, the correction will appear on the CAPUTE! page, usually within eight weeks. If you have specific questions about items or programs which you've seen in* **COMPUTE!**, *please send them to Readers Feedback, P.O. Box 5406, Greensboro, NC 27403.* ©

How To Type COMPUTE!'s Programs

Many of the programs which are listed in **COMPUTE!** contain special control characters (cursor control, color keys, inverse video, etc.). To make it easy to tell exactly what to type when entering one of these programs into your computer, we have established the following listing conventions. There is a separate key for each computer. Refer to the appropriate tables when you come across an unusual symbol in a program listing. If you are unsure how to actually enter a control character, consult your computer's manuals.

Atari 400/800

Characters in inverse video will appear like: INVERSE VIDEO
Enter these characters with the Atari logo key, {⋀}.

When you see	Type	See	
{CLEAR}	ESC SHIFT <		Clear Screen
{UP}	ESC CTRL -		Cursor Up
{DOWN}	ESC CTRL =		Cursor Down
{LEFT}	ESC CTRL +		Cursor Left
{RIGHT}	ESC CTRL *		Cursor Right
{BACK S}	ESC DELETE		Backspace
{DELETE}	ESC CTRL DELETE		Delete character
{INSERT}	ESC CTRL INSERT		Insert character
{DEL LINE}	ESC SHIFT DELETE		Delete line
{INS LINE}	ESC SHIFT INSERT		Insert line
{TAB}	ESC TAB		TAB key
{CLR TAB}	ESC CTRL TAB		Clear tab
{SET TAB}	ESC SHIFT TAB		Set tab stop
{BELL}	ESC CTRL 2		Ring buzzer
{ESC}	ESC ESC		ESCape key

Graphics characters, such as CTRL-T, the ball character ● will appear as the "normal" letter enclosed in braces, e.g. {T}.

A series of identical control characters, such as 10 spaces, three cursor-lefts, or 20 CTRL-R's, will appear as {10 SPACES}, {3 LEFT}, {20 R}, etc. If the character in braces is in inverse video, that character or characters should be entered with the Atari logo key. For example, {♥} means to enter a reverse-field heart with CTRL-comma, {5U} means to enter five inverse-video CTRL-U's.

Commodore PET/CBM/VIC/64

Generally, any PET/CBM/VIC/64 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the shift key. If you find an underlined key enclosed in braces (e.g., {10 N}), you should type the key as many times as indicated (in our example, you would enter ten shifted N's). Some graphics characters are inaccessible from the keyboard on CBM Business models (32N, 8032).

For the VIC and 64, if a key is enclosed in special brackets, ⦃ ⦄, you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as indicated.

Rarely, you'll see in a Commodore 64 program a solitary letter of the alphabet enclosed in braces. These characters can be entered by holding down the CTRL key while typing the letter in the braces. For example, {A} would indicate that you should press CTRL-A.

About the *quote mode*: you know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

Use the following tables when entering special characters:

When You Read:	Press:	See:	When You Read:	Press:	See:
{BLK}	CTRL 1		⦃5⦄	C= 5	
{WHT}	CTRL 2		⦃6⦄	C= 6	
{RED}	CTRL 3		⦃7⦄	C= 7	
{CYN}	CTRL 4		⦃8⦄	C= 8	
{PUR}	CTRL 5		{F1}	f1	
{GRN}	CTRL 6		{F2}	f2	
{BLU}	CTRL 7		{F3}	f3	
{YEL}	CTRL 8		{F4}	f4	
⦃1⦄	C= 1		{F5}	f5	
⦃2⦄	C= 2		{F6}	f6	
⦃3⦄	C= 3		{F7}	f7	
⦃4⦄	C= 4		{F8}	f8	

All Commodore Machines

Clear Screen	{CLR}	Cursor Left	{LEFT}
Home Cursor	{HOME}	Insert Character	{INST}
Cursor Up	{UP}	Delete Character	{DEL}
Cursor Down	{DOWN}	Reverse Field On	{RVS}
Cursor Right	{RIGHT}	Reverse Field Off	{OFF}

Apple II / Apple II Plus

All programs are in Applesoft BASIC, unless otherwise stated. Control characters are printed as the "normal" character enclosed in brackets, such as {D} for CTRL-D. Hold down CTRL while pressing the control key. You will not see the special character on the screen.

Texas Instruments 99/4

The only special characters used are in PRINT statements to indicate where two or more spaces should be left between words. For example, ENERGY {10 SPACES} MANAGEMENT means that ten spaces should be left between the words ENERGY and MANAGEMENT. Do not type in the braces or the words 10 SPACES. Enter all programs with the ALPHA LOCK on (in the down position). Release the ALPHA LOCK to enter lowercase text.

CAPUTE!

Modifications Or Corrections To Previous Articles

RATS! For The 64

The 64 version of this game from the July issue is in two parts. Sue Roberts suggests a simple addition which will cause the first part, the setup program, to make the second part, the game itself, LOAD and RUN automatically. Disk users should SAVE the main game program with the filename RATMAZE, then add the following line to the setup program (Program 2, p. 60):

```
160 LOAD"RATMAZE",8:RUN
```

Astrostorm For TI

In the TI-99/4A version of "Astrostorm" (June 1983, p. 82), line 780 should read:

```
780 IF CSHIP>0 THEN 810
```

Hawkmen Of Dindrin, VIC And 64 Versions

In the second part of the VIC version of this game from the June 1983 issue (Program 2, p. 92), the { 06 LEFT }(six cursor lefts) in line 58 should be omitted. If you happen to be pushing the joystick when you lose your last player, the game ends. Bruce Stevenson and others suggest the following additional line to give you time to release the joystick or fire button:

```
1024 FOR X=1 TO 700:NEXT X
```

In the 64 version, the misplaced line 288 should be omitted.

Checkers For The 64

Arnold J. Schmeling suggests the following addition and correction for this game from the May 1983 issue (p. 90), which prevent the computer from allowing illegal moves:

```
545 IF S(E,H)=1 AND B-H<1 THEN 1040
550 IF ABS(E-A)=2 AND S((E+A)/2,(H+B)/2)=
    >0 THEN 1040
```

UnNEW For The 64

Under most conditions, this utility program from the June 1983 issue (p. 213) works equally well on either the VIC or 64. However, to guarantee proper operation on the 64, reader Don Lewis suggests that the existing line 330 be replaced with:

```
330 116,164
```

and a new tape be prepared in accordance with the original instructions. ©

NEWS & PRODUCTS

Drawing By Touch

Koala Technologies has introduced a touch tablet that allows computer users to draw directly on the video display screen, bypassing the keyboard.

The Koalapad Touch Tablet, available in versions for the Apple, Atari, Commodore 64, and IBM computers, weighs about a pound, and connects to the computer through a joystick port.

It can be used as a sketch pad, as a custom keypad, or as a game controller.

Though other applications are available, the Touch Tablet is packaged with *Micro Illustrator* from Island Graphics. This combination allows the touch pad to be used as a drawing board. Images can be drawn with a finger or stylus, and shapes, colors, shadings, and various "paintbrushes" can be selected from a menu.

The Koalapad Touch Tablet can be used for drawing or as an auxiliary keyboard.

The touch tablet and *Micro Illustrator* package sells for $125. Additional application programs will sell for about $50.

Koala Technologies Corp.
4962 El Camino Real, Suite 125
Los Altos, CA 94022

Atari 400 Expansion

A 48K memory expansion kit, designed to upgrade the 8K or 16K Atari 400, is available from Atari.

The board is available for $130 installed at Atari Regional Repair Centers, or, for those who prefer to install the board themselves, it will be available through the Atari Program Exchange for $110.

Atari Inc.
1265 Borregas Ave.
P.O. Box 427
Sunnyvale, CA 94086
(800) 538-8543

Checkbook System

T & F Software has produced a check register program for the 8K VIC-20, the Commodore 64, and Atari computers.

CheckEase! allows maintenance of multiple accounts, in-

ATARI
Computers for people.

800 48K $399.00*

* Plus $100.00 Rebate back from ATARI

810 Disk Drive....... $399.00

Educator	$109.75	Entertainer	$ 63.75
Programmer	$ 51.75	Bookeeper	$164.75

600 XL	$199.00
800 XL	$CALL$
1200 XL	$499.00
1400 XL	$CALL$
1450 XL	$CALL$
1025 Printer	$399.00
1020 Color Printer	$245.00
1027 Printer	$299.00
1010 Recorder	$75.00
410 Recorder	$75.00
810 Disk Drive	$399.00
1050 Disk Drive	$335.00
850 Interface	$CALL$

PARKER BROTHERS

Tutankham R	$33.75
Super Cobra R	$33.75
Astro Chase R	$33.75
Frogger R	$33.75
QBert R	$33.75
Popeye R	$33.75
Risk R	$42.75
Chess R	$42.75

SPINNAKER

Story Machine R	$26.75
Face Maker R	$24.75
Kinderomp R	$20.75
Fraction Fever R	$24.75
Delta Drawing R	$26.75

SSI

Battle of Shilo C/D	$26.75
Tigers in the Snow C/D	$26.75
Battle for Normandy C/D	$26.75
Knights of the Desert C/D	$26.75
Cosmic Balance C/D	$26.75

ON-LINE

Frogger	$24.95
Wizard & Prin	$26.95

ROKLAN

Wizard of War	$29.75
Gorf	$29.75
Delux Invader	$27.95

BIG 5

Miner 2049	$32.75

BUSINESS

Visicalc	$159.75
Letter Perfect	$115.75
Letter Perfect	$149.75
Data Perfect	$99.75
Text Wizzard	$49.75
Spell Wizzard	$64.75
File Manager	$69.75
Home File Mgr	$69.75
Bookeeper	$119.75
C.R.I.S	$199.75
Atari Word Pro	$109.75
Tax Advantage	$35.75
Home Accountant	$59.75
Bank Street W	$49.75
Atari Writer	$55.75

ADVENTURE

Preppie	$19.95
Preppie 2	$19.95
Diskey	$39.95
Sea Dragon	$25.50
Stratos	$25.00
Treasure Quest	$13.50
Series 1-12	$15.95
Saga Series 1-6	$24.95
Stone of Sisyphus	$29.95
Eliminator	$18.95

apple®

Apple is a Registered Trademark of Apple Computer, Inc.

SSI

Battle of Shilo	$26.75
Tigers in the Snow	$26.75
Cosmic Balance	$26.75
Knights of the Desert	$26.75
Battle for Normandy	$26.75
Germany 1985	$36.75

RANA DISK DRIVES

Elite 1	$295.00
Elite 2	$449.00
Elite 3	$559.00

MICRO-SCI

A2	call
A40	call
A70	call

MUSE

Castle Wolfenstein	$20.75
Caverns of Frietag	$20.75
Robot War	$26.75

CONTINENTAL

Home Accountant	$51.75
Book of Apple Software	$16.75

BRODERBUND

Bank Street Writer	$44.75
AE	$24.75
Apple Panic	$21.75
Choplifter	$24.75
David's Midnight	$24.75

SPINNAKER

Kindercomp	$21.75
Story Machine	$23.75
FaceMaker	$23.75
Snooper Trooper	$29.75
Delta Drawing	$34.75

EPYX

Temple of Apshai	$26.95
Star Warrior	$26.95
Crush, Crumble & Chomp	$22.75

ADVENTURE

Saga#1 Adventureland	$29.95
Saga#2 Pirate Adventure	$29.95
Saga#3 Secret Mission	$29.95
Stone of Sisyphus	$24.95

ALIEN GROUP

Atari Voice Box	$99.00
Apple Voice Box	$129.00

commodore

VIC 64	$AVE
1541 Disk Drive	Call
1525 Printer	Call
1530 Datasette	$64.00
1110 8k Ram	$53.75
1211 Super Expander	$53.75
1212 Programmers Ad	$44.75
1213 Vicmon	$44.75
Vic 20 dust cover	$6.99
Vic 64 dust cover	$6.99

MICROSPEC 64

Inventory Mgr	$79.75
General Ledger	$79.75
Payroll Mgr	$79.75
Accounts Rec	$79.75
Accounts Pay	$70.75
Data Base	$69.75
Checkbook Bal	$64.75

HES 64

64Forth R	$55.75
Hesmon R	$29.75
Turtle Graphics R	$49.75
Heswriter R	$38.75
Gridrunner R	$29.75
Attack of Mut Cam R	$34.75
Turtle Tutor R	$29.75
Turtle Trainer R	$29.75
Paint Brush R	$23.75
Benji Space Rescue D	$29.75
Home Manager C/D	$39.75
Time Money Mgr D	$55.75
OmniCalc D	$79.75
Sword Point D	$24.75

EPYX 64

Temple of Apshai	$28.00
Upper Reaches of A	$15.00
Crush Crumble & C	$23.00
Jumpman	$28.00

CARDCO

Cardprinter / LQ1	$499.00
Cardprint DM1	$109.00
5 Slot Expansion 64	$54.00
64 Write NOW	$39.00
64 Mail NOW	$29.00
20 Write NOW	$29.00
64 Keypad	$29.00
Universal Cass. Int	$29.75
Printer Utility	$19.75
6 Slot Expansion	$79.95
3 Slot Expansion	$24.95
Vic 20/64 Printer int	$59.95

BRODERBUND 64

Serpentine R	$26.75
Choplifter R	$32.75
Seafox R	$26.75

PARKER 20

Frogger (ROM)	$33.75
QBert (ROM)	$33.75
Tutankham (rom)	$33.75

SPINNAKER 64

Kindercomp	$21.75
Story Machine	$23.75
Face Maker	$23.75
Snooper Trooper	$29.75
Delta Drawing	$34.75
Shamus II c/d	$24.95
Pinhead c/d	$22.95

QUICK BROWN FOX

QBF Word Processor	$49.95

LJK

Letter Perfect	$105.00
Data Perfect	$95.00

ADVENTURE INTERNATIONAL

S. Adams Adventure	$28.75

VIC-64

Household Finance C/D	$24.75

VIC 20

King Arthurs Heir Cass	$24.75
Monster Maze Rom	$24.75

PERCOM

HARD DISK DRIVES for

APPLE IBM-PC TRS-80*

5 MAGABYTE DRIVE	$1099.75
10 MEGABYTE DRIVE	$1399.75
15 MEGABYTE DRIVE	$1799.75
20 MEGABYTE DRIVE	$2275.75

*Add $30.00 for TRS-80 Drives

SINGLE DRIVE AT88 S1	$379.00
DUAL DRIVE AT88 S2	$599.00
DRIVE with printer port	$CALL
AT88 Expansion Board	$CALL
DUAL HEAD 44S1	$499.00
DUAL DRIVE 44S2	$949.00
SINGLE DRIVE 40S1	$479.00
ADD ON DRIVES	$CALL

Lyco Computer Marketing & Consultants

TO ORDER CALL US

TOLL FREE **800-233-8760**

In PA 1-717-327-1824

16K ATARI RAM	$49.75
32K RAM	$69.75
48K RAM	$99.75
64K RAM	$129.75
128K RAM	$399.75
TECH NOTES	$29.75

BOOK of ATARI SOFTWARE 1983

346 pages $16.75

CX415 HOME FILING MANAGER	$41.75
CXL4007 MUSIC COMP	$33.75
CXL4002 ATARI BASIC	$45.75
CX8126 MICROSOFT	$65.75
CX4119 FRENCH	$45.00
CX4118 GERMAN	$45.00
CX4120 SPANISH	$45.00
CX4108 HANGMAN	$12.75
CX4102 KINGDOM	$12.75
CX4112 STATES	$12.75
CXL4003 ASSEMBLER	$45.75

EASTERN HOUSE

Monkey Wrench 2 ....$52.75

INHOME

Baseball................$29.95

IDSI

Speedway Blast......... $29.95
Pool 1.5................ $26.95

GALAXIAN	$29.75
DEFENDER	$29.75
DIG DUG	$29.75
SPEED READING	$53.75
ATARI WRITER	$54.75
BOOKKEEPER	$102.75
CX4018 PILOT HOME	$54.75
CX 405 PILOT EDU	$91.75
CX404 WORD PRO	$99.75
CXL4013 ASTEROID	$25.75
CXL4020 CENTIPEDE	$29.75
CXL4022 PACMAN	$29.75
CXL4011 STAR RAIDER	$29.75
CXL4004 BASKETBALL	$25.75
CXL4006 SUPER BREAK	$25.75
CXL4008 SPACE INVAD.	$25.75
CX8130 CAVERNS OF M.	$27.75

APX

Eastern Ft.41	$25.50
DeRay Atari	$19.95
Math-Tic-Tac	$15.95
Pres of US	$15.95
3R Math	$19.95
Typo Attack	$24.95
Family Budget	$19.95
F. Cash Flow	$19.95

BRODERBUND

Bank Street Writer D	$44.75
AE D	$24.75
Apple Panic D	$23.75
Choplifter ROM	$32.75
David's Midnight	$24.75
Stellar Shuttle C/D	$18.75
Ft. Apocalypse	$24.75

HES

Gridrunner R	$27.75
Sword Point D	$24.75

INFOCOM

Zork 1/2/or 3 .... $29.95

FIRST STAR

Astro Chase.............. $22.95

RANA DISK DRIVE

1000 .....$295.00

ALIEN GROUP

Voice Box 2 . $99.75

DON'T ASK

Sam	$41.75
Abuse	$15.95
Teleatri	$27.95
Poker Sam	$24.95

Amulet

Nuke Sub	$16.75
Magic Story Book	$24.75
Thunder Island	$13.95

ARTWORX

Hazard Run	$24.95
Hodge Podge	$16.95
S. Poker	$26.95
Bridge 3.0	$18.95

EPYX

Jumpman	$26.97
Temple of Apashi	$27.75
Star Warrior	$27.75

ATARI
Computers for people.

MODEMS

ANCHOR MARK I	$79.00
ANCHOR MARK II	$79.00
HAYES SMART	$239.00
HAYES MICRO II	$309.00
Micro Bit	
MPP-1000	$159.00
NOVATION	
CAT	$144.00
D-CAT	$155.00
J-CAT	$115.00
APPLE CAT II	$279.00
212 APPLE CAT	$589.00

MONITORS

NEC JB1260	$125.00
NEC JB1201	$155.00
NEC TC1201	$315.00
Amdek Color I	$275.00
Amdek 300 Green	$149.00
Amdek 300 Amber	$149.00
Gorilla Green	$99.00

WICO JOYSTICKS

APPLE - VIC - ATARI - TI

Command Control	$20.95
Redball	$22.75
Apple Trackball	$58.00
TRACK BALL	$52.75
EXTENSION CORD	$9.75
APPLE ADAPTOR	$18.95
T.I. ADAPTOR	$9.95
POINT MASTER	$12.75

apple®

Apple is a Registered Trademark of Apple Computer Inc.

APPLE DUMPLING GX	$99.75
APPLE DUMPLING 64...(16 Buffer)	$179.75

INFOCOM

Zork I, II, or III	$26.75
Deadline	$33.75

BLANK DISKETTES

ELEPHANT SS/SD	$18.25
ELEPHANT SS/DD	$21.75
KANGAROO SS/DD with storage case	$24.75
MAXELL MD I	$29.75
MAXELL MD II	$39.75
DISK CASE (holds 10)	$4.95
DISK CASE (holds 50)	$19.75
ROM CASE (holds 10)	$19.75

SANYO

MBC 1000 (with micropro software package)	$1549.00
MBC 1250	$2195.00
MBC4050	$2749.00
EFD 160 Disk	$499.00
5500 Letter Q. Printer	$649.00

PRINTER INTERFACING AVAILABLE

APPLE DUMPLING GX..........$99.75
APPLE DUMPLING 64...(16 Buffer)...$179.75

	ATARI	APPLE	VIC
EPSON	$35.00	$79.00	$65.00
CITOH	$35.00	$79.00	$65.00
NEC	$35.00	$79.00	$65.00
OKIDATA	$35.00	$79.00	$65.00
GEMINI	$35.00	$79.00	$65.00
SMITH CORONA	$35.00	$79.00	$65.00

Atari parallel Printer cable (6') ...$39.95
Microbits Parallel Cable........$89.95

commodore

HES VIC-20

Torg C	$14.75
HES Games I C	$14.75
HES Games II C	$14.75
VIC Fortit Rom	$42.75
HES MON Rom	$28.75
Turtle Graphics Rom	$28.75
HES Writer Rom	$28.75
Shamus Rom	$28.75
Protector Rom	$31.75
Robot Panic Rom	$28.75

DISKETTE SALE

BULK DISKS (qty 150) ....$1.50 ea.
Centron CC-10 (3 blank cassettes)...$3.75

PRINTER RIBBONS AVAILABLE

PROWRITER 2P	$699.00
GEMINI 15	$449.00
PRINTMASTER	$1589.00
SMITH CORONA TP1	$549.00
CITOH 8600B	$1025.00
STARWRITER	$1099.00
OKIDATA 82	SAVE
OKIDATA 83	LOWEST
OKIDATA 84P	PRICE AVAILABLE
OKIDATA 93	SAVE
TRACTOR	$49.75
OKIDATA 92	$SAVE

SAVE on these in-stock PRINTERS

GORILLA GX-100	$199.00
EPSON	$CALL
GEMINI 10	$299.00
PROWRITER	$365.00
NEC 8023	$399.00

MasterCard VISA

TO ORDER

CALL TOLL FREE
800-233-8760
In PA 1-717-327-1824

or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

POLICY

In-stock items shipped within 24 hours of order. Personal checks require four weeks clearance before shipping. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. PA residents add sales tax. All products subject to availability and price change. Advertised prices show 4% discount offered for cash, add 4% for Master Card or Visa. DEALER INQUIRIES INVITED.

cludes checkbook reconciliation and budget categories, and allows changes or deletions in any check or deposit with automatic updating of all balance figures.

CheckEase! is compatible with Commodore's *Personal Finance* program. The VIC version sells for $24.95. The Atari and 64 versions are $29.95 for tape, and $34.95 for disk.

Among other new products available from T & F are *Space Sentinel* and *Slot Trivia*. *Space Sentinel* is an arcade-type game for the Commodore 64. The object is to protect Earth from alien attackers who hurl heat missiles at our polar ice caps. The game sells for $29.95.

Slot Trivia is a trivia question-and-answer game in a slot-machine format. The game, available on disk for Atari computers, includes more than 500 questions in 11 categories.

T & F Software
10902 Riverside Drive
North Hollywood, CA 91602
(213) 501-3856

80-Column Thermal Printer

Alphacom has produced an 80-column thermal printer with graphics capabilities that is designed to be compatible with several computers.

The Alphacom 81 printer includes upper- and lowercase letters and wraparound for lines longer than 80 characters. It prints at 80 characters per second, recognizes standard ASCII control codes, and can print bit-mapped graphics.

Alphacom also has produced a series of interface cables that make the printer compatible with the TRS-80 Color Computer as well as the Commodore 64, VIC-20, Atari, and Apple computers.

The Alphacom 81 will sell for $169.95. Interface cables start at $44.95.

Alphacom, Inc.
2323 South Bascom Ave.
Campbell, CA 95008
(408)559-8000

Alphacom 81 is an 80-column thermal printer with graphics capabilities.

COMPU ENE

CS1 **QUICK BROWN FOX** $55.00
The Word Processor of this decade! For the VIC-20 and C-64.

COMMODORE 64®
$435.00

Plus you receive a free QBF Word Processor valued at $55.00

Write for FREE Catalog!

VIC-20®
$88.00

When you buy our 6 Game Pac or 6 Finance Pack $43.00

C-64 Software

Pet Emulator		$27.95
Editor Pac		67.00
File Pac		32.36
Account Pac		57.00
Farm Management I (Agricultural Software)		47.25
Home Budget		29.95
Stock Investments		76.95
Calc Result		140.00
650Z Professional Development System		27.95
Mail List		34.95
Vic Easy Lesson & Easy Quiz		35.97
Loan Calculator		15.95
Data Files		14.95
Research Assistant 2.0		28.00
Total Label 2.6 (Mailing Lables)	Tape	19.95
	Disk	22.00
Total Time Manager 2.6		37.00
Total Text (Word Processor) 2.6		40.00

C-64 Games

Flight 64 (Flight Simulator)	Tape	$13.95
	Disk	15.95
Gunslinger		13.95
Spellathon		16.95
Motor Mania		25.95
Renassance		25.95
Vic Clowns		25.00
Radar Rat Race		25.00
Jupiter Lander		25.00
Temple of Apshai		39.95
Upper Reaches of Apshai		16.95
Curse of Ra		26.95
Sword of Fargoal		19.95
Jump Man		26.95

VIC-20 & C-64 Hardware

VIC-1541	**Disk Drive**		347.00
VIC-1530	**Datasette**		67.50
VIC-1515	**Printer**		334.95
VIC-1010	**Expansion Module**		139.95
VIC-1311	**Joystick**		9.95
	Wico Joystick		28.00
VIC-1312	**Game Paddles**		19.95
	Telephone Modem		99.95
	Terminal Emulator (64)	Tape	9.95
		Disk	15.95
	40x25 Terminal Emulator VIC		40.95

VIC-1210 **VIC 3K Memory Expander Cart.** 34.95
Plugs directly into the VIC's expansion port. Expands to 8K RAM total.

VIC-1110 **VIC 8K Memory Expander Cart.** 52.50
8K RAM expansion cartridge plugs directly into the VI.

CM102 **24K Memory Expander Cart.** 119.95

VIC-1011A **RS232C Terminal Interface** 39.95
Provides interface between the VIC-20 and RS232 telecommunications modems. Connects to VIC's user port.

PETSPEED — Basic Compiler for Commodore	140.00
Vic Rabbit Cartridge	35.00
CBM 64 Rabbit	35.00
Star G-10 Printer	360.00
Mura Modem	120.00
Smith Corona TP-1 Printer	650.00

CARDCO HARDWARE

CARDBOARD 6 $87.50
An expansion interface for the VIC-20. Allows expansion to 40K or accepts up to six games. May be daisy-chained for more versatility.

CARDBOARD 3 $35.95
Economy expansion interface for the VIC-20.

CARD "?" CARD/PRINT $76.00
Universal Centronics Parallel Printer Interface for the VIC-20 or CBM-64. Use an Epson MX-80 or OKIDATA or TANDY or just about any other.

CARDETTE $30.95
Use any standard cassette player/recorder with your VIC-20 or CBM-64.

LIGHT PEN $29.95
A light pen with six good programs to use with your VIC-20 or CBM-64.

16K Memory Expander $50.50
All CARDCO Products have a lifetime warranty.

COMMODORE SOFTWARE

VIC-1211A **VIC-20 Super Expander** $55.00
Everything Commodore could pack into one cartridge — 3K RAM memory expansion, high resolution graphics plotting, color, paint and sound commands. Graphic, text, multicolor and music modes. 1024x1024 dot screen plotting. All commands may be typed as new BASIC commands or accessed by hitting one of the VIC's special function keys. Includes tutorial instruction book. Excellent for all programming levels.

VIC-1212 **Programmer's Aid Cartridge** $45.99
More than 20 new BASIC commands help new and experienced programmers renumber, trace and edit BASIC programs. Trace any program line-by-line as it executes, pause to edit. Special KEY command lets programmers redefine function keys as BASIC commands, subroutines or new commands.

VIC-1213 **VICMON Machine Language Monitor** $48.99
Helps machine code programmers write fast, efficient 6502 assembly language programs. Includes one line assembler/disassembler.

VIC-20 Software

for Business & Home Applications

6502 Professional Development System		$25.00
Vic Forth (Advance Computer Language)		49.95
Hess Mon (Machine Language Monitor)		34.95
Hess Writer (Word Processor)		34.95
Turtle Graphics		34.95
Total Label 2.1	Tape	17.95
	Disk	22.00
Total Time Manager 2.1	Tape	28.00
	Disk	32.00
Research Assistant 2.0	Tape	28.00
	Disk	32.00
Total Text 2.5	Tape	30.00
Encoder		34.95
Acct. Payable & Receivable	Tape	29.00
	Disk	35.00

VIC-20 Games

Exterminator Plus (Excellent)		$17.95
Anti Matter Splatter (Disaster)		17.95
Rescue From Nufon (Great)		12.95
Tank Wars (War Game)		15.95
Simon (Great for kids)		13.45
Dam Bomber (Avoid the enemy)		13.45
Breakout		7.95
Snack Man (Pac Man)		17.95
Defender on Tri		17.95
Amok	Cart.	23.95
	Cassette	19.94
Starfighter		17.95
Torg		15.95
Gridrunner		34.95
Invasion Orion		20.95
8K Backgammon		19.95

TO ORDER:
P.O. BOX 768
WICHITA, KS 67201
(316) 263-1095

Handling charges $2.00
C.O.D. (Add $2.00)
Personal checks allow 3 week delivery
VIC-20® is a registered trademark of Commodore
Prices subject to change

Timex Tutorial

The Programming Kit I, a BASIC tutorial program for Timex/Sinclair users, is among three new programs produced by Timeworks. *The Programming Kit,* a how-to learning approach to programming, includes an explanation of an eight-step approach to program design.

Family Pak is a set of five 2K programs designed for day-to-day home tasks. The programs are Check Book Balancer, Recipe Recorder, Mini-Money Manager, Homework Helper and Memoboard, a family message center.

Timeworks' third new program, *Scyon's Revenge,* is a deep-space combat game that includes 3-D simulation. The game is provided with a keyboard overlay to give you the feeling of punching command buttons rather than hunting and pecking on a keyboard.

Timeworks, Inc.
405 Lake Cook Road
Building A
Deerfield, IL 60015
(312)291-9200

Kindergarten Gallery

Midwest Software has developed a series of computer programs designed for kindergarten children.

The *Kinder Koncepts* software addresses reading awareness, math concepts, pattern recognition, letters, numbers, colors, and shapes.

All programs follow the same general format, operate with a single keystroke, and keep the necessity for reading to a minimum.

In each program, ten problems are presented. A correct answer is rewarded with a smiling face and a tune. An incorrect answer results in a frown and the chance to try again. Each program has a built-in graph so that progress can be monitored at a glance.

The programs are available on cassette or disk for all Commodore computers except the VIC-20. Disk versions also are available for the Apple II+. The cost is $7.95 per program for cassette or $69.50 for a disk with ten programs.

Midwest Software
Box 214
Farmington, MI 48024
(313)477-0897

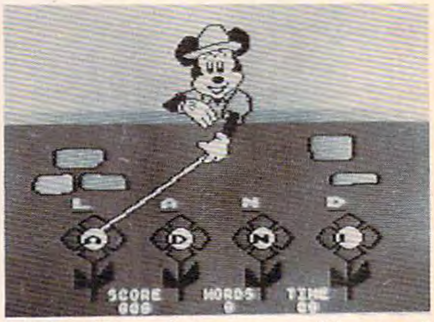
Mickey's New Adventure

Walt Disney Productions has entered the computer software market, and it's making its debut with the help of Mickey Mouse.

Mickey in the Great Outdoors is a pair of interactive adventure games for children seven to ten years old. Mickey Goes Hiking develops grammar and spelling skills by requiring players to finish sentences and unscramble words to help Mickey through his adventure. Mickey Goes Exploring is a similar game, but is based on math skills and equation solving.

Mickey in the Great Outdoors, is being offered only for Atari computers, and distributed through Atari. This program, however, is just the tip of Walt Disney's software iceberg, according to the company.

Plans call for as many as 50 additional Walt Disney programs to be released this year, sup-



Mickey Mouse helps unscramble a word in Mickey in the Great Outdoors.

Products for Commodore, Atari, Apple, and others!

NEW — THE MONKEY WRENCH II
A PROGRAMMERS AID FOR ATARI 800
NEW AND IMPROVED — 18 COMMANDS
PLUGS INTO RIGHT CARTRIDGE SLOT

$59.95

If you are a person who likes to monkey around with the ATARI 800, then THE MONKEY WRENCH II is for you!! Make your programming tasks easier, less time-consuming and more fun. Why spend extra hours working on a BASIC program when the MONKEY WRENCH can do it for you in seconds. It can also make backup copies of boot type cassette programs. Plugs into the right slot and works with ATARI BASIC cartridge.

The MONKEY WRENCH provides 18 direct mode commands. They are: AUTO LINE NUMBERING — Provides new line numbers when entering BASIC program lines. RENUMBER — Renumbers BASIC's line numbers including internal references. DELETE LINE NUMBERS — Removes a range BASIC line numbers.

VARIABLES — Display all BASIC variables and their current value. Scrolling — Use the START & SELECT keys to display BASIC lines automatically. Scroll up or down BASIC program. FIND STRING — Find every occurrence of a string. XCHANGE STRING — Find every occurrence of a string and replace it with another string. MOVE LINES — Move lines from one part of program to another part of program. COPY LINES — Copy lines from one part of program to another part of program. FORMATTED LIST — Print BASIC program in special line format and automatic page numbering. DISK DIRECTORY — Display Disk Directory. CHANGE MARGINS — Provides the capability to easily change the screen margins. MEMORY TEST — Provides the capability to test RAM memory. CURSOR EXCHANGE — Allows usage of the cursor keys without holding down the CTRL key. UPPER CASE LOCK — Keeps the computer in the upper case character set. HEX CONVERSION — Converts a hexadecimal number to a decimal number. DECIMAL CONVERSION — Converts a decimal number to a hexadecimal number. MONITOR — Enter the machine language monitor.

In addition to the BASIC commands, the Monkey Wrench also contains a machine language monitor with 16 commands used to interact with the powerful features of the 6502 microprocessor.

NEW — VIC RABBIT CARTRIDGE AND CBM 64 RABBIT CARTRIDGE

NEW FEATURE!
DATA FILES!

"High Speed Cassette Load and Save!"

$39.95
(includes cartridge and manual)

for VIC

Don't waste your Life away waiting to LOAD and SAVE programs on Cassette Deck.

Load or Save 8K in approximately 30 seconds! Try it—your Un-Rabbitized VIC or 64 takes almost 3 minutes. It's not only fast but VERY RELIABLE.

Almost as fast as 1541 Disk Drive! Don't be foolish — Why buy the disk when you can get the Rabbit for much, much less!

Allows one to APPEND Basic Programs!
Easy to install — just plugs in.
Expansion Connector on rear of the VIC Rabbit.
Works with or without Expansion Memory.
Works with VIC or 64 Cassette Deck.
12 Commands provide other neat features.
Fast Data Files - two data file modes.
Also Available for 2001, 4001, and 8032.

Now for the "64" STCP — 300/1200 Baud
Standard Terminal Communications Package

·PFO· IOD OOA CP<D1>D2 BELL = 12:30:00 10:14:36

Don't settle for non-standard Communications Protocol! Access Micro Net, Source, Bulletin Boards, Local Mainframe, etc.

- Complete Package — Includes RS232 Interface Board and software (does not include modem)
- Communicates in Industry Standard ASCII
- Upload/Download to/from Disk
- Automatic File Translation
- Can be controlled from keyboard or user supplied basic or machine language program

Specify 3.0 or 4.0 ROMS or 8032 Commodore Computer 4040 or 8050 or PEDISK II Disk or CBM64 on 1541.

Price: $129.95

PET BASIC SCROLL PROGRAM
Scroll thru Basic Programs using cursor up/down keys. Specify computer. $6.00 on cassette, $9.00 on diskette.

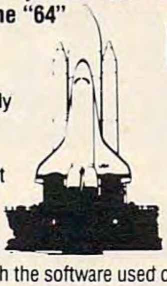
65C02 MAE
Same as our MAE but enhanced for the new 65C02 Opcodes. Turns your computer into a development system for the new ROCKWELL 65C02 Microprocessor. $200.00 — Specify Computer.

6800 CROSS ASSEMBLER
A Cross Assembler based on the MAE that runs on the PET, Apple, or Atari but assembles opcodes for the Motorola 6800 microprocessor. Turns your computer into a development system for the Motorola 6800 Microprocessor. $200.00 — Specify Computer.

ATARI and VIC Cartridges
EHS can supply large quantities of ATARI and VIC Cartridges for software developers. If you need cartridges, call for pricing.

More than just an Assembler/Editor! Now for the "64"

It's a Professionally Designed Software Development System



MAE
for
PET
APPLE
ATARI
~~$169.95~~
New Price
$99.95

Blast off with the software used on the space shuttle project!

- Designed to improve Programmer Productivity
- Similar syntax and commands — No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI
- Coresident Assembler/Editor — No need to load the Editor then the Assembler then the Editor, etc.
- Also includes Word Processor, Relocating Loader, and much more.
- Options: EPROM Programmer, unimplemented opcode circuitry
- STILL NOT CONVINCED: Send for free spec sheet!

ATARI AND PET EPROM PROGRAMMER
Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = $75.00 — ATARI (includes sophisticated machine language monitor) = $119.95

TRAP 65

TRAP 65 is a hardware device that plugs into your 6502's socket. Prevents execution of unimplemented opcodes and provides capability to extend the machines' instruction set.
For PET/APPLE/SYM.
Reduced from $149.95 to $69.95

5¼ INCH SOFT SECTORED DISKETTES
Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers. $22.50/10 or $44.50/20

Prowriter Printer - Excellent dot matrix print parallel = call
Serial = Call IEEE = Call

DC Hayes Smart Modem = $235.00
DC Hayes Micro Modem II = $289.00

Rana Disk Drive - 375
4 Drive Controller - 114

EPROMS 2716 = $4.50 2532 = $7.50
Over 40 Commodore Programs by Baker (on 4040) = $25.00

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA · MasterCard

porting Atari, Radio Shack, Texas Instruments, NEC, and Panasonic computers, and distributed through those companies.

Walt Disney Telecommunications
500 South Buena Vista St.
Burbank, CA 91521
(213)840-1111

Add 64K To Timex/Sinclair

Sunflower Systems has produced a 64K RAM pack for the Timex/Sinclair computers.

The self-contained memory expander plugs into the computer just like the Timex 16K RAM. No additional equipment is needed.

The unit, housed in metal to eliminate radio frequency interference, sells for $119.95 plus $5 for shipping.

Sunflower Systems
718 East Avenue B
Hutchinson, KS 67501
(316)662-2134

Science Fiction Text Adventure

Cyborg is a science fiction text adventure that has no treasures to collect and no score to tally.

The adventure is available from Sentient Software for $34.95 in versions for Atari, Commodore 64, Apple, and IBM.

The game includes character development, animals to talk to, opinions from the Cyborg, and a command structure that allows full sentences.

Sentient Software, Inc.
P.O. Box 4929
Aspen, CO 81612
(303) 925-9293

Turn On The Juice

Tronix, a young company that made its first splash in the VIC-20 market, has added the Commodore 64 to its repertoire.

The company's latest creation is *Juice*, a fast-paced strategy game for the 64 and Atari computers. The hero in *Juice* is Edison, whose job is to complete circuit boards in the face of all the troubles his adversary – Killerwatt – can throw his way.

The game includes six play levels, each with three rounds plus a bonus round. The 32K Atari version sells for $29.95, and the Commodore 64 version sells for $34.95.

Another Tronix offering for the 64 is *Kid Grid*, which previously had been released in an Atari version. In the game, "the Kid" darts around a grid trying to connect all the dots while eluding four bullies. *Kid Grid* sells for $34.95.

In addition to branching into the 64 market, Tronix has bolstered its VIC-20 lineup with the addition of three new cartridge games, *Deadly Skies*, *Scorpion*, and *Gold Fever!*.

Deadly Skies is a shoot-em-up

EXPOTEK

2723 W. Windrose • Suite 3
Phoenix, Arizona 85029

1-800-528-8960

GUARANTEED LOW PRICES

APPLE CARDS
16K RAM — $78   Z80 CARD — $235
Videx Card — $227   Viewmax-80 — $175
Micro Soft Prem. PK — $465

ALTOS
5-15D — $2120   586 — 14 CALL
580-10 — $4695   586-10 — $5698
580-14 — $9395   8600-12 — $8950

CITOH
F10 40cps — $1090   F10 55cps — $1499
1550P — $599   1550CD — $655
8510P — $375   8510BCD — $499

DATASOUTH
DS120 — $595   DS180 — $1169

DIABLO
630RO — $1725   620 — $895

APPLE-LOOK-A-LIKE Call

HAZELTINE
1500 — $995   ESPRIT I — $498

MICRO SCI A2
$265 — Apple Drive/Card — $375

MODEMS
HAYES — MICROMODEM — $263
HAYES — SMARTMODEM — $199
HAYES — 1200 Baud — $499

MONITORS
Amdek 300 — $130   Color I — $295
Amdek Color II — $425   Amber — $145
BMC Green — $85   USI Amber — $149

NORTHSTAR
Advantage — $2250   280A — $1950
5m Byte — $3350   15m Byte — $4499

NEC
8023 — $399   7710 — $1999
3510 — $1375   3550 — $1750

OKIDATA
M92A — $ CALL   M93A — $ CALL
M82A — $ CALL w/Tractor & Grap. $ CALL
M84P — $ CALL   M84S — $ CALL
Pacemark 2350P — $ CALL

TELEVIDEO
802 — $2520   802H — $4450
806 — $4950   800A — $1250
803 — $1890   1603 — $2695

TELEVIDEO
910 — $569   925 — $718
970 — $1040   950 — $899

TI
810 — $1240   820 — $1795

ZENITH
Z19 — $670   Z89 — $2129

SOFTWARE
All Major Brands - 25% off list - $ CALL

DISKETTES/BOXES
Elephant — $19 Scotch — $25 Dysan — $35

All Prices Subject To Change

Customer Service
602-863-0759

ACCOLADE COMPUTER PRODUCTS

ATARI 850 Interface	$185.00
ATARI 410 Recorder	75.00
ATARI Disk Drive	449.00
PERCOM Double Density	599.95
EPSON Mx 80 ft. w/graftrax	$479.95
EPSON Mx 100 ft. w/graftrax	697.50
EPSON Fx 80 (160 CPS)	595.00
GEMINI 10x (120 CPS)	329.95
GEMINI 15	649.95
OKIDATA 82A (120 CPS)	459.95
DAISY Silver Reed	799.95
BMC (12" Green)	99.95
BMC (12" Color)	342.50
ATARI Writer	$77.50
Programmer's Kit	49.95
Family Finance	38.95
Juggles House	22.95
Juggles Rainbow	22.95
My First Alphabet	27.95
ATARI Basic	45.95
Missile Command	28.95
Pacman	34.95
Centipede	34.95
Qix	34.95
Dig Dug	34.95
Defender	34.95
Galaxian	34.95
Donkey Kong	38.75
E.T. Phone Home	38.75
Eastern Front	38.75
Miner 2049er	37.50
Zaxxon	29.95
Frogger	26.95
Choplifter	33.95

Dealer Inquiries Welcome
WE WILL TRY TO MEET OR BEAT ANY ADVERTISED PRICE

ACCOLADE COMPUTER PRODUCTS
4858 Coronado Ave.
San Diego, Ca 92107
(619) 223-8599

Calif. Res. add 6% Sales Tax
Shipping & Handling $4.00
Foreign Orders add $5.00
Payment in U.S. Funds only
COD Charge $1.65
COD's, Cashier Check or Money Orders
Personal checks must take 2 weeks to clear

WHY SPEND MORE?
SUPERBYTE BUSINESS PACK
ON DISK FOR THE
Commodore 64*
OR
Vic 20 (8K)*

INCLUDED ARE...

***THE ACCOUNTANT**
General Ledger, Income Statement & Balance Sht.
***ACCOUNTS RECEIVABLE/PAYABLE**
Journal for current & paid accounts
***THE EDITOR**
Full feature word processor
***SPREAD SHEET CALC.**
Complete spread sheet/calculating program
***BUSINESS INVENTORY**
Inventory control system
***CHECKBOOK MATE**
Checkbook maintenance & writer
***THE MAILMAN**
Address file with sorting
***PLUS 6 UTILITY PROGRAMS**
Profit Margin/Bus. Calendar & Data Base
Pert Model/Linear Regression/Depreciator
Amortization

FULL PRINTER CAPABILITIES
COMPLETE REFERENCE MANUAL

ONLY $100.00
By Check, Charge or COD ($ Shipping)

SUPERBYTE SOFTWARE
Master Card   2 CHIPLEY RUN   Visa
WEST BERLIN, NJ 08091
(609) 346-3063

*Trademarks of Commodore Business Machines

CARDRAM 16

16 K Memory Expansion Cartridge for the VIC-20® Personal Computer

FEATURES:
- Provides the equivalent of two 8K RAM cartridges.
- Each 8K is individually switch selectable by block.
- Opening in case to provide easy switch access.
- Uses 8 low power CMOS 2K x 8 static RAM chips.
- High quality glass/epoxy circuit board.
- Gold plated contacts.
- Individually tested.
- Made in the U.S.A.

$50.50

COMPU SENSE

TO ORDER:
P.O. BOX 768
WICHITA, KS 67201
(316) 263-1095

AMERICAN EXPRESS ® VISA MasterCard

Handling charges $3.00
C.O.D. (Add $2.00)
Personal checks allow 3 week delivery
VIC-20® is a registered trademark of Commodore
Prices subject to change

game in which the player, equipped with a squadron of helicopters, takes on a military base. *Gold Fever!* is a maze game in which a prospector faces runaway boxcars, boulders, claim jumpers, and a limited supply of oxygen. The object of *Scorpion* is to keep the snake alive and fed in the midst of a world filled with dragons, frogs, Venus's-flytraps, stalkers, worms, and pods.

Each of the VIC-20 games sells for $39.95.

Tronix Publishing, Inc.
8295 S. La Cienega
Inglewood, CA 90301
(213)215-0529

A Program To Remember

Memory Trainer, an interactive program to teach memory improvement, is available from Einstein.

The program, which is available for the Apple, Atari 800, and Commodore 64, is based on memory improvement research from the past 100 years.

MemoryTrainer includes five lessons in a three-disk package that sells for $89.95. The lessons teach the ability to remember faces, dates, telephone numbers, lists, and quotations, and to use association as a memory tool.

The package also includes Memory Mix, a game that provides practice for each memory skill.

The Einstein Corporation
11340 W. Olympic Blvd.
Los Angeles, CA 90064
(213) 477-6733

VIC Wafer Storage

A low-cost micro-wafer storage device for the VIC-20 will be available later this year from Unitronics, through a licensing agreement with Vadem, the unit's builder.

The V-20 Expander is described as an inexpensive alternative to floppy disk storage for low-end computers. It reads or writes data to small tape cassettes at a speed approaching that of disks.

The device, which measures 5x6x7 inches, plugs into the VIC's cartridge expansion slot. It includes a 10K RAM memory expansion board, a 64K data wafer and high-speed micro-wafer drive, a filing system, and VWOS – the Vadem Wafer Operating System.

Because VWOS is able to access the computer's memory bus directly, rather than through a serial port, the V-20 is able to improve on the data transfer rates of existing micro-wafer devices.

The expander is expected to sell for about $100.

Vadem
3517 Ryder Street
Santa Clara, CA 95051
(408) 738-0571

Stand-On Game Controller

The Joyboard, a game controller that involves the whole body rather than just the hands, has been introduced by Amiga for the VIC-20 and Atari computers.

The Joyboard, which comes with *Mogul Maniac*, a skiing simulation game, will sell for about $50. Other games designed for use with the Joyboard – *Surf's Up* and *Off Your Rocker* – will cost about $20.

The Joyboard also can be used with many existing maze-type games to provide a different challenge, or, for shoot-em-up games, a conventional joystick can be plugged into the Joyboard to control firing, while your feet control direction.

Amiga also has produced a version of its Power-Stick joystick for the TI-99/4A. This includes two controllers hard-

HARMONY VIDEO & COMPUTERS
800-221-8927

JVC • PANASONIC • SONY PROFEEL • ZENITH • HITACHI • QUASAR

COMMODORE	
VIC 20	89.95
COMMODORE	199.95
VIC 2N DATASETTE	59.95
COMMODORE DISC DRIVE	209.95
1525B PRINTER	209.95
COMMODORE MONITOR	249.95
VIC TELEPHONE MODEM	69.95
RS 232 TERMINAL INTERFACE	CALL
IGEE-488 INTERFACE	CALL
VIC 8K MEMORY PAC	39.95
VIC 16K MEMORY PAC	89.95
MOTHER BOARD	89.95
VIC 3K SUPER EXPANDER	
VIC PROGRAMMER'S AID	CALL
VIC MON	CALL
INTRO TO BASIC I & II	69.95

APPLE 2 PLUS
999.95

APPLE 2E
CALL

EPSON
MX 80 PRINTER
399.95

EPSON
FX 80 PRINTER
599.95

ATARI	
ATARI 400 W16K	129.95
ATARI 800 W48K	249.95
ATARI 1200 XL W64K	369.95
ATARI 410 RECORDER	76.95
ATARI 1010 RECORDER	77.95
ATARI 810 DISC DRIVE	389.95
ATARI 830 ACOUSTIC TEL. MODEM	159.95
ATARI 850 INTERFACE	169.95
ATARI 822 THERMOL PRINTER	CALL
ATARI 1020 40 COL. PRINTER	219.95
ATARI 1025 80 COL. PRINTER	379.95
ATARI 16K MEMORY EXPANDER	89.95
ATARI WORD PROCESSOR	69.95
ATARI BASIC REF. GUIDE	CALL
ATARI PROGRAMMER KIT	CALL
ATARI ENTERTAINER KIT	29.95

G.E. • SANYO • SONY PROJECTION • TDK • MAXELL • FUJI • SCOTCH • SONY • RCA

WE CARRY A FULL LINE OF SOFTWARE FOR ATARI AND COMMODORE
CALL FOR LOWEST PRICES

ALL SONY T.V.'s IN STOCK CALL FOR LOWEST PRICES $ $

VCR PRICES TOO LOW TO PRINT!

ALL VCR's IN STOCK CALL FOR LOWEST PRICES $ $

VIDEO TAPE (By case of 10 only)

	VHS T120	VHS T160	HIGH GRADE	BETA L500	BETA L750	BETA L830
TDK	8.85	14.75	10.75	6.50	7.99	10.25
FUJI	9.65	14.75	11.75	6.50	7.99	
MAXELL	8.40	13.50	10.25	6.50	7.99	10.25
MEMOREX	8.75	13.50	10.50	6.50	7.99	10.25
SCOTCH	8.50	13.50	10.25	6.50	7.99	
SONY			10.25	6.99	8.40	10.25
JVC	9.50	13.50	11.50			
RCA	8.90					
PANASONIC	8.50					

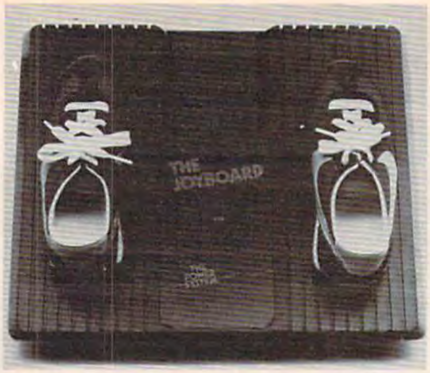
TDK PROFESSIONAL T120	15.95
MAXELL PROFESSIONAL L750 OR T120	15.95
MEMOREX PROFESSIONAL	16.95

FOR THE LOWEST PRICES ON VIDEO CALL 800-221-8927

To order simply dial toll free 800-221-8927 with your Master Card or VISA and your order will arrive via UPS or send certified check or money order only to: HARMONY VIDEO AND ELECTRONICS, 2357 Coney Island Ave., Brooklyn, N.Y. 11223, and add approximate shipping postage and insurance charges. Credit cards for phone order only. For customer service please dial (212) 627-8960. All pries and availability subject to change without notice. All orders shipped out of state. Dealer inquires invited!!! For sales info dial (212) 627-6989. Open Sun. 10-4 Mon.-Thur. 9-7 Fri. 9-3.

800-221-8927

COMMODORE ATARI APPLE TEXAS INSTRUMENTS EPSON



The Joyboard from Amiga transfers videogame control from your hands to your entire body.

wired into a single plug to fit the TI's single jack configuration. The pair will sell for about $20.

Amiga Corporation
3350 Scott Boulevard, Building 7
Santa Clara, CA 95051
(408) 748-0222

Add-On Adventure

Infocom, the company that produced the *Zork*, the popular underground fantasy series, is launching a new trilogy that will take the adventurer into the world of magical powers and perilous predicaments.

The first in the new series, *Enchanter*, scheduled to be available by mid-September, is a prose adventure that takes place in an abandoned castle. The passage of time plays an important role in the game: you must eat, drink, and sleep regularly, or your powers will fail.

The game, which will be available in versions for most popular microcomputers, will retail for $49.95 to $59.95.

Infocom, Inc.
55 Wheeler St.
Cambridge, MA 02138
(617) 492-1031

Battle Games

Thorn EMI Home Video has released a pair of new battle games, *Orc Attack* and *Fourth Encounter*.

In *Orc Attack*, the player must defend his castle against the Orcs, who erect ladders and scale the castle walls under cover of a volley of crossbow bolts from their archers. The game, which is available for the Atari 400 and 800, sells for $39.95.

Fourth Encounter is a cartridge game for the VIC-20. The challenge here is to save a planet from an invasion of aliens, who bring with them slavery, death, and destruction. *Fourth Encounter* is available for $39.95. In addition to these two new games, Thorn EMI has converted a couple of other games into new formats. *Submarine Commander*, previously released as an Atari game, is now available for the VIC, and *River Rescue* now can be played on the Atari.

Thorn EMI Home Video
1370 Avenue of the Americas
New York, NY 10019

APPLE COUNTRY LTD.®

YOU'RE GONNA LOVE THESE ROCK BOTTOM PRICES!

ACTION	99.00	68.95
ADVANCED MUSIC SYSTEM	29.95	21.95
AE	34.95	24.49
ANTI-SUB PATROL	29.95	21.95
APPLE CIDER SPIDER	39.95	27.95
ARCADE MACHINE	59.95	41.49
ARMOR ASSAULT	39.95	27.95
ATARI BOOKKEEPER KIT	249.95	179.95
ATARI MACRO ASSEM/TEXT ED	89.95	64.49
ATARI MICROSOFT BASIC II	89.95	64.49
ATARI MUSIC COMPOSER-ROM	39.95	29.95
ATARI PILOT EDUCATION-ROM	129.95	94.95
ATARI PROGRAMMER KIT	59.95	45.95
ATARI SPEED READING	74.95	54.95
ATARI TECH USER NOTES	29.95	21.95
ATARI TOUCH TYPING	24.95	17.95
ATARI WRITER	79.95	56.95
BANDITS	34.95	24.95
BANK STREET WRITER	69.95	49.95
BASIC COMPILER	99.95	68.95
BATTLE FOR NORMANDY	39.95	27.95
BATTLE OF SHILOH	39.95	27.95
BILESTOAD	29.95	21.95
THE BLADE OF BLACKPOOL	39.95	27.95
BOOK OF ATARI SOFTWARE'83	19.95	14.95
BUG/65	34.95	24.49
C/65	80.00	54.95
CAP'N COSMO	34.95	18.95
CASTLE WOLFENSTEIN	29.95	21.95
CENTIPEDE-ROM	44.95	31.95
CHOPLIFTER - ROM	44.95	31.95
COCO	49.95	34.49
COCO II	39.95	27.95
COLOR PRINT	39.99	27.95
COMMUNICATOR KIT	279.95	214.95
CONVERSATIONAL FRENCH	59.95	42.95
CONVERSATIONAL SPANISH	59.95	42.95
THE COSMIC BALANCE	39.95	27.95
CROSSFIRE-ROM	44.95	29.95
CYTRON MASTERS	39.95	27.95
DA FUZZ	44.95	31.95
DATA PERFECT	99.95	74.95
DATALINK	39.95	27.95
DAVID'S MIDNIGHT MAGIC	34.95	24.49
DEADLINE	49.95	34.49
DEFENDER	44.95	31.95
DE RE ATARI	19.95	14.49
DIG DUG	44.95	31.95
DISK MANAGER	29.95	21.95
DISK WORKSHOP	34.95	24.49
DISKETTE INVENTORY SYSTEM	24.95	17.49
DISKEY	49.95	34.49
DISKSCAN	40.00	28.00
DISKWIZ	29.95	21.95
DIVISION I	44.95	31.95
DNIEPER RIVER LINE	30.00	21.95
DODGE RACER	34.95	24.49
EASTERN FRONT (1941)	29.95	21.95
EDIT 6502-ROM	199.95	144.95
THE EDUCATOR KIT	164.95	109.95
FACEMAKER	34.95	24.49
FANTASTIC VOYAGE-ROM	34.95	26.49
FILE MANAGER +	99.95	68.95
FINANCIAL WIZARD	59.95	39.95
FLAME LORDS	34.95	24.95
FLASH GORDON-ROM	34.95	26.95
FLIP OUT	29.95	21.95
FIREBIRD-ROM	39.95	27.95
FORT APOCALYPSE	34.95	24.49
FROGGER	34.95	24.49
GALAXIAN	44.95	31.95
GHOST ENCOUNTERS	29.95	21.95
GHOSTLY MANOR	24.95	17.95
GLOBE MASTER	29.99	21.95
GORF-ROM	44.95	28.49
GRAPHIC GENERATOR	24.95	17.49
GRAPHIC MASTER	39.95	27.95
THE HOME ACCOUNTANT	74.95	54.95
JOURNEY TO THE PLANETS	29.95	21.95
JUMPMAN	39.95	27.95
KIDS AND THE ATARI-BOOK	19.95	13.95
KINDERCOMP	29.95	21.95
K-RAZY SHOOTOUT-ROM	49.95	34.49
KING ARTHUR'S HEIR	29.95	21.95
LEGIONNAIRE	35.00	25.00
LETTER PERFECT (40/80)	149.95	109.95
LETTER PERFECT UTILITY	29.95	21.95
LUNAR LEEPER	29.95	21.95
MASTER MEMORY MAP	6.95	4.95
MASTER TYPE	39.95	27.95
MATING ZONE	29.95	21.95
MATCH BOXES	29.95	21.95
MAURAUDER	34.95	24.49
MAX/65 (WITH OS/A+)	80.00	54.95
MICKEY IN GREAT OUTDOORS	49.95	35.95
MINER 2049'ER-ROM	49.95	34.49
MISSILE COMMAND-ROM	34.95	26.49
MONSTER SMASH	29.95	21.95
MORL	44.95	31.95
NAUTILUS	34.95	24.49
NECROMANCER	34.95	24.49
THE NIGHTMARE	29.95	21.95
OPERATION WHIRLWIND	34.95	24.49
OS-A+ & BASIC A+	80.00	54.95
PAINT	39.95	29.95
PINBALL	29.95	21.95
P.M.ANIMATOR	34.95	24.49
POKER-S.A.M.	24.95	17.95
PREPPIE II	34.95	24.49
PYRAMID PUZZLER	44.95	31.95
QIX	44.95	31.95
RASTER BLASTER	29.95	21.95
READING FLIGHT	44.95	31.95
ROUNDABOUT	29.95	21.95
SAMMY LIGHTFOOT	34.95	24.49
SAMMY THE SEA SERPENT	23.95	16.95
SEA DRAGON	34.95	24.49
SEA FOX	29.95	21.95
747 LANDING SIMULATOR	22.95	16.95
SHADOW WORLD	34.95	24.49
SHAMUS-ROM	44.95	31.95
SNEAKERS	29.95	21.95
SOFTWARE AUTO-MOUTH (SAM)	59.95	41.49
SPEED READ PLUS	59.95	41.49
SPEEDWAY BLAST	29.95	21.95
SPELLING BEE GAMES	39.95	27.95
SPELL WIZARD	79.95	54.95
STARCROSS	39.95	27.95
STAR MAZE	44.95	31.95
STAR RAIDERS-ROM	44.95	31.95
STAR WARRIOR	39.95	27.95
STORY MACHINE	34.95	24.49
SUPERMAN III	49.95	35.95
SURVIVOR	34.95	24.49
SWIFTY TACH MASTER	29.95	21.95
SYN ASSMBLER	49.95	34.49
TELECOM	69.95	49.95
TELETALK	49.95	36.95
TELETARI	39.95	27.95
TEMPLE OF APSHAI	39.95	27.95
TEXT WIZARD I	99.95	68.95
TIGERS IN THE SNOW	39.95	27.95
TIME WISE	29.95	21.95
TYPE ATTACK	39.95	27.95
TUTTI FRUTTI	24.95	17.95
VC	25.00	17.49
VISICALC	199.00	149.95
WARLOCK'S REVENGE	34.95	24.49
WAY OUT	39.95	27.95
WIZARD&PRINCESS HIRES ADV	32.95	22.95
WIZARD OF WOR - ROM	44.95	31.95
YOUR ATARI COMPUTER-BOOK	16.95	12.95
ZAXXON	39.95	27.95
ZORK I II OR III	39.95	27.95

AMDEK COLOR I PLUS MONITOR		319.00
APPLE EMULATOR	CBM64	79.50
CARDBOARD (6 CART + RESET)	VIC-20	70.00
THE CARD? (FOR PARALLEL PRTR)	CBM64&VIC-20	58.00
CARDETTE (CASSETTE INTFCE)	CBM64&VIC-20	22.00
CARDAPTER/1 ATARI 2600 INTFCE	VIC-20	50.00
CARDWRITER/1 LIGHT PEN W/CASS	CBM64&VIC-20	16.50
COMMODORE 64 HOME COMPUTER	CBM64	369.50
COMMODORECOLOR PLOTTER	CBM64&VIC-20	179.50
COMMODORE 1530 DATASETTE	CBM64&VIC-20	65.00
COMMODORE 1541 DISK DRIVE	CBM64&VIC-20	329.50
COMMODORE 1525 PRINTER	CBM64&VIC-20	329.50
COMMODORE 1600 MODEM	CBM64&VIC-20	95.50
COMMODORE COLOR MONITOR	CBM64&VIC-20	269.50
COMMODORE 1650 AUTODIAL MODEM	CBM64&VIC-20	169.50
MICRO EXPANSION CHASSIS	CBM64	35.75
VIDEO PAK 80 (80 COLUMN)	CBM64	129.00
VIDEO PAK 80 WITH CP/M	CBM64	219.00
ATARI 800 48K	$489-$100 REBATE=	389.00
ATARI 1200 64K	$519-$100 REBATE=	419.00
ATARI NUMERIC KEYPAD		94.95
ATARI PROGRAMMER KIT		49.95
410 RECORDER		72.95
810 DISK DRIVE		424.95
850 INTERFACE MODULE		169.95
C. ITOH PROWRITER I		394.95
C. ITOH PROWRITER II		649.95
C. ITOH STARWRITER		1325.00
CASSETTE 'N CARTRIDGE FILE		21.95
ELEPHANT SS/SD DISK		10/16.95
ELEPHANT SS/DD DISK		10/19.95
ELEPHANT DS/DD DISK		10/25.95
EPSON FX-80 W/TRACTOR		LOW!!
EPSON MX-100 F/T		LOW!!
FLIP'N-FILE		21.95
GEMINI-10 PRINTER		LOW!!
GEMINI-15 PRINTER		LOW!!
IN HOME 400 KEYBOARD		94.95
INTEC 32K RAM		69.95
INTEC 48K RAM		119.95
MOSAIC 64K RAM SELECT		149.95
HAYES SMARTMODEM 300 BD		194.95
HAYES SMARTMODEM 1200 BD		499.95
NEC 8023 PRINTER		459.95
NEC 12" HIRES GREEN SCRN		149.95
NEC 12" ECONO GREEN SCRN		79.95
NOVATION J-CAT MODEM		109.95
NOVATION SMART-CAT 103		179.95
NOVATION SMART-CAT 212		424.95
PERCOM SS/SD/1DR (88K)		419.95
PERCOM SS/DD/1DR (176K)		539.95
PERCOM SS/DD/2DRS (352K)		859.95
PERCOM DS/DD/1DR (352K)		649.95
PERCOM DS/DD/2DRS (704K)		939.95
SIGNALMAN MK II MODEM		79.95
USI 12" AMBER MONITOR		159.95
VERSAWRITER GRAPH TABLET		239.95
WICO JOYSTICK		21.95
WICO REDBALL JOYSTICK		24.95
WICO DELUXE JOYSTICK		29.95
WICO TRACKBALL		49.95
WICO 12 FT EXTENSION CORD		6.95

commodore

AGGRESSOR	CART/VIC20	27.50
ALIEN SOCCER	CASS/VIC20	10.25
ANDROMEDA CONQUEST	CASS/C64	12.50
APE ESCAPE	DISK/VIC20	27.50
BLADE OF BLACKPOOLE	DISK/C64	27.50
CAVE-IN	CART/VIC20	27.50
CHECKBOOK	CASS/VIC20	13.75
COMPUTER FOOTBALL STRATEGY	CASS/C64	11.00
COMPUTER STOCKS & BONDS	CASS/C64	13.75
CONCENTRATION	CASS/VIC20	11.00
CRITICAL MASS	DISK/C64	27.50
CROSSFIRE	CASS/VIC20	21.00
DEADLINE	DISK/C64	34.50
DEADLY DUCK	CART/VIC20	24.00
DEADLY SKIES	CART/VIC20	27.50
DEMON ATTACK	CART/VIC20	27.50
DRELBS	CASS&DISK/64	24.00
FACEMAKER	DISK/C64	24.00
FAST EDDY	CART/VIC20	27.50
FINAL ORBIT	CART/C64	24.00
FT. APOCALYPSE	CASS&DISK/64	24.00
FROGGER	DISK/C64	24.00
FRUIT FLY	CASS/VIC20	9.00
FUEL PIRATES	CASS/VIC20	11.00
GALACTIC BLITZ	CASS/VIC20	17.25
GALACTIC CROSSFIRE	CASS/VIC20	10.25
GOLD FEVER	CART/VIC20	27.50
GOLD MINE	DISK/VIC20	27.50
GRIDRUNNER	CART/64/VIC20	27.50
HANGMAN	8KCASS/VIC20	6.25
HARRIER	CASS/VIC20	24.00
HESMON MONITOR	CART/64&VIC20	27.50
HESWRITER W/P	CART/VIC20	27.50
HESWRITER 64	CART/C64	31.00
HEY DIDDLE DIDDLE	DISK/C64	21.00
IN SEARCH OF THE MOST AMAZING THING	DISK/C64	27.50
JUMP MAN	DISK/C64	27.50
KINDERCOMP	DISK/C64	21.00
MARTIAN RAIDER	DISK/VIC20	13.75
METEOR	8KCASS/VIC20	7.50
MONSTER MAZE	CART/VIC20	27.50
MULTISOUND SYNTH.	DISK/VIC20	13.75
NUMBERCHASER	16KCASS/VIC20	17.25
NUMBER CRUNCH	CART/VIC20	27.50
NUMBER GULPER	8K CASS/VIC20	17.25
PHAROH'S CURSE	CASS&DISK/64	24.00
PREDATOR	CART/VIC20	27.50
PROTECTOR	CART/VIC20	30.25
QUICK BROWN FOX W/P	CART/VIC20/64	44.75
RAID ON ISRAM	CASS/VIC20	13.75
REAGANOMICS	CART/VIC20	27.50
REPTON	DISK/C64	27.50
RESCUE AT RIGEL	16KCASS/VIC20	21.00
RETRO BALL	CART/C64	27.50
RICOCHET	8K CASS/VIC20	13.75
ROBOT PANIC	CART/VIC20	27.50
SCORPION	CART/VIC20	27.50
SHAMUS	CART/VIC20	27.50
SHARK TRAP	DISK/VIC20	13.75
SIDEWINDER	8K CASS/VIC20	21.00
SIMON	CASS/VIC20	11.00
6502 PROF DEV SYS	CASS/64/VIC20	21.00
SKI RUN	8KCASS/VIC20	12.50
SNAKE BYTE	CART/C64	24.00
SNOOPER TROOPS #1	DISK/C64	27.50
SPACE ATTACK	8K CASS/VIC20	6.25
SPIDER CITY	CART/64/VIC20	27.50
SPORTS SEARCH	CASS/VIC20	13.75
SQUISH'EM	CART/64/VIC20	24.00
STARCROSS	DISK/C64	27.50
SURVIVOR	CASS&DISK/C64	24.00
SWARM!	CASS/VIC20	21.00
SWORD OF FARGOAL	16KCASS/VIC20	21.00
SYN THE SOUND/MUSIC	CART/VIC20	41.00
TELENGARD	CASS/C64	15.75
TORG	CASS/VIC20	12.25
TURMOIL	CART/VIC20	27.50
TURTLE GRAPHICS	CART/VIC20	27.50
TURTLE GRAPHICS II	CART/C64	41.00
TYPE ATTACK	CART/64/VIC20	27.50
THE VEIN GAME	DISK/VIC20	24.00
VIC FORTH	CART/VIC20	41.25
VIC-MEN	8KCASS/VIC20	15.00
VIC MUSIC COMPOSER	CART/VIC20	30.00
VICTREK	CASS/VIC20	12.25
VIC VANGO	CASS/VIC20	9.00
VIDEO SEARCH	CASS/VIC20	13.75
WORD SEARCH	CASS/VIC20	13.75
ZORK I II OR III	DISK/C64	27.50

ATARI PERCOM RanaSystems Hayes FRANKLIN NEC IBM MOSAIC 64K RAM SELECT EAGLE TEXAS INSTRUMENTS COLUMBIA DATA PRODUCTS INC OKIDATA

FOR FASTEST DELIVERY: CASHIER'S CHECK OR VISA/MASTERCARD (NO EXTRA CHARGE FOR CARDS. **INCLUDE NUMBER, EXPIRATION DATE, NAME, ADDRESS & PHONE**). PERSONAL CHECK ALLOW 2 WEEKS TO CLEAR. PURCHASE ORDER MUST INCLUDE CHECK. **SHIPPING & HANDLING:** CONTINENTAL U.S. 5% ($5 MIN), U.P.S. **STREET ADDRESS REQUIRED;** APO FPO ALASKA HAWAII & MONITORS 5% ($10 MIN); FOREIGN 15% ($15 MIN). **INCLUDE PHONE NUMBER WITH ALL ORDERS.** ALL ITEMS ARE NEW WITH MANUFACTURER'S WARRANTY. APPLE COUNTRY, LTD. CANNOT GUARANTEE THE MERCHANTABILITY OF ANY PRODUCT. PRICES ARE SUBJECT TO AVAILABILITY & CHANGE WITHOUT NOTICE. DUE TO OUR LOW PRICES, ALL SALES ARE FINAL. RETURNED MERCHANDISE IS SUBJECT TO A RESTOCKING FEE & MUST COME IN ORIGINAL UNDAMAGED CARTON WITH RMA NUMBER. NO SOFTWARE EXCHANGES. CALIFORNIA RESIDENTS ADD 6% TAX. SEND $1 FOR NEW FALL CATALOG (GOOD TOWARD FIRST PURCHASE). APPLE COUNTRY, LTD. IS A **DISCOUNT MAIL ORDER HOUSE FOR THE MICROCOMPUTER INDUSTRY** & IS A CALIFORNIA CORPORATION NOT AFFILIATED WITH APPLE COMPUTER INC. APPLE IS TRADEMARK OF APPLE COMPUTER INC. 6.1.83

Call us... we can help! (619) 765-0239

P.O. Box 1099,
Julian, Calif. 92036

Apple Country, Ltd. is a DISCOUNT MAIL ORDER HOUSE for the micro computer industry and is a California corporation not affiliated with Apple Computer Inc. Apple is a trademark of Apple Computer Inc.

Ancient Game On Computer

A computer version of the ancient strategy game, *GO*, is available from Hayden Software.

The program is designed to teach the novice as well as hone the skills of the experienced *GO* player.

The object of *GO* is to encircle, trap, and capture the computer's playing pieces while defending against the computer's attempts to trap you.

Versions of the game are available for $34.95 for Apple and $29.95 for Atari.

Also available from Hayden is an action-packed maze game, *Wargle*. In the game, priced at $34.95 for Apple and Atari, the player must take evasive action while using a laser beam to eliminate the Wargles. The game includes seven mazes and six levels.

Hayden Software Company
600 Suffolk St.
Lowell, MA 01853
(617) 937-0200

Printer Buffer

The Microbuffer In-Line, a printer buffer with a memory expandable to 256K, is available from Practical Peripherals.

The buffer is compatible with almost any serial or parallel printer, modem, word processor, or computer equipped with an RS-232 serial output device.

The buffer includes a COPY feature that allows printing up to 255 copies of any document with the touch of a button, and the PASS feature allows data to be routed around the buffer when appropriate.

Microbuffer In-Line with 32K memory is available for $299. A 64K version sells for $349, and additional memory expansion is available for $179 per 64K.

Practical Peripherals, Inc.
31245 La Baya Drive
Westlake Village, CA 91362

Apple In Space

Mission: Escape! is an arcade-type space game for the 64K Apple II computer.

To play, you pilot your shuttlecraft through asteroids and meteors to save the inhabitants of the 12 planets in the Galaxy of Appel, which is about to self-destruct because of violent volcanic activity.

The hazards increase with each planet you attempt to evacuate.

MicroSparc, Inc.
10 Lewis St.
Lincoln, MA 01773
(617) 259-9710

T/S Text Editor, Input Utility

An input utility program and a text editor for the Timex/Sinclair are available from SyncMaster. Each program sells for $14.95 plus $1 for shipping.

The *Screen Machine* is a 1.5K machine language utility that allows inputs anywhere on the screen. The routine performs length verification of responses, compacts numbers, and allows dates in MMYY or MMDDYY formats.

The *Vu-Write Text Editor* is a menu-driven program for machines with at least 16K RAM.

The program includes insert, delete, change, and save functions. It leaves 11K available for text and allows line length to be set by the user.

Vu-Write Text Editor is written to be compatible with the ZX81 printer, but the program is listable and can be modified for any printer.

SyncMaster
P.O. Box 511
Oak Ridge, NC 27310
(800) 334-0854, or
(919) 643-7120 in North Carolina

Alphabet Zoo

Spinnaker Software is scheduled to release another game in its early learning series this fall.

The game, *Alphabet Zoo*, is designed to teach three- to eight-year-olds the relationship between letters and sounds. It incorporates two maze games, colorful graphics, and sound.

Alphabet Zoo will be available on disk for Apple, Atari, IBM, and Commodore 64 computers. Cartridge versions will be available for the 64 and Atari.

Another fall offering from Spinnaker is *Cosmic Life*, a computer learning game in the style of checkers and Go. It is designed to strengthen planning, strategy, and pattern recognition skills.

Cosmic Life will be available in cartridge for the Atari and Commodore 64.

Spinnaker Software
215 First St.
Cambridge, MA 02142
(617) 868-4700

EPROM Programmer

Gloucester Computer has produced a Commodore 64 version of its VIC Promqueen EPROM programmer.

The PQ/64 cartridge includes a 28 pin Textool ZIF socket, a matrix switch EPROM type selection that accommodates all JEDEC pinout devices that work on 5 volts, RS-232 communications software, faster burn process, a burn test procedure, and a 24K workspace.

The PQ/64 is expected to retail for $299.

Gloucester Computer, Inc.
One Blackburn Center
Gloucester, MA 01930
(617) 283-7719