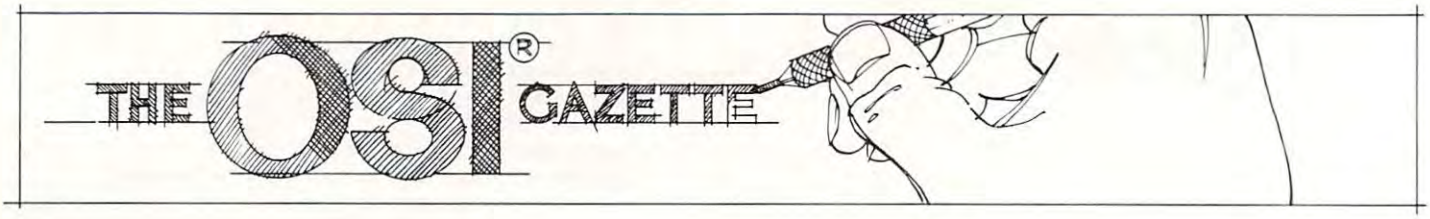


OSI Relocation Or What's NEW?

Mark Guzdial
Royal Oak, MI

After reading Elizabeth Deal's article on "Relocation of BASIC Programs on the PET" in **COMPUTE!** #13, I tried to implement the program relocation on my OSI C1P. After doing some POKEing around, both on and off-line, I found not only how very simple it was to do on an OSI machine, but also some observations about how the NEW command works.

Ms. Deal broke the relocation process down into four steps:

1. Set up one or more partitions
2. Adjust all BASIC pointers
3. Change the tape header information
4. Correct the forward pointers in the relocated program.

Since OSI machines store programs on tape in ASCII (not in the internal format) steps (4) and (5) are unnecessary. All one needs to relocate is to set up the partitions and to adjust the pointers.

In Williams and Dorner's *The First Book of OSI*, doing just that is described. In an example in that book, the authors show how a new workspace may be created by two (or three) POKEs and the NEW statement. One needs to adjust the pointer at $079, $007A (decimal 132,133), the beginning of BASIC workspace pointer, and to POKE in the initial null (Williams and Dorner do this with only two POKEs in the book, but normal use would probably entail three POKEs). So, if one needed to create a workspace at $1000, $0079 should be set to $01 (one byte past the initial null), $0071 should be set to $10, and a null should be inserted at $1000. After these POKEs, NEW should be typed to reset all other necessary pointers.

A normal LOAD should then load the program on tape into the new workspace, provided enough memory is left to hold it.

To return to a program LOADed into a new workspace, change the beginning of Basic workspace pointer with POKEs and type RUN.

What Else Was NEW Doing?

What I became interested in was the effect of the NEW upon the process. What other pointers were being reset? What else was NEW doing? Was it possible to relocate a BASIC workspace merely with POKEs?

I quickly found out that just leaving out the NEW only caused a machine crash. Taking a clue from Harvey Herman's article in the same issue of **COMPUTE!**, I tried modifying the single variable storage pointer, which is the same as the end of text pointer. In a normal cold start the end of text pointer ($007B, $007C) points to a memory address two bytes past whatever address $0079,7A points at. That is, after just cold-starting my C1P, $0079,7A points at $0301 (768) and $007B,7C points at $0303 (770). So, I tried setting my beginning of workspace to $1001 and my end of text pointer to $1003.

Well, it didn't crash — immediately. Obviously, there was something else the NEW statement did.

Checking around, I found that in a cold start, and in a NEW, addresses $0300 to $0302 are nulled out. So, I tried adding the POKEs to null $1001 and $1002 to the POKEs I had already.

Success! By POKEing the beginning of text pointer, the end of text pointer and nulling out the first three bytes of my new workspace, I could seemingly relocate my BASIC workspace without using the NEW command. But seven pokes seemed quite a bit more work than three pokes and a NEW statement, so I decided to see if any of the three nulls could be removed.

I found that the initial null could be left out to enter a program and to LIST it, but the RUN command generated a syntax error. If the null at $0302, or in the modified workspace $1002, was left out, the program would not LIST. But the null at $0301 or $1001 could be left out with no obvious problems.

$0301 and $0302 are pointers to the second line in the program in the 6502 format of least significant byte (LSB), most significant byte (MSB) for addresses. I knew that both being nulled indi-

OSI

AARDVARK NOW MEANS BUSINESS!

OSI

WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabibilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.
MAXI-PROS — $39.95

NEW-NEW—NEW TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -,=,*,/, ⟨⟩,⟨⟩,Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.
TINY COMPILER — $19.95 on tape or disk

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.

2) Complete listings of two word processors for BASIC IN ROM machines.

3) Moving the Directory off track 12.

4) Listings for 20 game programs for the OSI.

5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00
Vol. 2 (1981) 2 back issues and subscription for 4 additional issues - $9.00.

ACCOUNTS RECEIVABLE — This program will handle up to 420 open accounts. It will age accounts, print invoices (including payment reminders) and give account totals. It can add automatic interest charges and warnings on late accounts, and can automatically provide and calculate volume discounts.

24K and 0S65D required, dual disks recommended. Specify system.
Accounts Receivable. $99.95

* * * SPECIAL DEAL — NO LESS! * * *

A complete business package for OSI small systems — (C1, C2, C4 or C8). Includes MAXI-PROS, GENERAL LEDGER, INVENTORY, PAYROLL AND ACCOUNTS RECEIVABLE — ALL THE PROGRAMS THE SMALL BUSINESS MAN NEEDS. $299.95

P.S. We're so confident of the quality of these programs that the documentation contains the programmer's home phone number!

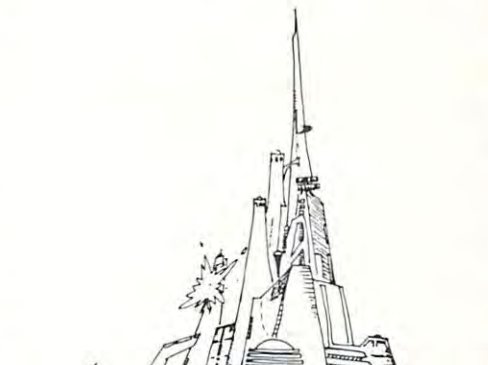
SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for — SUPERDISK II $29.95 (5 1/4") $34.95 (8").

AND FUN, TOO!



BOOKKEEPING THE EASY WAY — WITH BUSINESS I

Our business package 1 is a set of programs designed for the small businessman who does not have and does not need a full time accountant on his payroll.

This package is built around a **GENERAL LEDGER** program which records all transactions and which provides monthly, quarterly, annual, and year-to-date PROFIT AND LOSS statements. GENERAL LEDGER also provides for cash account balancing, provides a BALANCE SHEET and has modules for DEPRECIATION and LOAN ACCOUNT computation.
GENERAL LEDGER (and MODULES) $129.95.

PAYROLL is designed to interface with the GENERAL LEDGER. It will handle annual records on 30 employees with as many as 6 deductions per employee.
PAYROLL - $49.95.

INVENTORY is also designed to interface with the general ledger. This one will provide instant information on suppliers, initial cost and current value of your inventory. It also keeps track of the order points and date of last shipment.
INVENTORY - $59.95.

GAMES FOR ALL SYSTEMS

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95

NEW — NEW — NEW

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95.

NIGHT RIDER - You've seen similar games in the arcades. You see a winding twisting road ahead as you try to make time and stay on the road. NIGHT RIDER uses machine code to generate excellent high speed graphics - by the same author as MINOS.
NIGHT RIDER — $12.95 cassette only

THIEF - Another machine code goody for the C1P cassette only. You must use mobile cannon to protect the valuable jewels in the middle of the screen from increasingly nasty and trigger happy thiefs. Fast action and fun for one or two players. THIEF $13.95 on C1 cassette only!

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95
C1E/C2E similar to above but with extended machine code monitor. — $59.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

VISA

OSI

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

master charge

OSI

cated the end of the program, i.e. there are no other lines to be pointed to. But the need for only one null at $0302 convinced me to start experimenting with nulls in the MSB of pointers later on in the program.

I found that regardless of what value the LSB contained, a null in the MSB of any line's pointer had the effect of deleting that line and all lines following.

For example, let's say that a multi-line program has been typed into an OSI C1P at the normal workspace of $0300, and the memory addresses $0301 and $0302 contain $0E and $03, respectively. This means that the second line of the program begins at address $030E, and the pointer to the third line is at ($030E,$030F). If we change $030F to a null ($00), we will not be able to LIST or RUN past the first line of the program. Everything from the second line on will be deleted. Except for changing the pointers to FREe the now unused memory, we have effectively NEWed most of our program. But, if we restored the value contained in $030F before it was nulled, we would also find our program restored, as long as we hadn't added any new lines of code which would overwrite the pseudo-NEWed code.

This means that during a RUN or LIST, BASIC does not check both the LSB and the MSB for nulls to indicate the end of the program-just the MSB. Actually, this makes sense since no real program line would exist in Page Zero memory (addresses that start with $00).

Though relocation without a NEW statement is still six pokes (four for the pointers and two for inserting nulls), it seems to me that knowing exactly what each POKE is doing is a lot safer than using such a dangerous command as NEW. Further, this information gives us a number of practical benefits such as the capability of a reversible NEW.

So these are the results of my excursion into OSI relocation. I hope users of both OSI and other BASIC systems may find my experiences to be helpful.

References

Deal, Elizabeth, "Relocation of Basic Programs on the PET", **COMPUTE!**, June 1981

Herman, Harvey B., "Memory Partition of Basic Workspace", **COMPUTE!**, Jan.–Feb. 1980 (original printing)

Williams and Dorner, *The First Book of OSI*, Aardvark Technical Services, 1980