P6

the
AARDVARK JOURNAL
DECEMBER 1981 VOL. 2 NO. 5

'MERRY CHRISTMAS' AND HAPPY 'NEW YEAR' FROM RODGER, JUDY & CYNDI! WE WOULD LIKE TO THANK YOU ALL FOR YOUR CONTINUED SUPPORT THROUGHOUT '1981'. WE HOPE TO MAKE '1982' ANOTHER SUCCESSFUL YEAR FOR EVERYONE. HAPPY COMPUTING!

IN THIS ISSUE

FINALLY - AT LAST!!! AN INDEX!!!. A nice, concerned hard working reader, Jack Vaughn sent us a complete alphabetical index of the first two years of the journal. It is published in this issue. No more hopeless thumbing through 16 issues of dogeared paper to find that elusive article on the Quick Printer I.

We also have two excellent games to keep the kiddies busy during the Holidays. My own CONCENTRATION and Billy Smith's ALIEN RAIN. Both work on all OSI systems and Concentration is two player.

John Wilson, who did Caterpillar as our first Compiler written game, has contributed on article on some of the tricks that he learned about using the compiler. It turns out that it is a lot different that using BASIC. He and Bob Retelle have promised another article next month as they have continued to learn how to use the program.

Donald Sherry has contributed a neat C1 only program that plays tne Number square game from childhood. Your kids will be able to play it on a $1000 computer rather than the 15 cent plastic toy that we used. Progess!!

We also have a neat idea to add more ROM space to your system witho a new board, a good idea for Joysticks, and a cheap parallel printer interface from Australia. (Did you know that OSI was popular in Australia?).

Danial Wolfe sent in a normalized keyboard routine for those of you who are too cheeep for the C1E or a Maxipros.

David Pitts (Revered and Respected Author of Tiny Compiler) has contributed a normalized keyboard routine for the Word Processor that we have played with for so long.

David has also sent us a simple home budget program that works with Disk.

The home budget program and a neat plotting program from Randy Shedden have been published defensively. I dont like either kind of program and am tired of getting them in the mail. Figure if I publish a couple for free, people will stop asking me to review them.

P.S. Why do I hate them? Home budgetting is rarely done by anyone under any conditions. The few who want to do it can us a checkbook, 3x5 cards and a T.I. calculator easily. People often purchase the programs for home budget but rarely use them for more than a month. Despite that, if anyone submits a complete home budget program, we will probably publish it in the Journal - just to make certain that people stop sending them in.

Graphing programs have a different problem. The one we have in this journal is good - but all it does it to graph an equation. To me marketable, a graphing program must graph something in the real world - i.e. profits this year, profits as a percentage of sales, frequency distributions, acidity, something.

NEW AT AARDVARK

A bunch of new machine code games - including one for the C4P.

We have CATERPILLAR - the first of our Compiler written machine code games. A Caterpillar runs across and down the screen dodging mushrooms while you blast it. You also have to dodge hungry spiders while you play. A very fast, very smooth arcade piece for C1/4/8. $14.95

At last - an ASTEROIDS good enough for AARDVARK. For the C1P in all machine code. $12.95

We also have another great Edson game. Dave has combined a graphics game with an adventure attitude and come up with it's own. $14.95 - and a bargain! C1P ONLY.

How about a simple 30 character (58 character on Model II) video conversion for the C1P. No foil cuts, no jumpers, one part mounted externally on the case does the whole thing. Installation time is less than 5 minutes. Of course, the catch is that one part is a real video monitor. Now that they are so cheap, it makes more sense to buy a monitor than to butcher the C1P. Black & White monitors are available from AARDVARK for $99.95 - Green screen monitors are $119.00.

PUTTING THE TINY COMPILER ON A DIET
BY JOHN WILSON

John just submitted an excellent all Machine Code version of Centepede (Caterpillar) which he wrote with the Compiler. Here is part of what he learned while doing it.

While writing an arcade type game using the Tiny Compiler, I have collected some interesting notes which might be of some use. It doesn't take long to realize that the Compiler has some limitations, but with smart planning most of these can be avoided. One such limitation is memory consumption. This is a two fold problem. First, you have a Compiler which uses almost 5.7K of RAM along with your source program. During compilation, the object code will be poked into your remaining RAM. The length of the object code depends on the contents of your source program and how you use the Compiler's syntax.

Let's discuss the first problem. Using the compiler. I found by reading the documentation that it is very easy to delete portions of the program that would not be used. For example, if your program doesn't use the division portion, delete it. This could save you a couple of pages of memory. The locations for catch routine are given in the documentation supplied with the Compiler. Also put as much on a line as possible as the Compiler dimensions for 50 lines. This can be increased, but that would increase memory usage. So if you like to write in style, forget it! My first attempt at a game resulted in 9K object code. This was from a source program of only 3K. What a pig! It was then I decided to put the Compiler on a diet. And to this end I have come up with these facts.

Use the screen clear given in the back of the documentation. It uses at least 125 bytes less than a FOR-NEXT loop that fills the screen with blanks. It takes 15 bytes each time you assign a variable (e.g. A=10) but less than assigning it from a different variable (e.g. A=B). Also, the use of numbers anywhere possible saves a substantial amount of memory throughout the program. (Note: This is backwards from BASIC where it's cheaper to use a variable name- ed.) This was the first phase of the diet. It took a big 600 bytes off the original program.

After a quick study of how the Compiler works, it revealed that each time you added, divided or multiplied any numbers, it would poke in the code to perform the arithmetic. For that matter, anytime it did anything over again it duplicated code. In normal BASIC it would be foolish to make a subroutine such as A=A+2:RETURN or X=A*B;RETURN, but in using the Compiler this could save you lots and lots of memory!

This means that the structure in your source program must be carefully written. Combine any subroutines possible, in fact write your program to use common subroutines that can be called up to perform its function. Remember that it takes memory to assign variables, so measure the trade off to see where it would break even.

With this knowledge in hand, I shrunk the program down to less than 6K. Wow, the same program only 3K less to load in!!

Not bad for a little bit of thought! Next issue Bob Retelle and I will cover some major additions we have made to the Compiler that has drastically changed it's memory usage and flexibility.

OS-65D V3.3 by STEVEN GALE

'THANK GOD!!' The first words I said after assembling my 8" drive and memory-floppy controller board. Even before I had the system running, I bought the new 65D. When the disk was working I had already read and reread the 65D manuals. In just about three days I have learned some of the things OSI forgot to mention and I think that OSI did a good enough job for me to try to get you to run out and buy the new DOS.

The new keyboard decoder that makes the keyboard 'normal' has a few minor problems. First of all, <CTRL> X crashes the system. Those of you who use AARDVARK's 'INPUT w/o Scroll' should note that the location to peek is 9059, NOT 9815. Because 'RUB OUT' now is true back-space, the DOS won't let me output a string of CHR$(95) to underline. Other than that the new decoder is great.

The only other problem I have found is with data files and the use of the null input pokes and : and , terminator disable pokes. These must all be reset to the original value to do a DISK OPEN command. They forgot to tell you that the original value for location 9976 is 38.

THE GOOD THINGS:

The keyboard decoder makes the keyboard act like a typewriter. The <SHIFT LOCK> key now is a caps only key and <REPEAT> has to be held down for repeated characters. The <ESC> key can be used with the number keys to give these functions.

<ESC> 1	Clear screen, home cursor and set screen to wide character mode.
<ESC> 2	Same as <ESC> 1 but set to narrow character modes.
<ESC> 3	Home cursor
<ESC> 4	Clear screen from cursor to the bottom
<ESC> 5	Move cursor up one line.
<ESC> 6	Move cursor down one line.
<ESC> 7	Scroll down from the line under the cursor down.
<ESC> 8	Scroll down from top to the line of the cursor.
<ESC> 9	Turn color off.
<ESC> 0	Turn color on.

<RUBOUT> is now a true back-space and <SHIFT> P now removes the line you have started to type from the screen.

The video display now supports 'PRINT AT', scroll windows, many complex cursor functions, and printing in color. The cursor is a flashing half block and may be changed to any character flashing or non-flashing. The input prompt may also be changed to any character. It also has many general video formatting commands.

BASIC has also been 'souped up'. There is now a built in editor that is line oriented like the TRS-80's. Lower case may be used with, or instead of upper case at any time, including inputs. This means that an 'IF' will be true even if an upper case is compared with a lower case character. The 'TRAP' command has been added. The manual claims that acts like an 'ON ERROR GOTO' command. This is not completely true. If a 'TRAP 100' is encountered by a program, the system will automatically do a 'GOTO 100' whenever the program has to break. This means errors as well as <CTRL> C, and 'STOP' commands. It also means that the '? SN ERROR..' is printed on the screen. The 'PRINT USING' command has now been added and there now are printer commands like Form Feed and an MX-80 video dump command.

The best is still to come! The data file handling is much faster and easier to use. 'DISK FIND' now lets you scan a whole data file for a string very quickly (8K/sec. on 8"). Random files are improved by not reading a track if it already is in memory. The 'DISK PUT' command is now optional. An open command on an empty file does not give an 'ERR #6' in the V3.3.

The utilities that are supplied are great. The system powers up with a numbered menu BEXEC*. It will do a directory, Create, Rename, or Delete a file, create an all data disk, and set up buffers without calling other programs. The single or dual disk drive copier is not included in the BEXEC*, but can be run from the BEXEC* without much trouble. Other utilities include Trace, a modem driver, a program to add or remove buffers from your program, Resequencer, Repacker (like AARDVARK's Packer), a Disassembler, General String Oriented Sort, a Data file copier, and many more. Most of the passwords have been removed too.

Suprise! The manuals are great! The package came with 5 tutorial disks and one blank disk (2 of the disks were master 65D (V3.2 & V3.3) but you weren't told this at first to avoid confusion). There is also a 250 page manual (very good) and two reference manuals, BASIC and ASSEMBLER/EXTENDED MONITOR. Also included was a 'QUICK REFERENCE' card with summary of BASIC and DOS including page number and manual where more info may be found. Although I found a few omissions, I think the manuals were detailed AND readable. GOOD FOR YOU, OSI!!

The OS-65D V3.3 is well worth the 80 bucks I paid, and is much improved over the V3.2. It's time for a Disk drive all you cassette people, and you Disk people need to upgrade your DOS, soon!

COMMENTS ON, SPEEDING UP DISK DIRECT FILES AARDVARK JOURNAL VOL 2, #4 by

Donald VanSyckel, Vermont

I suppose at this point I should identify myself as the author of the original article. I am only shedding my anonymity because I feel a couple of corrections and explanations should be pointed out. I will list the corrected lines with an asterisk (*) below the changed area.

```
9020 ...FILE$+" ":...

9030 ...08,1":GOSUB9100

9050 ...08,2":GOSUB9100
```

In FUNCTION 2: WRITE RECORD there is a statement missing after the "a:" and before "LISTING #6". That statement is:

```
R=X:GOSUB9210 (X-user defined)
```

Replace line 9810 - 9840 with:

```
9810  DISK!"CA  B300="+RIGHT$
(STR$(100+TK),2+",1":RETURN
```

Likewise, replace lines 9860 - 9890 with:

```
9860 DISK!"SA "+RIGHT$
(STR$(100+TK),2)+",1=B300/D
```

It should be noted here that the two above changes not only save time and space but also allow the use of tracks 1 thru 76. The leading zero for tracks 1 thru 9 is picked up by adding the "100".

Part of a sentence is missing in the first paragraph on the top righthand side of page 10. The sentence should read:

"If it is not, then the WRITE FLAG is tested; if the buffer is dirty then the old track is written before the new track is called."

In FUNCTION 3: READ RECORD there is a statement missing after the "a:" and before "LISTING #7". That statement is:

```
R=X:GOSUB9310 (X - user defined)
```

FUNCTION 4 should be entitled "CLOSE FILE".

```
940 IF W=0 THEN RETURN
```

In the section where disadvantages are discussed it should be noted that the page of memory is no longer required to pass the CALL and SAVE commands to the DOS. Therefore line 10 may be changed in part to POKE133,178 for 48K systems.

The original article neglected to mention data file preparation. The user has several choices here as to what manner he wishes to proceed in. The most forward is to write a small program which writes to each disk record at least as many nulls or variables as the application program has variables in the READ and WRITE statements. These nulls must be zeroes (0) if the BASIC interpreter is not set up to accept null inputs. The second method is for the applications program to never read any given record before it has been written into. This is not as difficlut as it may seem since usually data is generated initially in a sequential order. The third method is a hybrid of the first two methods and is the one which I generally use. Write a separate disk initialization program which querries you for information such as NAME, record length, first data record number, and other "HEADER" type information and writes that into RECORD 0 (and 1,2.... as required). Then the applications program calls Record 0 with a special read routine which sets up all the parameters for that disk. This technique allows the same applications program to handle data disks with different size records. In this manner record length may be chosen on a case by case basis to provide for the maximum disk utilization.

NORMAL KEYBOARD INPUT ROUTINE

by Don VanSyckel, Middlebury, Vt.

One of the most difficult things to get used to when I was a new OSI user was the keyboard input routine. It worked with the shiftlock key down and seemed to only work for lower case letters and the space bar with the shiftlock key up. After some time I got a word processor and once again was painfully reminded that the keyboard was much less than desirable. When typing normal English on the word processor each upper case character had to be shiftlocked and then unlocked and each punctuation character had to be shiftlocked and unlocked also and sometimes shifted to boot. Of course, I just "love to type" and these features just endeared the task even more.

Now that I have gotten some time and have some incentive, my wife is using my system, I have written a normal keyboard input routine which will generate each of the 128 ASCII codes and have burned an EPROM with it in and installed it in the system. I started by examining the existing routine which lives at $FD00 - $FDFF. The fact that the OSI routine consumes all 256 bytes of the page is suspicious in itself and leads me to believe that possibly they are not happy with it but could not find a way to shorten the routine and do all the functions. The routine which is presented here works for all applications I have tried to date and has an extra feature of being able to "test" the keyboard for a character without waiting for one. However, I must point out here that my routine is also exactly 256 bytes. The routine was 6 bytes too large but by shuffling program segments I picked up one byte by changing the one jump in the program to a branch and found the other 5 bytes by combining a 5 byte table into the main table (more about this later). I considered searching for one more byte so that I could claim my routine did not use a whole page of memory but decided that the main routine is smaller than one page since a new function has also been implemented.

The routine operates in the following manner:

1) The keyboard is scanned for a contact closure.
2) The contact closure is converted to a character.
3) The shift function is applied.
4) The REPEAT function is applied.
5) The CONTROL function is applied.
6) The debounce function is applied.

The ASCII characters generated are the normal or expected ones (not OSI normal) for all keys of a CRT type terminal when the left or right shift or the shiftlock is used. That covers the ASCII range of $20 - $3F - $41 - $5A, $61 - $7A, and $7F. The REPEAT key has been encoded to add $10 when depressed to the generated ASCII code. With this in mind the 11 ASCII characters between $20 and $7F which were not listed above as being directly keyable are accessible by depressing the REPEAT key and entering the character which appears in the ASCII table one column left of the desired character. The most pertinent three are:

REPEAT "O" = "-" or DELETE CHARACTER
REPEAT "N" = "^"

As expected the ASCII control characters are generated by depressing the CONTROL (CTRL) key and entering the characters in the $4X column for the $0X column and entering the characters in the $5X column for the $1X column in the ASCII table. There are 5 keys for which the shift function is skipped. These are LINE FEED ($0A), CARRIAGE RETURN ($0D), SPACE ($20), ZERO ($30), and RUBOUT or DEL ($7F). They are listed in the EXCeption table. Due to structure of the ASCII lookup table (TABLE) in this routine there were 3 unused locations following LINE FEED and CARRIAGE RETURN which just happened to be next to each other. (This is where the 5 byte table is tucked inside the large table.)

On occasion I have needed to poll the keyboard to determine if a character was there and what it was, but did not want to relinquish program control to just sit and wait for a character. The routine KEYBOARD READ (KYBDRD) which begins at $FD11 fulfills this need. As can be seen by examining the first three lines of code of the standard routine, KEYBOARD WAIT (KYBDWT), the routine

KYBDRD actually does all the work. KYBDRD will return either the debounced ASCII value of the key being depressed or a zero if no key is being depressed. This allows the computer to continue performing some task and occasionally sample or poll the keyboard for an ASCII character, not just a switch closure as is done with PEEK's and POKE's.

Of course, if you aren't going to burn an EPROM, the subroutine can be assembled anywhere in memory where it will fit. Either the operating system or a specific BASIC program can load it. In any case, if the routine is someplace other than $FD00 you will need to change the 2 bytes of the DOS JSR instruction which points to it. These are located at $2532 (LO) and $2533 (HI) for OS65D-V3.2.

If anyone would like the routine in EPROM just send a blank 2716 type EPROM and $10 or no EPROM and $20 and I'll send back the EPROM with a modified SYNMON V1.0 in it. The address is:

Donald VanSyckel
Halladay Rd.
Middlebury, VT 05753

I'm not aware of other version monitors, but if you have one, something might be worked out if you can spare the monitor chip for a couple of weeks.

SEE PAGE 18,19 & 20

DANIEL WOLF

I always enjoy reading the excellent machine code programming suggestions from other Superboard users in your pages. While constructing some dumb terminal emulator programs for the Superboard I needed a non-waiting machine code keyboard read routine and a non-waiting ACIA read routine. The routines in OSI's monitor wait forever for a keypress or ACIA ready, respectively. What I needed were routines that would give one result if they read something and another result if no read was done. For a dumb terminal, that's all that's required but the display routine.

There have been published remedies for these "endless" loop problems, but the remedies use BASIC. I desired an all machine code solution to use with a variety of other applications as well. I'm sure the following is not the only machine code solution to this problem, but it is easy and cheap!

OSI's keyboard routines lives at $FD00 in the monitor. Inspection of the code there revealed a single byte that causes the endless loop, $FD74=$8F. The ACIA routine is at $FE80 (I leave the detection of the single offending byte in this routine as an excercise to the reader, or see how I handled it below). A little experimentation revealed that I could move these routines into RAM, make my simple modifications, and have useful working routines I needed.

Specifically:

I copied page $FD to page $1F and changed $1F74 to a $4F. This gave me a keyboard read routine that has two possible outcomes. If a key is depressed, the routine goes to $1FC1. If no key is depressed, it goes to $1FC4. Into these two locations I could place the appropriate code to handle the two possibilities, such as JSR's or JMP's.

To demonstrate the principles of moving and modifying the monitor routines, the following all-machine code dumb terminal program is both educational and useful as is. The whole thing is less than one page of memory. Here's how to get it running:

1. Use whatever method you like to move page $FD to page $1F (in BASIC IMMEDIATE MODE : FOR A=0TO192:B=PEEK(64768+A) : POKE7936+A,B:NEXT)
2. Use the monitor itself to enter the following few bytes of machine code into the addresses shown:

```
$1F74    $4F                 Change the looping
byte to a $4F
 1FC1        4C  D8 1F        Key depressed,
jump to 1FD8 for output
 1FC4        AD 00 F0      No key, read the
ACIA status
 1FC7    4A                  Rotate left
 1FC8          90                Branch if carry
clear (no character to read) to 1FD5
 1FC9    0B
 1FCA        AD 01  F0         Read character
from ACIA
 1FCD    EA EA EA       Timing no-ops
 1FD0    2A 7F              Strip bit 7
 1FD2      20 2D BF          Display the
resulting ASCII character
 1FD5    4C 04 1F       Jump back to begin
the keyboard read again
 1FD8         20 B1  FC            Output  the
depressed key to the ACIA
 1FDB        A9 00           *Load accumulator
with a zero
 1FDD    EA EA EA       * see below
 1FE0    4C 04 1F       Jump back to begin
the keyboard read again
```

Now just set the monitor address at $1F00 and type a G.

*As written, it functions as full-duplex (doesn't display keyboard characters, only incoming ACIA characters) To simulate half-duplex (that is, display keyboard and ACIA characters) place A9 00 at 1FDE and 20 2D BF at 1FDB.

The disadvantage of this routine is it only does the OSI style keyboard decode. Its advantages are worth it; all machine code, general purpose, easy, almost no programming skill required by using pre-written monitor routines (after all, why re-invent all wheels?). Most important, it is fast enough to function as a useful dumb terminal at 300 baud. I've used it with MICRONET and TELEMAIL with no problems. It's dumb, it's an endless loop itself, but it works! Using similar techniques, I have borrowed a number of monitor routines for my own purposes, including the BF2D display routine modified for any length up to 24 (decimal).

PARALLEL PRINTER INTERFACE

by Jeff Rae & Geoff Cohen from the Omega Newsletter, Australia

Jeff Rae very kindly sent me an idea for a simple, very cheap parallel printer interface for all OSI computers. (in fact this should work on any Computer that uses an ACIA on its serial output port). So if you dont want to pay around $150 for a serial interface for a Microline 80, Epson MX-80 or similar, read on, as the total cost of this project should be around $15.00.

Jeff's early version would not work on all types of printers, so a few changes were made, and the following circuit emerged. The principle is very simple, as are all greak inventions, and best of all, no software changes are required. The computer still thinks that the printer is a serial one, on port number one.

The data is latched by the 74LS374, whenever the processor is writing to the ACIA data register, by the address decoder 74LS02 and 74LS10. This is when serial data would normally be sent to a serial printer. After a few microseconds delay (to meet the printer specifications) by the 4093, a signal is sent to the parallel printer that the valid data is available (not DATA STROBE).

The printer then sends its BUSY signal high, and this is sent to the normal serial handshaking input, CTS (clear to send) on the ACIA. Using the standard serial software, the Computer will not send any more data until the printer is ready, and then the cycle repeats until all data has been printed.

There is a printed circuit board available ($4-50 undrilled or $5.50 drilled, plus $2.00 p&p) (write Geoff Cohen, 72 Spofforth St., Holt, ACT, 2615, Australia - phone (062) 547608(H) 492688(W)), which is very simple to install, although the circuit could be built up on Veroboard or similar, if there are any masochists amongst our readers.

There are two ways of installing the PCB. If there is sufficient height above the Computer, a Wire Wrap socket can be soldered in the space marked ACIA SOCKET (on the component layout). When all components are installed on the printer board, and the printer ribbon cable connected, the ACIA is removed from the Computer, and placed in the socket market ACIA 6850 (on the printer PCB). Now the WireWrap sockets long pins are plugged into the ACIA socket on the Computer, and thats it. Turn on the printer and away you go.

The other method of assembly is very similar, but uses a 24 way ribbon cable, with a 24 pin DIP plug on each end instead of the Wire Wrap socket. This cable is then used to connect from the ACIA socket on the Computer to the other socket on the printer PCB. The Wire Wrap method is simpler (and slightly cheaper), and there is no need to screw the printer PCB to the Computer, but otherwise they are indentical.

PRINTER POKES

Although the above parallel printer will work with no changes to the system software, two Pokes will enable much faster printing. On the printer I use at work (a Microline 80), and most similar ones, the printer fills its print buffer, until a carriage return is detected.

Then (and only then) does the line get printed. While the printer is doing its thing to the paper, the Computer is not allowed (the BUSY - CTS handshaking signals) to fill up the printers buffer. The result of this is that the time spent filling up the print buffer is wasted, as there is no printing in this time. What we really want to do is speed up this process.

Luckily this is very simple, merely by reprogramming the ACIA. The ACIA has two registers, the DATA register to send data in or out, and the STATUS register, which is the one we want to change. OSI set up the ACIA for an internal divide by 16, and if we change this to divide by one, the print buffer will fillup 16 times faster, at an effective baud rate of 19200 on a C4-MF.

The programming necessary is to first POKE the ACIA STATUS register with a 3, to reset it, and then POKE with a 16 to set divide by one mode. The simplest way for Cassette systems is as follows -

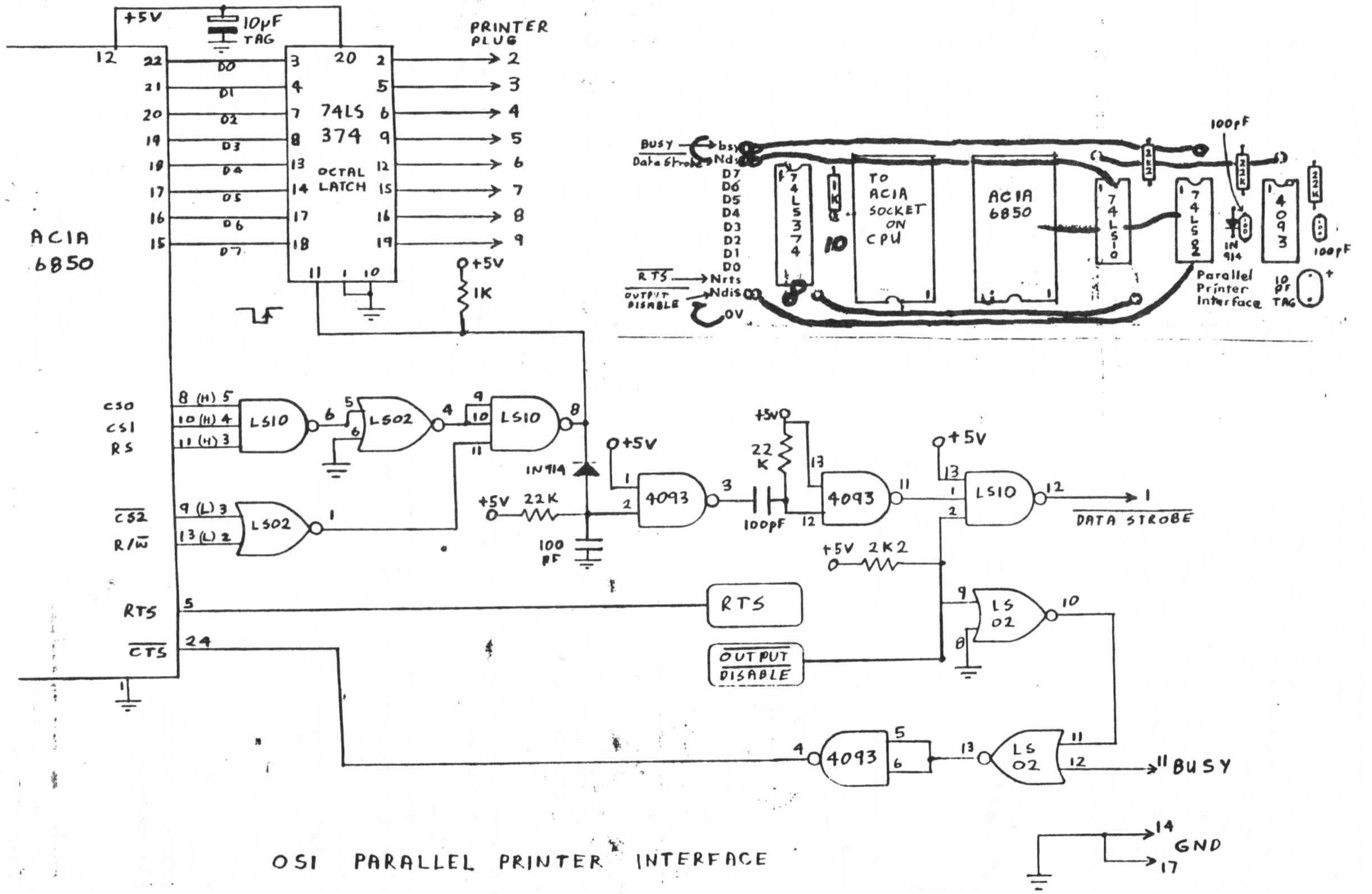
```
  C1 - POKE61440,3:POKE61440,16
  C4 - POKE64512,3:POKE64512,16
```

This can also be done on disk systems, probably in BEXEC*, but I prefer to change the ACIA initialization routine on track zero. If track zero is called into location $4200, address $424B changed from 11 to 10 and then saved back on track zero, the fast ACIA will automatically be set on bootup.

Another gem from Jeff Rae is a fix for the CONTROL-P routine (Disk system only). This should allow the printer to be toggled and off by typing Control P. Unfortunately the DOS is set for a parallel printer, but this can be fixed by a -

```
POKE 9618,1
```

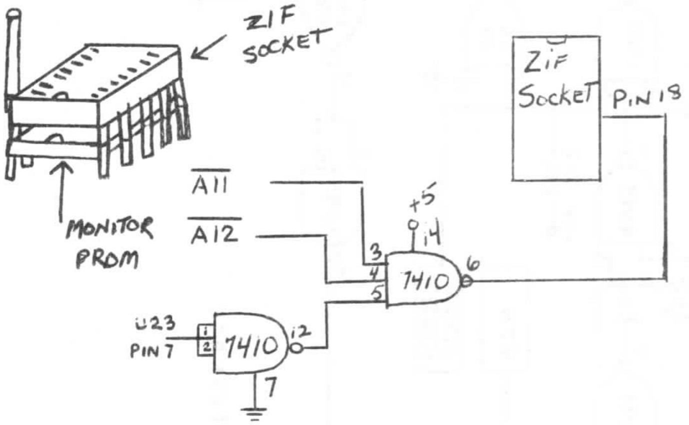
This could also be changed in track zero, the hex address is $2592, or $4592 when track zero is loaded at $4200.



OSI PARALLEL PRINTER INTERFACE

CHARLES W. SCHALL, WEST HAVEN, CONNECTICUTT

Because of two failures in BASIC in ROM chips for my C1P, I recently purchased AARDVARK's Eprom Programmer board in order to creat some spare chips. After burning the chips, I decided to expand the ROM capability of my machine. First I added some active ROM space for 2716's at the addresses $E000-$E7FF. A crude but effective method was to solder a textool 24 pin zero insertion force socket directly to a copy of the monitor prom (C1E). All pins except pin 18 (chip select) were soldered. Pin 18 was connected as shown in the following diagram. The 7410 was placed into a prototype socket.



The next step was to put something into the Eprom space. The extended monitor was an ideal choice since I use it frequently and it fits in 2K.

To get it from tape into ROM, I used the following procedure:

1. Load EXMON
2. Use EXMON to move the code from $0800-$0FFF to $1000-$1FFF. (This allows BASIC programs to be loaded underneath).
3. Coldstart, answering 4000 to memory size, enter and run the following:

```
10 FOR X=4096TO6144
20 A=PEEK(X)
30 IFA=76THENGOSUB100
40 IFA=32THENGOSUB100
50 IFA=173THENGOSUB100
60 IFA=189THENGOSUB100
70 NEXTX
80 END
100 D=X+2
110 E=PEEK(D)
120 IFE>15THENRETURN
130 IFE<8THENRETURN
140 C=E+216
150 POKED,C
160 RETURN
```

This program corrects all the appropriate jumps, jumpsubs, and LDA's.

4. Enter the monitor and correct the jump table as explained in the excellent article by the UK users group in the December 1980 Aardvark Journal. Note that in this case $D8 must be added to each Hi byte from $1160 to $1199. Example: Change $0B to $E3.
5. Coldstart, answering 4000 to memory size, and load the eprom programmer program. Run the program to put code from $1000 to $17FF into the eprom starting at eprom location 0.
6. When programming is complete, move the prom from the programmer to your new prom space. Enter the monitor and go from $E000. EXMON should be there.

RANDY SHEDDEN, VANCOUVER, WASHINGTON

This program is basically a function plotter for the C1P. I would have made it convertable to the C4 and C8 except for the fact that I own neither and the program would seem redundant with their expanded video resolution. The problem with a function plotter for the C1P is that the 32 X 32 screen makes for an inaccurate representation of the function. (especially since the graphics are cut down to 25 X 25 on most monitors) Now if you want to quadruple your grahping capability and you don't want to mess around with the ROM or any hardware doctoring, (the prospect makes my skin crawl) you can make use of the graphics characters already residing in the character generator. The following is a program to graph a function on a 64 X 64 screen. Cut down to 50 X 50 on most monitors.

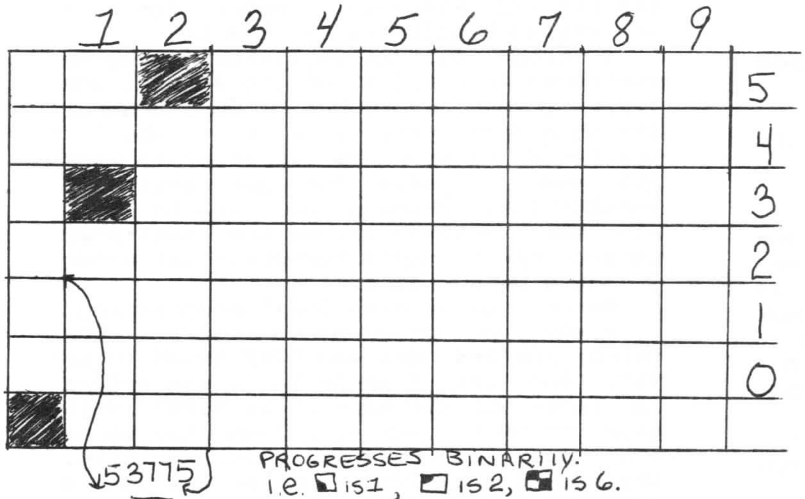
```
10 FORX=0TO24:PRINT:NEXT CLEAR SCREEN
20  O=53775:H=24:L=24 SET
ORIGIN,HEIGHT,LENGTH
30  DIMB(15) DIMENSION  BINARY GRAPHICS
CHARACTER
40 FORX=0TO15:READB(X):NEXT FILL BGC
50 DEFFNA(X)=INT(SIN(X*3.14159/L)*H)
DEFINE FUNCTION
60 FORX=O-12TOO+12:POKEX,128:NEXT DRAW X
AXIS
70 FORX=O-384TOO+384STEP32:POKEX,136:
NEXT DRAW Y AXIS
80 POKEO,209 DRAW ORIGIN
90 FORX=-24TO24STEP2 SET PERIMETERS OF X
COORDINATES
100 A=INT(X/2) HORIZONTAL DISPLACEMENT
110 B=FNA(X)/2 HALF OF THE Y VALUE
120 C-INT(B) VERTICAL DISPLACEMENT
130 POKEO+A-C*32,B(2*(B-C)+1) EXPLAINED
BELOW
140 D=FNA(X+1)/2  HALF OF THE  NEXT  Y
VALUE
150 E=INT(D) NEXT VERTICAL DISPLACEMENT
160 F=PEEK(O+A-E*32) IS THERE  SOMETHING
ALREADY THERE?
170 Y=0
180 IFF=B(2)THENY=2 SET Y AS CONTENTS OF
SCREEN LOCATION
190 IFF=B(1)THENY=1
200 POKEO+A-E*32,B(4*(2*(D-E)+1)+Y)
EXPLAINED BELOW
210 NEXT NEXT X COORDINATE
220  DATA32,167,168,156,165,154,
169,178,166,170,155,  175,157,176,
177,166
```

And now the explanation that was promised. In line 50 the function is a sine function, however, this is where you can substitute your own function. For instance, 50 DEFFNA(X)= INT(LOG(X/L)*H) or INT((1/X)/L)*H). In the first example you will get a FC ERROR when X is less than or equal to 0, and in the second when X=0. To avoid this, set the perimeters in line 90 for positive or none-zero integers. In line 130 the equation O+A-C*32 and O+A-E*32 in line 200 is the screen location. The equation 2*(B-C)+1 will yield a 1 if the Y coordinate is even and a 2 if it is odd. This insures either B(1) or B(2) in line 130. In line 200, 2*(D-E)+1 serves the same purpose. 4*(2*(D-E)+1) will yield a 4 or 8. When you add Y to this you insure that you will not wipe out the last dot plotted. This will all make more sense when you look at the accompaning diagrams. You can change the frequency by changing L, and the height by changing H. The origin by O.

This program can also be used to enhance the displays of video games. It would probably be more useful done in machine language if you were going to use it in a fast video game.

(The above program was published as a defense against seeing more plotting programs. We get about one a week in the mail. I I've keep explaining to authors that a plotting program must plot something - i.e. stocks or profit- to be marketable.)

THE BGC IS DIVIDED INTO 4 SECTIONS AND



PROGRESSES BINARILY. i.e. ▖ is 1, ▘ is 2, ▚ is 6.

2 8 / 1 4 ~PIXEL B(1)= ASCII(167) B(2)=168 B(8)=189

BILL REED, WHITTIER, CALIFORNIA

I have been in DP since the days when we walked around inside the calculator and it cost $75,000.00 a month to rent. Now of course, you can buy a better machine today for $20.00 and carry it around in your shirt pocket.

My personal system is an OHIO C3-OEM with double dual floppy's and a TI 810 RO printer all of which I am very happy with. I use both 65D and 65U and have fooled around with CP/M, but not seriously - as OHIO's DMS is my favorite work horse - now for the question. Has anybody out there interfaced to an S-100 buss? If so, I would be interested in hearing from them.

DEAR MR. REED:

There have been a couple of S-100 Buss adapters for OSI on the market. D & N Micro products in Fort Wayne fooled around with one for a while and there were a couple advertised in old issues of MICRO at one time. We even considered doing one here at AARDVARK.

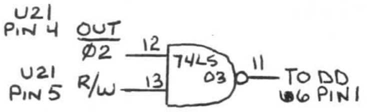
Unfortunately, the experience that we all had was that there was no S-100 buss. Every manufacturer seemed to find the buss somewhat different, we were never able to get two boards from two different manufacturers to run at the same time. They all seemed to define some lines differently than others did. In any case, virtually everything you could want for a computer is available for the OSI either from OSI themselves or small companies like ourselves or D & N Engineering. I therefore, see very little chance that anyone will bother to make an S-100 buss adaptor in the future.

AL MCCANN, GLENOLDEN, PENNSYLVANIA

Recently I added the AY3-8910 Sound Generator chip in a circuit similar to the one in the August 81 Journal. This was just the article I was looking for. The only difference between my layout and the published one is that I had some extra 16 address chip selects on a 6522 I/O board floating around and used one of those.

When I checked this circuit out, I discovered that it would not work correctly. Much hair pulling, chip pulling and screaming later, I discovered what the problem was.

I have Mittendorf's High Res. Video board which will not operate with the 8T28 Data Bus Buffers. It has it's own anyway, so as per instructions supplied with said High Res board, the 8T28's were by passed. Now with all the other goodies I've added on, the problem with the sound chip turns out to be buss loading. Putting the 8T28 chips in made it work. But video didn't. So I moved the data buss of the High Res board to the input side of the 8T28's. Now it works. I still had conflicts generating D D signal because the 6522 I/O board drives it with regular TTL, not open collector. So I disconnected all DD wires to all board and worked this circuit out



Now the data buss is buffered, and always enabled.

Now for something completely different. Here is a way to add new commands to BASIC. Mittendorf does this with the software for their High Res. board. Credit goes to them for this. POKE542,48:POKE543,2 This points the "LOAD" vector at 0230 HEX where a M.L. program will reside. At 0230 HEX put something like this:

```
     0230 20 C2 00          JSR TO CHARGOT
     0233 C9 XX             CMP WITH XX
```

In our BASIC program a new command would look like this:
LOAD,TOKEN OR CHARACTER, VARIABLE IF NEEDED.
The first 20 C2 00 is a jump to subroutine called CHARGOT. It gets our token or character following the Load TOKEN, ignoring spaces, and puts it in the accumulator. Then we can run it through some comparisons to jump to different subroutines we want to do.
Before returning to BASIC we must find the end of line on the ":" that would be an exit routine and would be something like this:

```
     20 BC 00      JSR TO CHAR GET
     C9 00         CMP TO END OF LINE
     DO 01         BRANCH IF NOT
     60            RETURN TO BASIC
                   IF IT IS
     C9 3A         CMP TO ":"
     DO F4         BRANCH BACK TO
                   BEGINNING OF
                   EXIT ROUTINE IF NOT
     60            RETURN IF IT IS
```

To get data we must also call the CHAR GET routine and get the data:

```
     20 BC 00      JSR TO CHAR GET
20  C1  AA OR 20 AE B3      AAC1 GET F.P.
VALUE OR B3AE GET 8 BIT VALUE
```

The AAC1 routine puts the result in the F.P. accumulator and the B3AE routine puts the result in 00 AE, AF HEX. If you use the AAC1 routine call AE05 invar routine and that converts the FPA to 16 bit binary and puts the result in OOAE,AF. To return a value to a BASIC variable:

```
LO BYTE ONLY  -  20 C1 AF        JSR TO OUT
VAR
HI  BYTE IN A -  20 7H B7        JSR TO PUT
FPA  IN  VARIABLE  POINTED AT BY 0071,72
LAST USED VARIABLE RTS RETURN TO BASIC
```

The real First Book Of OSI is very helpful in figuring these routines out. Using these ideas I modified Mittendorfs software to add a new command to determine whether or not a point was lit.

ROBERT C. ALLISON, PUNTA GORDA, FLORIDA

I have been a ham for 53 years, and before retiring, was a chem. engineer exposed to all kinds of instrumentation. I had 12 scientific programmers on a staff, and was deeply involved in setting their goals, but never in programming.

Last Nov. I ordered a C1 locally. OSI had not delivered by Feb., so I got a C4 with the assurance that they had programs for Morse and RTTY. However, it developed that they would not work on a C4. (Since then, I have changed the Morse program to work on the C4, but it is not a good running program in that one has to wait for a character to finish before pressing the key for the next character.) This makes it awkward to use except at high sending speeds. A Morse keyboard I built works much better.

I ordered the Morse-RTTY HAM 1 program from the Computer Works in Feb. and recieved it two months later. Then, although it worked, it would send RTTY only at 132 words per minute, over twice as fast as the correct speed for most Ham RTTY. They assure me that they have fixed the program, and will supply a corrected program in a few weeks.

Also, the Computer Works were supposed to have an improved Morse-RTTY program ready in February. It still is not ready. They call this the HAM 1B program.

So you can see that I am disenchanted with computers as far as HAM radio application is concerned, although I appreciate the difficulty involved and the limited financial reward for selling such software for the C4.

I typed out the BATTLEFLEET program published in Vol.2 #3 of the Journal and could not get it to run. So I ordered the tape from you, it runs. The published version has an error in Line 211 where it says for tape systems, Y=PEEK(513), whereas the tape in Line 210 has Y=PEEK(531), so the published version has a simple typographical error I should have caught.

Being new at even BASIC programming, I study your programs to learn. One thing puzzles me; you POKE 56900 to go back and forth from 34 to 72 characters per line, but OSI says it is 56832. Both seem to work, but sometimes one works and sometimes the other. Incidentally, the Manual from OSI I got for the C4P is 70% instructions for the disk system, and in a lot of it, they don't say which system it is for. So it takes about six months to get things straight.

I am amazed at how little standardization there is between different manufacturers of home computers, even when they have the same CPU. The industry would prosper more if they at least used identical BASIC statements throughout.

As an old electronic buff very experienced with solid state and chips, etc., I am amazed at how much does work in the C4, and probably will do better with it as I learn more.

ROBERT CAMNER, THE MARET SCHOOL, WASHINGTON D.C.

Earlier this year I ordered from you the fact sheets for the C1P "GT Conversion" and "Reverse Video". A busier than expected schedule prevented me from attempting the modifications until recently.

I have the C1P Series II (600 board Rev. D) and discovered that the board is quite different from the descriptions you provided on your two fact sheets. The chips where the jumpers are attached are of course there, but the surrounding areas and the locations of the plated-thru holes are different. Upon completing the mods, this is what I found:

1) The GT mod works as described, as long as the machine is in the "large" 24 X 24 character mode. A poke to set the machine to the 12 X 48 character mode locks up the machine. It appears (for reasons I don't understand) that your GT option won't work at 2 MgHz.

2) The "Reverse Video" mod is less successful; the background switches to all white, but no characters are generated. Perhaps the "unused" inverter you describe in the fact sheet is no longer unused on the 600 Rev. D.

3) The "disable" the break key mod you describe on page 11 of an early Aardvark Journal works, of course, but again the 600 board Rev. D has the traces for the key in a different place and thus your directions had to be modified.

(NOTE: When odering Data sheets or Roms, you should specify which model (C1P or Superboard) you have. Some stuff does not work in both!!)

HARLEY CAMPBELL, OKLAHOMA CITY, OKLAHOMA

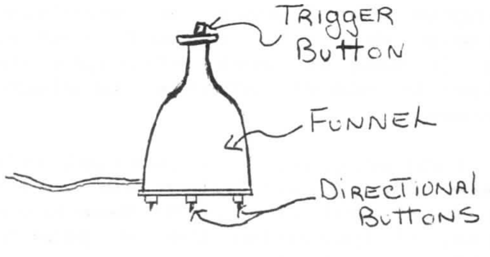
I discovered by accident that I have a non-scrolling 25th line on my Superboard II (1978 Rev. B). It's screen address is immediately below the bottom line shown on the 24 X 24 video map (address 54149 to 54173). I have used your"print at" routine to insert text in this location as follows:

```
10  A$="COPYRIGHT 1981":REM ENTER  INFO
HERE
20  N=54154:REM A POINT ON NON-SCROLLING
LINE
30  FOR I=1  TO  LEN(A$):POKEN+1,
ASC(MID$(A$,I,1))
40 NEXTI
```

The text can be removed by POKEing the line with blanks.

LARRY CORNINE, MACON, MISSOURI

I have an improvement on the "homemade" joysticks. I bought two small funnels instead of bowls. I simply put the trigger-button in the end of the funnel with the directional buttons in the bottom. This can be held as a conventional joystick on your desk top.



CARL M. KING, SARASOTA, FLORIDA

I have been looking forward to an opportunity to use the INPUT WITHOUT SCROLLS routine that I learned from you. However, one drawback with it, that I could foresee, was that it didn't have a back-space. It wouldn't be much good if you could only type going forward, and not be able to go back to correct errors.

While experimenting with your little routine, I accidently discovered an existing back-space combination. It is the key combination <CTRL>, <H>.

The following three line program that inputs without scrolls and which back-spaces as I have indicated.

```
30 DISK!"GO 252B":P=PEEK(9815)
50 PRINTCHR$(P)CHR$(32)CHR$(8);
80 P$(I)=P$(I)+CHR$(P):GOTO30
```

DAVID WHIPP, SALT LAKE CITY, UTAH

I had presumed this was common knowledge, but the slow and awkward polling techniques I've seen show it isn't. By using POKE530,1:POKE57088,64 the entire keyboard is activated! Everything, that is, except "RUBOUT". And there will be duplicates, but there will be immediately available 8 different PEEKs of 57100, and using key combinations like CTRL and numbers 1,3,4,5,6,7 there can be more unused PEEKs. Of course, by adding the option of lifting the SHIFT LOCK, the possibilities are doubled. If you need more, try three keys & at once!

RON THOMPSON, PANACEA, FLORIDA

Here is a little item you might find interesting. If you solder a 100 UF capacitor across the terminals on the underside of the BREAK key (negative lead to the ground side, of course) it will give you reset-on-power-up. No fancy delay on reset like on the Series II, but who needs it.

DAVID E.PITTS, HOUSTON, TEXAS

In the process of doing the homework you assigned in Vol. #2 of the Journal (A Simple Processor), I have written a routine for upper and lower case for the 4PMF keyboard with a rubout of the previous characters.

I've also included a simple home budget program that I wrote as practice for simple disk I/O. My next task with it is to make it more efficient (storage wise) by use of strings in place of arrays.

1) load program, 2) indirect file, 3) make one 8 page buffer for unit 6. 4) Xc (i.e. control x), 5) Save program on disk, 6) thereafter the 8 page nutfer will load with the program.

```
10 REM UPPER AND LOWER CASE KEYBOARD
12 REM Y=KEY,X=SHIFT
13 REM SHIFT RUB OUT DELETES  PREVIOUS
CHARACTER
15 REM DAVID PITTS, HOUSTON TEXAS
90 A$="":POKE55104,62
91  DISK!"GO  252B":X=PEEK(57088)
:Y=PEEK(9815)
92 IF(X<2ORX=33)ANDY>64THENY=Y+32
93 IFX=1THEN109
94 IFY>80THENY=Y-16
96 IFY=64THENY=80:GOTO109
98 IFX=0ANDY=108THENY=Y-32:GOTO109
99  IF(X=33ANDY=95)OR(X=0ANDY=92)
THENY=Y+16
100 IFY=13THENGOTO110
102 IFY<>95GOTO109
103  POKE55104+LEN(A$),32:A$=
LEFT$(A$,LEN$(A$)-1):GOTO91
109 A$=A$+CHR$(Y):POKE55401+LEN(A$),Y
:GOTO91
110 PRINT:A$(L)=A$:L=L+1:GOTO90
```

```
5  REM DAVID PITTS, HOME BUDGET PROGRAM
10 DIMS(12,20),C$(20)
15 FORI=1TO20:READC$(I):NEXT:RESTORE
20  INPUT"YEAR E.G.
Y80";F$:INPUT"MONTH";M
25  INPUT"WANT DISK
DATA";X$:IFLEFT$(X$,1)<>"Y"THEN50
30 DISK OPEN,6,F$
40 FORI=1TO12:FORJ=1TO20:INPUT
#6,S(I,J):NEXT:NEXT
50 FORI=1TO32:PRINT:NEXT
60  FORI=1TO20:PRINTTAB(18);CHR$(64+1);
TAB(25);C$(I);TAB(35);S(M,1):NEXT
61 J=0:FORI=1TO20:J=J+S(M,I):NEXT:PRINT
62 PRINTTAB(25);"TOTAL";TAB(35);J
67  PRINT:PRINT"SET CATEGORY=* TO PUT ON
DISK, =? TO CHANGE MONTH"
70 PRINT:PRINT:INPUT"CATEGORY";X$:CAT=
ASC(X$)-64:IFX$="*"THEN2000
80 IFX$="?"THEN20
90 INPUT"DOLLARS";DOL:S(M,CAT)=
S(M,CAT)+DOL:GOTO50
2000  REM  SAVE S(,) ON DISK
***************
2010 DISK
OPEN,6,F$:FORI=1TO12:FORJ=1TO20:PRINT
*6,S(I,J):NEXT:NEXT
2020 DISK CLOSE,6:GOTO50:END
3001 DATAFOOD,CLOTHING,AUTO,ENTERTAIN,
SAVINGS,INSURANCE,NATGAS
3002  DATA
ELECTRIC,PHONE,HOME,CHARITY,COMPUTER,
HEALTH,SCHOOL
3003  DATAEMPLOY,GIFT,MISC,
OTHER,OTHER,OTHER
```

CORRECTION BY CARL M. KING, SARASOTA, FLORIDA

I was scanning through looking for an errata notice regarding the item in the June issue: PRINTING GRAPHICS WITH OS65D on page 17. I found that one interesting, but it didn't work. I believe there is a typo in it. I made a guess and found that it worked if the line reads: 110 POKE9633,234: POKE9634,234

CHARLES HEPNER, STERLING HTS, MICHIGAN

I noticed my BASIC MEMORY DUMP program in the October issue of the Journal and noticed I should have included the following column identifications:

1. BASIC LINE NUMBER IN DECIMAL
2. MEMORY LOCATION OF CURRENT LINE IN HEX.
3. MEMORY LOCATION OF NEXT BASIC LINE IN HEX. (LO BYTE/HI BYTE)
4. CURRENT BASIC LINE NUMBER IN HEX.
5. CONTENTS OF CURRENT BASIC LINE IN STANDARD HEX. FORMAT

This list is a by-product of a dis-assembler program I wrote. In that program I wanted to avoid string arrays so I used an op-code data file.

CLASSIFIED ADS

FOR SALE: OSI C2/4P MF, one 5-1/4" disk drive, 24K, cassette, sound, DAC, Joysticks, RS232 port and C2E monitor. Excellent condition. $1,000.00 (517) 423-4871 eve.

PRETZELAND SOFTWARE offers quality games on cassettes for all OSI systems, with full support of sound on all C1P's and Superboards. Send $1.00 for Photo-illustrated catalog & get a $1.50 credit good on your first order. Bob Retelle, 2005 C Whittaker Rd, Ypsilanti, MI 48197.

'OPINION' on VIDEO GAMES #3 by Orion Software

This is a three-programs-in-one machine code game; 'Meteor Wars', 'Space Wars', and 'Meteor Mission'. 'Meteor Mission' is supposed to be Asteroids style, 'Space Wars' is like the arcade game, and 'Meteor Wars' is a combination of the two.

I bought Video Games #3 for the 'Meteor Mission'. I expected a reasonable copy of Asteroids.

The Astroids are all single cell (character 226): when you shoot one it dissappears; the Astroids do not break up.

All Astroids travel at the same speed. When your ship hits an Asteroid, it does not reset to the middle of the screen. Instead, it keeps moving with the same direction and velocity. This ship does not slow down eventually on its own, as in the arcade version.

The ship 'jumps' several squares at a time when going fast.

Orion Software did a good job on the movement of the ship and missiles. Unfortunately, 'Meteor Mission' looks a lot like you're battling it out in a bowl of Spaghetti-O's, rather than a field of Asteroids.

The controls for Space Wars are 1,2,3,4,(left player) and 8,9,0,: (right player). It would be nice if it offered several playing variations like the arcade game, and if each player had different style ships. (a tank and an arrow, instead of two arrows)

'Meteor Wars' is a combination of the other two (the star from 'Space Wars' is not included).

The ships, missiles, and meteors can move in 8 directions. This is quite acceptable given the limited graphics on the OSI.

In summary, if you want n home version of Space Wars, and don't mind two players on the keyboard, this is a REASONABLE copy. If you like Asteroids, take your 60 quarters to the arcades.

BY BRUCE ROBINSON

DAVID PITTS, HOUSTON, TEXAS
CORRECTION TO GOBBLER

The Gobbler program has a few bugs in it in addition to the one that you mentioned in the Journal already. Change lines 5,135,160, and 280 as follows:

```
5 FORX=1TO99:NEXT:?"HIT SHIFT TO START"
135 REM FOR DISK POKE530,1=POKE2073,96
160 B=E:M=E
280 IFZ<5THENE=E+L-1
```

To fix the bug in the program where the Gobbler leaves an image of himself behind when he backs up: add the following:

```
190
TIME=TIME+1:IFTIME>40THENPOKEE,32:E=M:B=M

420  P=PEEK(E):IFP<>POTHENE=B:
IFTIME>40THENPOKEE,32: E=M:B=M:GOTO180
```

DONALD SHERRY, G.P.W.,MICHIGAN

Number Square is that plastic game from your childhood that looked something like this:

1	15	14	7
11	▨	6	9
10	12	8	4
3	6	5	13

You can slide 15 into the space and a space will appear between 1 and 14. The aim is to get the numbers in order with the space in thelower right-hand corner. It can be quite a challenge.

The computer program asks what you want as the difficulty. Then it will move the space around in the puzzle 7 times the difficulty (difficulty 3 means it is moved 21 times). It then displays the scrambled puzzle and asks you which number it should move into the space. -RETURN- isn't used, but to use "1", you must press -SPACE- afterwards. If you become totally frustrated with your puzzle, when you input your number, also press -REPT-. The computer will put the display back in the original scrambled display and the show you how to solve it.

The Variables (or at least the main ones):

CP(15) -- current position of argument
NP(16) -- number's place on screen

(I don't have a C1P, so you'll have to find out the numbers for NP(). Also check the corner (C) and line length (1).

```
 100 FORX=1TO32:PRINT:NEXT:POKE11,0:POKE
12,253
 110 POKE56832,4:FORX=57344TO57663:POKEX
,14:NEXT
 120 FORX=XTO57854:POKEX,8:NEXT
 130 FORX=XTO58494:POKEX,12:NEXT
 140 FORX=XTO58879:POKEX,4:NEXT
 150 A=260
 160 FORX=XTO59391:POKEX,6:NEXT
 170 DIMA(A)
 180 FORX=1TOA:READA(X):NEXT
 190 DATA32,32,32,32,32,32,193,32,32,32,
32,32,32,32,32,32,32,32
 200 DATA189,32,190,32,32,32,32,32,32,32
,32,32,189,32,32,32,190
 210 DATA32,32,32,32,32,221,222,201,32,3
2,32,32,32,200,221,222,32
 220 DATA32,202,199,202,32,32,32,32,32,1
99,202,199,32,32,136,143,136,32
 230 DATA32
 240 DATA193,32,32,143,136,143,32,32,136
,143,136,32,189,32,190,32,143
 250 DATA136,143
```

CONT'D

```
 260 DATA32,32,136,143,136,201,32,32,32,
200,143,136,143
 270 DATA32
 280 DATA32,136,143,136,140,166,172,168,
139,143,136,143,32
 290 DATA32,136,143,136,140,32,32,32,139
,143,136,143,32
 300 DATA32,136,143,136,202,32,32,32,199
,143,136,143,32
 310 DATA32,136,143,201,32,32,32,32,32,2
00,136.143,32
 320 DATA32,136,143,202,32,221,219,222,3
2,199,136,143,32
 330 DATA32,136,189,32,32,140,149,139,32
,32,190,143,32
 340 DATA32,189,32,32,32,140,149,139,32,
32,32,190,32
 350 DATA201,32,32,32,32,140,149,139,32,
32,32,32,200
 360 DATA140,32,32,32,32,140,149,139,32,
32,32,32,139
 370 DATA203,132,132,132,132,132,215,132
,132,132,132,132,206
 380 DATA32,136,143,201,32,200,187,201,3
2,200,136,143,32
 390 DATA32,209,208,202,32,199,187,202,3
2,199,209,208,32
 400 PRINT:PRINT:PRINT
 410 FORR=0TO8:PRINTSPC(9);:FORX=1TO13:P
RINTCHR$(A(R*13+X));:NEXT:PRINT
 420 NEXT
 430 FORR=0TO4:PRINTSPC(9);:FORX=1TO13:P
RINTCHR$(A(X+117));:NEXT:PRINT:
 440 NEXT
 450 FORR=10TO19:PRINTSPC(9);:FORX=1TO13
:PRINTCHR$(A(R*13+X));:NEXT
 460 PRINT:NEXT
 470 PRINTSPC(12)CHR$(135)CHR$(135)CHR$(
135)" "CHR$(135)CHR$(135);
 480 PRINTCHR$(135)
 490 GOSUB540
 500 X=USR(X)
 510 FORX=1TO15:PRINTSPC(13)CHR$(161)"
 "CHR$(161):NEXT
 520 PRINT"  THE START OF A NEW SPACE AG
E"
 530 GOTO530
 540 FORD=57986TO59074STEP64:POKED,15:PO
KED+4,15:NEXT
 550 FORX=57987TO57989:POKEX,15:NEXT:FOR
X=58119TO58121:POKEX,15:NEXT
 560 FORX=58567TO58569:POKEX,15:NEXT
 570 FORX=58115TO59011STEP128:POKEX,15:P
OKEX+2,15:NEXT
 580 FORX=58180TO58948STEP256:POKEX,15:N
EXT
 590 RETURN
 600 END
```

CONCENTRATION

```
 20 FORX=1TO6:PRINT:NEXT:REM CONCENTRATI
ON
 30 PRINT:PRINT"HIT SHIFT TO START"
 35 IFPEEK(57088)=1ORPEEK(57088)=254THEN
R=RND(8):GOTO35
 40 PRINT:PRINT:PRINT
 50 INPUT"DO YOU WANT INSTRUCTIONS";A$:I
FA$="YES"THEN570
 70 DIMB(40)
 80 PRINT:PRINT:PRINT:INPUT"HOW HARD SHO
ULD I MAKE IT (FROM 1 TO 10)";Y
 90 TF=11-Y
 100 IFPEEK(57088)<128THEN130
 110 DEFFNA(DI)=2*X+64*Y+53674-32*P
 120 GOTO140
 130 DEFFNA(DI)=2*X+128*Y+54213-64*P
 140 FORX=1TO32:PRINT:NEXTX
 150 DIMA(50):POKE56900,0
 160 FORX=1TO36:A(X)=161:NEXT
 170 FORX=1TO36
 180 READY
 190 M=INT(RND(1)*36+1)
 200 IFA(M)=161THENA(M)=Y:GOTO230
 210 M=M+1:IFM=37THENM=1
 220 GOTO200
 230 NEXTX
 240 FORX=1TO36:B(X)=161:NEXTX
 250 B$="PLAYER ONE"
 260 P=0:FORX=1TO32:PRINT:NEXT
 270 M=1:PRINT"PLAYER ONE   "S1:PRINT:PRI
NT"PLAYER TWO   "S2
 280 PRINT"         1 2 3 4 5 6"
 290 PRINT:FORX=1TO6:PRINTTAB(4)X:PRINT:
NEXT
 300 FORX=1TO6:FORY=1TO6
 310 POKEFNA(DI),B(M)
 320 M=M+1
 330 NEXTY:NEXTX
 340 PRINTB$;:INPUTY,X
 345 IFX>6ORY>6THENP=P+1:GOTO340
 350 T=6*X+Y-6
 360 P=P+1
 370 POKEFNA(DI),A(T)
 380 INPUT"SECOND SQUARE";Y,X
 385 IFX>6ORY>6THENP=P+1:GOTO380
 390 T2=6*X+Y-6
 400 P=P+1
 410 POKEFNA(DI),A(T2)
 420 IFA(T)=A(T2)ANDT<>T2THEN460
 430 P=0:FORX=1TO1000*TF:NEXT
 440 IFB$="PLAYER ONE"THENB$="PLAYER TWO
":GOTO260
 450 GOTO250
 460 B(T)=A(T):B(T2)=A(T22)
 470 IFB$="PLAYER ONE"THENS1=S1+1:GOTO49
0
 480 S2=S2+1
 490 IFS1+S2=18THENFORX=1TO15:PRINT:NEXT
:GOTO510
 500 FORX=1TO2000:NEXT:GOTO260
 510 IFS1>S2THENPRINT"PLAYER ONE WINS"
 520 IFS2>S1THENPRINT"PLAYER TWO WINS"
 530 PRINT"SCORE":PRINT"PLAYER ONE "S1:P
RINT"PLAYER TWO "S2
 540 IFS1=S2THENPRINT"A TIE":PRINT:PRINT
"LETS TRY AGAIN"
 550 FORX=1TO2000:NEXTX
 560 RUN
 570 FORX=1TO32:PRINT:NEXT
 580 PRINT"I WILL PUT UP 36 SQUARES IN A
 SIX BY SIX ARRAY
 590 PRINT"BEHIND EACH SQUARE IS A SYMBO
L
 600 PRINT"I WILL SHOW YOU THE TWO YOU P
ICK ON YOUR TURN
 610 PRINT"YOU GET ONE POINT FOR EACH MA
TCH YOU MAKE
 620 PRINT"THE HARD PART IS REMEMBERING
ALL THE ONES THAT
 630 PRINT"DIDNT MATCH
 635 IFPEEK(57088)>168THENPRINT:INPUT "R
EADY FOR MORE";A$
 640 PRINT:PRINT"YOU PICK A SQUARE BY EN
TERING THE NUMBER OF THE
 650 PRINT"COLUMN AND ROW LIKE THIS 1,4
 660 PRINT:PRINT"IF YOU MAKE A MATCH, YO
U GET ANOTHER TURN
 670 PRINT:PRINT"THE HARDER YOU CHOOSE T
HE GAME (1-10)
 680 PRINT"THE LESS TIME YOU WILL HAVE T
O STUDY
 690 PRINT:GOTO70
 700 DATA16,1,14,26,13,15,12,30,66,50,23
1,239,246,254,248,239
 710 DATA238,219
 720 DATA16,1,14,26,13,15,12,30,50,231,2
39,246,254,248,239,238
 730 DATA66,219
 740 DATA 226,226
```

ALIEN RAIN

```
 1 REM ALIEN RAIN
 2 REM BY BILLY SMITH
 3 REM ONE OF THE BEST C1/C2/C4 GAMES
 4 REM THAT WE HAVE PUBLISHED
 5 REM IN A WHILE
 10 GOTO270
 20 REM KEYBOARD MOVE AND FIRE
 25 P=PEEK(K):IFP=PPTHENRETURN
 30 IFP=LTHENIFCX>-12THENPOKECP,U:CP=CP-
1:CX=CX-1:POKECP,C:RETURN
 35 IFP=RTHENIFCX<12THENPOKECP,U:CP=CP+1
:CX=CX+1:POKECP,C:RETURN
 40 MP=CP-V
 45 POKEMP,U:MP=MP-V:IFPEEK(MP)=UTHENPOK
EMP,F:GOTO45
 50 PM=PEEK(MP):IFPM>UTHEN130
 55 FORA=U-5TOU:POKEMP,A:NEXTA:SC=SC+10
 65 IFCX=-3THENCY=1:GOTO95
 70 IFCX=-6THENCY=3:GOTO95
 75 IFCX=-9THENCY=5:GOTO95
 80 IFCX=3THENCY=2:GOTO95
 85 IFCX=6THENCY=4:GOTO95
 90 CY=6
 95 IFPM=TGTHENSC=SC+40:DB(CY)=-1:GOTO12
0
 100 POKEMP,TR
 105 IFD(CY)=1THENPOKEC(CY)+D(CY)*V,U
 110 D(CY)=D(CY)+1:IFD(CY)>5THEND(CY)=5
 115 POKEC(CY)+D(CY)*V,U
 120 PRINTCHR$(13);"-SCORE-";SC;:IFCO=OT
HEN445
 125 RETURN
 130 IFPM<SRTHENSC=SC-5:RETURN
 135 SC=SC+INT(100*RND(6)+100):POKEDB(7)
,U:POKEDB(7)+1,U
 140 DB(7)=-1:GOTO120
 145 REM ATTACK SEQUENCE
 150 IFDB(7)=-1THEN180
 155 POKEDB(7),U:POKEDB(7)+1,U:DB(7)=DB(
7)+V
 160 POKEDB(7),SL:POKEDB(7)+1,SR:IFDB(7)
<BBTHEN180
 165 FORB=0TOU:POKEDB(7)+B,TR:POKEDB(7)-
B,TR:POKEDB(7)+B,U
 170 POKEDB(7)-B,U:NEXT:CO=CO-1:CX=0:DB(
7)=-1:CP=BB+14:POKECP,C
 175 PRINTCHR$(13);"-SCORE-";SC;SPC(4);"
GUN";CO;:IFCO=0THEN445
 180 FORA=1TO6:IFDB(A)=-1THENNEXT:RETURN

 185 POKEDB(A),U:DB(A)=DB(A)+VV:IFDB(A)<
BBTHENPOKEDB(A),TG:NEXT:RETURN
 190 FORB=1TODD:POKEDB(A)+B,F:POKEDB(A)-
B,F:NEXT
 195 FORB=0TODD:POKEDB(A)+B,U:POKEDB(A)-
B,U:NEXT
 200 IFABS(DB(A)-CP)>DDTHEN215
 205 CO=CO-1:CP=BB+14:CX=0:POKECP,C
 210 PRINTCHR$(13);"-SCORE-";SC;SPC(4);"
GUN";CO;:IFCO=0THEN445
 215 DB(A)=-1:NEXT:RETURN
 220 REM MAIN LOOP
 225 POKETL(CC),TR:GOSUB25 :POKETL(CC),U

 230 IFDB(7)=-1THENIFRND(2)>DITHENDB(7)=
B(INT(3*RND(6)+7))
 235 POKETL(CC),TR:GOSUB25 :GOSUB150:POK
ETL(CC),U:TL(CC)=TL(CC)+DC
 240 IFTL(CC)<>C(CC)THEN235
 245 POKEC(CC)+D(CC)*V,TR:IFD(CC)>0THEND
(CC)=D(CC)-1:IFD(CC)>0THEN255
 250 IFDB(CC)=-1THENDB(CC)=B(CC):D(CC)=D
(CC)+1
 255 CC=CC+1:IFCC>6THENCC=1
 260 DC=-(DC):TL(CC)=MI:GOTO225
 265 INITIALIZE
```

```
 270 PP=PEEK(57100):FORU=1TO31:PRINT:NEX
T:PRINT"ALIEN RAIN!"
 275 PRINT"COPYRIGHT 1981":PRINT"BILLY D
. SMTTH":PRINT:PRINT
 280 PRINT"Alien attack fighters":PRINT"
fill attack tubes in"
 285 PRINT"in order to RAIN down":PRINT"
on your position in"
 290 PRINT"KAMIKAZE fashion.":PRINT:PRIN
T"Landing craft attempt"
 295 PRINT"to land on the surface":PRINT
"and overtake your"
 300 PRINT"position.":PRINT:PRINT"POINT
SCHEME":PRINTCHR$(29)"    10"
 305 PRINTCHR$(213)+CHR$(212)"   ??":PRIN
TCHR$(31)"    50":PRINT
 310 PRINT"MISS   -5 POINTS":PRINT:PRINT"
SHIFT TO START"
 315 IFPEEK(57100)=PPTHENA=RND(A):GOTO31
5
 320 PRINT:PRINT"CONTROLS":PRINT"L SHFT
 LEFT":PRINT"R SHFT   RIGHT"
 325 PRINT"REPEAT   FIRE":PRINT
 330 INPUT"DIFFICULTY(1-5)";DI:IFDI<1ORD
I>5THEN330
 335 DD=DI-1:DI=.99/DI+.3:IFDD>3THENDD=3

 340 V=32:K=57100:BA=53382:L=250:R=252:T
R=29:TG=31
 342 IFPP<128THENV=64:BA=53573:L=5:R=3:P
OKE56832,0
 345 VV=2*V:F=172:SL=213:SR=212:C=215:CO
=3
 350 FORU=1TO31:PRINT:NEXT
 355 FORA=1TO5:POKEBA+A*V,161:POKEBA+2+A
*V,161:POKEBA+3+A*V,161
 360 POKEBA+5+A*V,161:POKEBA+6+A*V,161:P
OKEBA+8+A*V,161
 365 POKEBA+12+A*V,161:POKEBA+14+A*V,161
:POKEBA+15+A*V,161
 370 POKEBA+17+A*V,161:POKEBA+18+A*V,161
:POKEBA+20+A*V,161:NEXT
 375 FORA=1TO3:POKEBA+8+V+A,161:NEXT
 380 CP=BA+10+22*V:POKECP,215
 385 MI=BA+10
 390 BB=BA-4+22*V
 395 FORA=16TO1STEP-1:POKEBA+8+A-V,214:P
OKEBA+10+A-V,211
 400 POKEBA+9+A-V,163:POKEBA+9+A-2*V,193

 405 FORB=0TO25:NEXT
 410 POKEBA+10+A-V,96:POKEBA+9-A-V,96:PO
KEBA+9+A-2*V,96:NEXT
 415 POKEBA+9-V,214:POKEBA+10-V,163:POKE
BA+11-V,211:POKEBA+10-V-V,193
 420 CC=1:TL(CC)=MI:DC=-3:FORA=1TO6:D(A)
=5:NEXT
 425 C(1)=MI-3:C(2)=MI+3:C(3)=MI-6:C(4)=
MI+6:C(5)=MI-9:C(6)=MI+9
 430 B(7)=MI+2*V:B(8)=MI-12:B(9)=MI+11:D
B(7)=-1
 435 PRINTCHR$(13);"-SCORE-";SC;SPC(4);"
GUN";CO;:IFCO=0THEN445
 440 FORA=1TO6:B(A)=C(A)+6*V:DB(A)=-1:NE
XT:GOTO225
 445 PRINT:PRINT:PRINT:PRINT"TOTAL SCORE
=";SC
 450 IFSC>HSTHENHS=SC
 455 PRINT:PRINT"-HIGH SCORE-";HS
 460 PRINT:INPUT"TRY AGAIN";A$:IFASC(A$)
=89THENSC=0:GOTO330
 465 END
```

AARDVARK JOURNAL INDEX
Compiled by Jack Vaughn

AAA

ACEY DUCEY, GAME; AUGUST,80,13 AND CATALOG
ADVENTURE WRITING; PART 1 JUNE,81,1: PART 2 AUGUST 81,1
ADVENTURES, SAVE IN PROGRESS; APR,81,12
ADVERTISING, SAVE IN PROGRESS; APR,81,12
ALIEN INVADERS APDATE; FEB,81,14
ANTI-AIRCRAFT ARTILLERY, GAME; JUNE,81,19
ARRAYS; OCT,80,6
ARTIST, GAME; FEB,81,17
ASSEMBLY SOURCE CODE; JUNE, 81,14
ASSEMBLER C4P; JUNE,81,15
ASSEMBLY PROGRAM SUBROUTINES; AUG,81,7
ASSEMBLER, IT IT CRASHES; AUG,81,14
ASTEROID RACE, GAME; OCT,80,15
ASTRO, GAME; DEC,80,17

BBB

BASIC BLOCK DELETE; AUGUST,81,14
BASICS SECRET FUNCTIONS; JAN 81 CAT P20
BASIC SHORTCUT SINGLE ENTRY COMMAND; APR,81,10
BATTLEFLEET, GAME; APR,81,15/16
CORRECTION, DEC,81,10
BIORHYTHMS; FEB,81,18
BREAK KEY, DISABLE; JUN,80,11 IMMEDIATE RESET, DEC,81,14
BREAK MOD; AUG,80,6
BREAK KEY MOD; JUNE,81,15: 8PDF OCT,81,14
E-Z BREAK MOD FOR ANYBODY; AUG,81,17
BREAKPOINT UTILITY; FEB,81,8

CCC

C1P 2 MHZ FOR DISC; OCT,81,7
C1P HARDWARE PROBLEM; APR,80,9
C1P/HYCOM PRINTER PARALLEL INTERFACE; FEB,81,9
C1P RESET MOD; JUN,81,15
C4P DUST COVER; AUG,81,16
C8P, SPEED UP DIRECT FILES; OCT,81,8
ADDITIONAL COMMENTS, DEC,81,3/4
CALCULATOR, SIMPLE, OCT,81,2
CASSETTE PROBLEMS; FEB,81,5
CASSETTE MOD; APR,81,13
CHANGING THE 'OK' PROMPT; OCT,81,5
CHRISTMAS CARD, C1P; DEC,80,18
COMMISSIONS, AARDVARK; AUG,80,9
COMMENTS ON GT. CONV. & REV. VIDEO,DEC,81,11
COMPILER, PEGASUS, DISC; FEB,81,15
COMPILER, AARDVARK; AUG,81,20
CONTINUED FUNCTION; JUN,80,8
CONTROLS; APR,81,3
CURSOR CONTROL INFO; APR,80,9
CURSOR,C2/4 UPDATE; DEC,80,16:FEB,81,13

DDD

D & D GAMES REVIEW; AUG,80,7
DATA FILES DISK; JUN,80,4
DATA STORAGE STRINGS; APR,80,6
DEATH SHIP ADVENTURE; JUN,81,4
DELETE, BASIC BLOCK; AUG,81,14
DISC, 14 CHAR. FILE NAMES; FEB,81,20 (BC)
DISC, ADDING; JUN,81,6
DISC, GRAPHICS OS65D; JUN,81,17
DISC, IMPROVE RANDOM ACCESS FILES; JUN,81,11
DISK SYS. OS65D; JUN,80,2
DISC RECORDS; OCT,80,14
DISC, CHANGE BUFFER LOC; FEB,81,8: JUN,81,16
DISC HEAD UNLOAD; APR,81,8
DISC MOTOR CONTROL; AUG,81,10
DISC PASSWORD; AUG,81,17
DISC QUIET; AUG,81,4: CORRECTION OCT,81,7: OCT,81,15
DISC TRICK, OS65D V.3; AUG,81,18
DISC, OS65D V.3/3.1 POKES; AUG,81,19
DISC, OS65D WITH C1E ROM; OCT,81,11
DISC, NON O.S.I.; OCT,81,12
DISPLAYS, APR,81,2
DOLLARS AND CENTS; FEB,81,13
DOLLARS TO STRINGS; AUG,81,18
DOUBLE SPACE; APR,80,9

EEE

EPSON PRINTER INTERFACE; AUG,81,15
ERROR EXPLANATION; DEC,80,2

FFF

FLIGHT/MILLENIUM FALCON; JUN,81,19
FUNCTION PLOTTER FOR C1P, DEC,81,8

GGG

GAME,ACEY DUCEY; AUG,80,13 AND CATALOG
GAME, ANTI-AIRCRAFT ARTILERY; JUN,81,19
GAME, ARTIST; FEB,81,17
GAME, ASTEROID RACE; OCT,80,15
GAME, ASTRO; DEC,80,17
GAME, BATTLEFLEET; APR,81,15/16 GAME, CONCENTRATION, DEC,81,14
GAME, HANGMAN; JUN,80,14
GAME, HELICOPTER PILOT; DEC,80,17: CORRECTION FEB,81,16
GAME, FLIGHT/MILLENIUM FALCON; JUN,81,19
GAME, GLEEP; DEC,80,19
GAME, GOBBLER; FEB,81,17 CORRECTION, DEC,81,13

RRR

RACECOURSE, GAME; APR,80,12
RANDOM ACCESS FILE; OCT.81,7
RANDOM NUMBERS, TRUE; JUN,80,7
RANDOM (X) WON'T WORK; JUN,80,3
RANDOM NUMBERS RANDOM SEQUENCE; AUG,81,8
RATTLESNAKE, GAME; AUG,81,9
RECORD SIZE, CHNGING DISK; JUN,80,4
RELOCATOR PROGRAM; AUG,81,6
'REM' ERASER; APR,81,12
REVERSE VIDEO MOD USING SOUND BD; APR,81,11
REVERSE VIDEO MOD; JUN,81,7
ROM, EXPAND CAPABILITY, DEC,81,8

SSS

S-100 BUSS, DEC,81,9
SAVING MEMORY; OCT.80,2 AND 9
SCREEN CLER; AUG,80,1:C4P; OCT,80,13
SCROLL DOWN, NOT UP; APR,81,13
SHIFT, PRESS TO START; JUNE,80,7
SINGLE STEP TRACE AND LISTER; OCT,81,6
SKEET, GAME; APR,81,17: JUNE,81,14
SLASHBALL, GAME; APR,81,17:JUNE,81,14
SOUND CHIP, DEC,81,9/10
SOUND BOARD, DECODES--; APR,81,11
SPACE SAVING (HIDDEN); AUG,80,5
SPEED IN BASIC; AUG,80,2
STANDBY POWER SUPPLY FOR C1/4P; AUG,81,6
STRINGS, AARDVARK TUTORIAL; APR,80,2
STRINGS FOR DIGITAL MANIPULATION; APR,80,5
STRINGS, DIMENSIONED; APR,80,6
SUPERBBOARD II SERIES TWO; APR,81,11
SUPERBOARD SUPER SOUND; AUG,81,11
SUPERBOARD JOYSTICK INTERFACE; AUG,81,11

TTT

TAPE TO DISC; AUG,80,9
TELETYPE, WITH C1P; OCT,80,12
TINY COMPILER, DEC,81,2
TRACE,SINGLE STEP AND LISTER; OCT,81,6
TROUBLESHOOTING BASIC; DEC,80,2
T-V MONITOR MOD; AUG,81,18

UUU

VVV

VARIABLE NAMES; JUN,80,2
VARIABLES, SUBSCRIPTED; OCT,80,6
VIDEO MOD C1P; AUG,80,6
VIDEO MOD FOR OVERSCAN C1 SB; DEC,80,7:CORRECTION FEB,81,1
VIDEO MOD CORRECTION; APR,81,13
VIDEO MOD MOVING YOKE; DEC,80,15

WWW

WAMPUS, KILL A-GAME; AUGUST,81,7/8
WORD PROCESSOR,SIMPLE; JUN,80,12
WORD PROCESSOR, CASS; JUN,81,10
WORD PROCESSOR MOD; OCT,81,5
WP 6502 WITH C2E ROM; JUN,81,14
WRITE FOR AARDVARK; APR,80,8

XXX
YYY
ZZZ

```
 10                     ;POLLED KEYBOARD INPUT ROUTINE
 20                     ;
 30                     ;COPYRIGHT BY DON VANSYCKEL 1981
 40                     ;
 50                     ;"UNDERLINE" OR SHIFT"O" IS RPT"O"
 60                     ;SHIFT"N" IS RPT"N"
 70                                              ;
 80 DF00=               KYBD    = $DF00          ;KYBD ADDR
 90                     ;STORAGE STARTS AT $0213 FOR APPLICATION
100 0213=               CHR2    = $0213          ;CHAR
110 0214=               TEMP    = CHR2+1         ;TEMP
120 0215=               CHR1    = TEMP+1         ;CHAR
130 0216=               CNT     = CHR1+1         ;COUNT
140                     ;PROGRAM STARTS AT $FD00 FOR APPLICATION
150 FD00                        *=$FD00
160 FD00 2011FD KYBDWT  JSR KYBDRD       ;READ UNTIL CHARACTER
170 FD03 F0FB           BEQ KYBDWT
180 FD05 60             RTS
190 FD06 20C0FD KYAA    JSR RD01         ;READ ROW '01'
200 FD09 2907           AND #$07         ;SHIFT?
210 FD0B D06E           BNE KY06         ;YES, JUMP
220 FD0D A020           LDY #$20         ;NO, OFFSET
230 FD0F D06A           BNE KY06         ;JUMP ALWAYS
240 FD11 8A     KYBDRD  TXA              ;SAVE X & Y
250 FD12 48             PHA
260 FD13 98             TYA
270 FD14 48             PHA
280 FD15 20C0FD KY01    JSR RD01         ;LOAD ROW '01'
290 FD18 2920           AND #$20         ;ECS?
300 FD1A F018           BEQ KY02         ;NO, JUMP
310 FD1C A91B           LDA #$1B         ;LOAD 'ESC'
320 FD1E D078           BNE KY10
330 FD20 8D1502 KYBB    STA CHR1
340 FD23 A902           LDA #$02         ;LOAD COUNT
350 FD25 8D1602         STA CNT
360 FD28 A005   KYCC    LDY #5           ;DELAY
370 FD2A A2C8   KYDD    LDX #$C8
380 FD2C CA     KYEE    DEX
390 FD2D D0FD           BNE KYEE
400 FD2F 88             DEY
410 FD30 D0F8           BNE KYDD
420 FD32 F0E1           BEQ KY01
430 FD34 A201   KY02    LDX #$01
440 FD36 8A     KY03    TXA              ;SET UP NEXT ROW
450 FD37 0A             ASL A
460 FD38 AA             TAX
470 FD39 D005           BNE KY04         ;END?
480 FD3B 8D1502         STA CHR1
490 FD3E F062           BEQ KY11
500 FD40 20C2FD KY04    JSR RD           ;READ KYBD
510 FD43 F0F1           BEQ KY03         ;NO KEYS, JUMP
520 FD45 20B9FD         JSR CONV         ;CONVERT COL BIT TO NUMBER
530 FD48 8C1402         STY TEMP         ;SAVE
540 FD4B 8A             TXA
550 FD4C 20B9FD         JSR CONV         ;CONVERT ROW BIT TO NUMBER
560 FD4F 98             TYA              ;MULTIPLY X 8
570 FD50 0A             ASL A
580 FD51 0A             ASL A
590 FD52 0A             ASL A
600 FD53 6D1402         ADC TEMP         ;ADD COL
610 FD56 A8             TAY
620 FD57 B9C9FD         LDA TABLE,Y ;READ TABLE
630 FD5A A005           LDY #5           ;5
640 FD5C D9DBFD KY05    CMP EXC-1,Y  ;NON-SHIFTED KEYS
650 FD5F F01F           BEQ KY07
660 FD61 88             DEY
670 FD62 D0F8           BNE KY05
680 FD64 8D1402         STA TEMP
690 FD67 AA             TAX
700 FD68 109C           BPL KYAA         ;POS, JUMP
710 FD6A A080           LDY #$80         ;OFFSET
720 FD6C 20C0FD         JSR RD01
730 FD6F 2906           AND #$06         ;SHIFT?
740 FD71 F008           BEQ KY06         ;NO, JUMP
750 FD73 A090           LDY #$90         ;OFFSET
760 FD75 E0B0           CPX #$B0         ;WHICH ROW?
```

```
 770 FD77 3002            BMI KY06
 780 FD79 A070            LDY #$70     ;OFFSET
 790 FD7B 18       KY06   CLC          ;ADD OFFSET
 800 FD7C 98              TYA
 810 FD7D 6D1402          ADC TEMP
 820 FD80 A8       KY07   TAY
 830 FD81 20C0FD          JSR RD01     ;LOAD ROW '01'
 840 FD84 AA              TAX          ;SAVE
 850 FD85 2980            AND #$80     ;RPT?
 860 FD87 F005            BEQ KY08     ;NO, JUMP
 870 FD89 18              CLC          ;YES, ADD OFFSET
 880 FD8A 98              TYA
 890 FD8B 6910            ADC #$10
 900 FD8D A8              TAY
 910 FD8E 8A       KY08   TXA          ;TEST FOR CTRL
 920 FD8F 2940            AND #$40
 930 FD91 F004            BEQ KY09
 940 FD93 98              TYA
 950 FD94 293F            AND #$3F
 960 FD96 A8              TAY
 970 FD97 98       KY09   TYA
 980 FD98 CD1502 KY10     CMP CHR1     ;SAME AS LAST CHAR?
 990 FD9B D083            BNE KYBB     ;NO, JUMP
1000 FD9D CE1602          DEC CNT      ;COUNT=0?
1010 FDA0 D086            BNE KYCC     ;NO, JUMP
1020 FDA2 A296     KY11   LDX #$96     ;DELAY COUNT
1030 FDA4 CD1302          CMP CHR2     ;REPEAT CHAR
1040 FDA7 D002            BNE KY12     ;NO, JUMP
1050 FDA9 A214            LDX #$14     ;DELAY COUNT
1060 FDAB 8E1602 KY12     STX CNT      ;SAVE COUNT
1070 FDAE 8D1302          STA CHR2     ;SAVE CHAR
1080 FDB1 68              PLA          ;RESTORE X & Y
1090 FDB2 A8              TAY
1100 FDB3 68              PLA
1110 FDB4 AA              TAX
1120 FDB5 AD1502          LDA CHR1
1130 FDB8 60              RTS
1140 FDB9 A0FF     CONV   LDY #$FF     ;CONVERT BIT TO NUMBER
1150 FDBB C8       C001   INY
1160 FDBC 0A              ASL A
1170 FDBD 90FC            BCC C001
1180 FDBF 60              RTS
1190 FDC0 A901     RD01   LDA #$01
1200 FDC2 8D00DF RD       STA KYBD     ;READ KYBD
1210 FDC5 AD00DF          LDA KYBD
1220 FDC8 60              RTS
1230 FDC9 B1       TABLE  .BYTE $B1,$B2,$B3,$B4,$B5,$B6,$B7,0
1230 FDCA B2
1230 FDCB B3
1230 FDCC B4
1230 FDCD B5
1230 FDCE B6
1230 FDCF B7
1230 FDD0 00
1240 FDD1 B8              .BYTE $B8,$B9,$30,$3A+$80,$2D+$80,$7F,0,0
1240 FDD2 B9
1240 FDD3 30
1240 FDD4 BA
1240 FDD5 AD
1240 FDD6 7F
1240 FDD7 00
1240 FDD8 00
1250 FDD9 AE              .BYTE $2E+$80,'LO'
1250 FDDA 4C
1250 FDDB 4F
1260 FDDC 0A       EXC    .BYTE $0A,$0D,$20,$30,$7F
1260 FDDD 0D
1260 FDDE 20
1260 FDDF 30
1260 FDE0 7F
1270 FDE1 57              .BYTE'WERTYUI',0
1270 FDE2 45
1270 FDE3 52
1270 FDE4 54
1270 FDE5 59
1270 FDE6 55
```

```
1270 FDE7 49
1270 FDE8 00
1280 FDE9 53           .BYTE'SDFGHJK',0
1280 FDEA 44
1280 FDEB 46
1280 FDEC 47
1280 FDED 48
1280 FDEE 4A
1280 FDEF 4B
1280 FDF0 00
1290 FDF1 58           .BYTE'XCVBNM',$2C+$80,0
1290 FDF2 43
1290 FDF3 56
1290 FDF4 42
1290 FDF5 4E
1290 FDF6 4D
1290 FDF7 AC
1290 FDF8 00
1300 FDF9 51           .BYTE'QAZ  ',$2F+$80,$3B+$80,'P'
1300 FDFA 41
1300 FDFB 5A
1300 FDFC 20
1300 FDFD AF
1300 FDFE BB
1300 FDFF 50
```

AARDVARK
TECHNICAL SERVICES
2352 South Commerce
Walled Lake, MI 48088