

1981

PAGE

CONTENT

1

Letters

4

Notebook

5

Software
Reviews

6

Hardware
Reviews

11

Software

Appendix

6502
CLASS

cover
Shel
sacks

MARCH
1981

All contents
copyright
1981

No Publication
Rights
without

By the
Author(s)
OSUNY

Express
Permission
of the Author

6502

DEAR EDITOR,

I AM PROMPTED TO WRITE THIS LETTER BECAUSE OF TWO THINGS I HAVE READ RECENTLY. ONE OF THEM WAS LAST MONTH'S EDITORIAL IN 'OS-ITEMS'. THE OTHER WAS THE SOFTWARE PROTECTION COLUMN IN THE JAN. 15 'COMPUTER SHOPPER'.

IN THE 'OS-ITEMS' EDITORIAL IT WAS IMPLIED THAT PEOPLE WERE JUSTIFIED IN COPYING PROGRAMS BECAUSE OF THE HIGH PRICE AND POOR SELECTION. THE 'COMPUTER SHOPPER' COLUMN GAVE ITS UNQUALIFIED BLESSING TO SOFTWARE PIRACY PROFFERING THE 'REASONS' OF HIGH PRICES, POOR QUALITY, AND THE QUESTIONABLE (THEIR ACTUAL WORDS WERE "GET RICH QUICK") MOTIVES OF PEOPLE IN THE SOFTWARE INDUSTRY. BOTH AUTHORS SEEM TO FEEL THAT IF COMPUTER HOBBYISTS 'PUNISH' US NASTY SOFTWARE HOUSES ENOUGH BY COPYING PROGRAMS, WE WILL LOWER PRICES AND IMPROVE QUALITY SO THAT THEY WON'T BE 'FORCED' TO RESORT TO SUCH EXPEDIANTS ANYMORE.

LET'S FORGET, FOR THE MOMENT, WHAT WE ALL KNOW, THAT COPYING SOFTWARE IS ILLEGAL. THESE AUTHORS, AND MOST COMPUTER HOBBISTS, SEEM THINK THAT THIS IS AN UNIMPORTANT TECHNICALITY (AFTER ALL AREN'T LAWS MEANT TO BE BROKEN?). LET'S LOOK AT WHAT WILL BE ACCOMPLISHED BY FOLLOWING THE STRATEGY THAT THEY ESPOUSE.

WHO MAKES UP THE HOBBYST SOFTWARE INDUSTRY? AS I SEE IT, THE INDUSTRY NOW HAS THREE LEVELS OF COMPETITOR. FIRST THERE ARE THE 'COTTAGE INDUSTRY' TYPES. THESE ARE THE PEOPLE WHO GENERALLY HAVE A FEW PROGRAMS TO SELL; USUALLY USE CLASSIFIED ADS. OR SMALL DISPLAY ADS; AND ADVERTISE INFREQUENTLY. IF ANY OF THESE PEOPLE HARBORED ANY 'GET RICH QUICK' IDEAS THEY WERE DISPELLED QUITE SOON WHEN THEY LEARNED THAT ORDERS DON'T COME POURING IN. AND YET THIS SECTION OF THE INDUSTRY IS VERY IMPORTANT, IN AS MUCH AS SOME VERY GOOD AND USUALLY INEXPENSIVE SOFTWARE CAN COME FROM SUCH PEOPLE WHO ARE CARRYING LITTLE, IF ANY, OVERHEAD. REMEMBER, AT ONE TIME THE WHOLE MICROCOMPUTER INDUSTRY WAS 'COTTAGE INDUSTRY'. EVENTUALLY SOME OF THESE PEOPLE WILL GROW INTO INDUSTRY LEADERS. LOOK AT OHIO SCIENTIFIC OR ADVENTURE INTERNATIONAL.

THEN THERE ARE THE PEOPLE LIKE ORION. WE HAVE GOTTEN INTO THE INDUSTRY WITH A LITTLE MORE CAPITAL. WE MAINTAIN A REAL PLACE OF BUSINESS, AND ARE TRYING TO PROVIDE SOFTWARE THAT WORKS WELL, AT A REASONABLE PRICE. MOREOVER WE HOPE TO PROVIDE THE CONVENIENCE THAT YOU HAVE COME TO EXPECT WITH OTHER GOODS; THAT IS, OVER-THE-COUNTER SALES AT LOCAL DEALERS, WHERE YOU CAN 'TRY BEFORE YOU BUY' AND RECEIVE SUPPORTIVE SERVICES IF YOU REQUIRE THEM. WE ARE IMPORTANT TO THE INDUSTRY BECAUSE WE KEEP THE 'BIG BOYS' FROM RAISING THEIR PRICES ASTRONOMICALLY AND BECAUSE, IF WE HIT UPON A GOOD PROGRAM, WE HAVE THE MONEY, DEALERS AND RESOURCES TO DEVELOP AND POPULARIZE IT.

LASTLY THERE ARE THE 'INDUSTRY LEADERS'. YOU KNOW WHO THEY ARE, BUT IF YOU DON'T, THINK OF INSTANT SOFTWARE (KILOBAUD/MICROCOMPUTING), CREATIVE COMPUTING SOFTWARE, ADVENTURE INTERNATIONAL, ETC. THESE GUYS HAVE LOTS OF CAPITAL, AND IN MOST CASES, HAVE PUBLICATIONS BEHIND THEM PROVIDING LOTS OF FREE PUBLICITY.

IF HOBBISTS FOLLOW THE ABOVE MENTIONED STRATEGY, THE RESULTS WILL PROBABLY BE AS FOLLOWS. COTTAGE INDUSTRY OPERATORS WILL ALMOST UNDOUBTEDLY BE FORCED OUT OF THE MARKETPLACE. AS SOON AS THEY WOULD ANNOUNCE A NEW PROGRAM SOMEONE FROM MOST CLUBS OR 'LIBRARIES' WOULD

ORDER ONE COPY OF THE PROGRAM AND THE COPYING PROCESS WOULD BEGIN. UNLESS THESE PEOPLE ARE UNBELIEVABLY PROLIFIC WRITERS, THEY WILL NOT BE ABLE TO EXIST ON THE SCORE OF SO COPIES OF EACH NEW PROGRAM THAT THEY WOULD SELL.

FOR THE MID-SIZED COMPANYS, WIDESPREAD PIRATING WOULD MEAN A SLOW DEATH, BUT EVENTUAL DEATH NONTHELESS. BECAUSE THESE COMPANIES USUALLY HAVE A FEW DEALERS, THEY CAN SELL SOME PROGRAMS TO NEWCOMERS TO THE COMPUTER FIELD. BUT AS SOON AS THESE PEOPLE BECOME FAMILIAR WITH THE WIDESPREAD PIRATING, THEY WILL STOP BUYING PROGRAMS TOO.

I DON'T THINK THAT WIDESPREAD PIRATING WOULD KILL INSTANT OR CREATIVE. THEY HAVE THE FINANCIAL BACKING OF THEIR MAGAZINES BEHIND THEM. THEY WOULD TAKE THEIR LOSSES AND BIDE THEIR TIME UNTIL THEY HAD A CLEAR FIELD TO THEMSELVES. THEN THEY WOULD COME DOWN ON SOFTWARE PIRATING LIKE A TON OF BRICKS, AND THEY WILL HAVE THE MONEY TO DO IT. REMEMBER INSTANT SOFTWARE'S \$10,000 BOUNTY OFFER OF A FEW YEARS AGO? COMBINE BOUNTIES WITH THE MONEY AND LEGAL STAFF TO MAKE A FEW COPYRIGHT INFRINGEMENT CASES STICK, AND THEY WILL BEGIN TO REVERSE THE PROCESS THAT PUT THEM IN UNDISPUTED HOLD OF THE FIELD.

LOOK BEYOND THE HORIZION, AND YOU WILL SEE THAT THEN, WITH ONLY A FEW PEOPLE IN THE INDUSTRY, THEY CAN BAND TOGETHER AND AGREE ON SOFTWARE/HARDWARE PROTECTION FOR THEIR PROGRAMS. ROM ADAPTERS EMBODYING SOFTWARE AND/OR HARDWARE PROTECTION WOULD BE POSSIBLE. SERIALIZED MASTER PROGRAMS REQUIRING SERIALIZED SUBPROGRAMS (MUCH LIKE THE AUTOMATED SIMULATIONS SERIES) WOULD CUT DOWN ON PIRACY SINCE AFTER BUYING THE MASTER PROGRAM DIRECT, YOU WOULD BE FORCED BUY MATCHING SERIAL NUMBERED PROGRAMS AGAIN DIRECT, FOR PROPER OPERATION.

WITH COMPUTER COMPANIES RELYING MORE AND MORE ON SOFTWARE SUPPORT TO SELL THEIR PRODUCTS (E. G. RADIO SHACK), A FEW VERY POWERFUL SOFTWARE HOUSES MIGHT REQUIRE NEW MODEL COMPUTERS TO HAVE ROM PACKS (LIKE THE ATARI) ONLY MUCH MORE DIFFICULT TO 'BREAK' FROM A SOFTWARE-STEALING STANDPOINT. FURTHERMORE THE DAY OF THE CUSTOM MICROPROCESSOR IS UPON US. MANY CHIP MANUFACTURERS NOW OFFER MICROPROCESSOR DEVICES THAT CAN HAVE THEIR LANGUAGE INTERPRETER OR COMPILER BURNED RIGHT ON THE CHIP, TO THE CUSTOMER'S SPECIFICATIONS. THIS WILL ALLOW A COMPUTER MANUFACTURER TO PROTECT LARGE CHUNKS OF MEMORY FROM THE PRYING EYES OF POTENTIAL SOFTWARE THIEVES (WITNESS TI'S DISABLING OF THE PEEK COMMAND WITHIN THE AREA OCCUPIED BY THEIR BASIC AND OS). THE POTENTIAL METHODS OF SECURITY THAT WILL COME INTO COMMON USE IF PIRACY CONTINUES ARE STAGGERING.

WHAT WILL HAPPEN, SHOULD ALL OF THIS COME TO PASS, IS THAT YOU WILL BE PAYING UNBELIEVABLY HIGH PRICES FOR SOFTWARE OF WHATEVER QUALITY THESE HOUSES BELIVE IS SUFFICIENT. THE PROGRAMS WILL PROBABLY BE IN AN 'UNBREAKABLE' ROM OR IN AN 'UN-COPIABLE' SERIALIZED FORM. SOUND UNBELIEVABLE? ALL OF THIS HAS ALREADY COME TO PASS IN THE 'MAINFRAME' INDUSTRY WHICH HAS BEEN BATTLING THE SAME PROBLEMS FOR YEARS. THE COMBINATION OF SOFTWARE THEFT AND TREND TO MAKE HOME COMPUTERS MORE 'APPLIANCE-LIKE' MAY MEAN THAT YOUR NEXT COMPUTER MAY HAVE NO REMOVABLE STORAGE AT ALL; MERELY AN ASSORTMENT OF ROM PACKS, FOR SYSTEMS PROGRAMS, AND LARGE CAPACITY NON-VOLITILE MEMORY (MAGNETIC BUBBLE OR VIDEO DISK) FOR USER PROGRAMS. NO CASSETTE OR DISK!

WHAT DOES A HEALTHY SOFTWARE INDUSTRY HAVE TO OFFER YOU?

CONVENIENCE

~~CONVENIENCE~~ IS ONE THING. WOULDN'T IT BE BETTER FOR YOU TO BE ABLE TO BUY A 'CANNED' PROGRAM THAT RUNS WELL SO YOU COULD SPEND YOUR TIME ENJOYING YOUR COMPUTER RATHER THAN PROGRAMMING IT? SOPHISTICATION IS ANOTHER. PROFESSIONAL PROGRAMMERS WILL HAVE THE TIME AND EXPERTISE TO OFFER REALLY GREAT GAMES OR APPLICATIONS PROGRAMS (WITNESS ATARI'S STAR RAIDERS, WHICH TOOK 8 MAN MONTHS FOR A PROFESSIONAL PROGRAMMER TO COMPLETE), WHEREAS YOU MIGHT NOT. SOFTWARE HOUSES CAN OFFER NEITHER OF THESE IF THEY ARE DENIED THE MONEY TO DO THEM WITH. COST SAVINGS IS ANOTHER ADVANTAGE. IN ANY MONOPOLISTIC INDUSTRY THE 'INDUSTRY LEADERS' CAN SET PRICES AS THEY WILL; THIS IS EXACTLY WHAT WILL HAPPEN WHEN ONE COMPANY CAN FORCE YOU TO BUY FROM THEM. LOOK, FOR EXAMPLE, AT STAR RAIDERS; IT'S GREAT PROGRAM, BUT SINCE YOU MUST BUY IT FROM ATARI ON THEIR ROM CARTRIDGE, IT COSTS YOU \$60.00.

IN ALL I THINK THAT HOBBYISTS HAVE A LOT MORE TO LOSE THAN THEY STAND TO GAIN BY SOFTWARE PIRACY. THE SHORT-TERM ADVANTAGES WILL PROBABLY NEVER MATERIALIZE AND THE LONG TERM EFFECTS COULD BE DISASTEROUS.

TERRY TERRANCE
ORION SOFTWARE ASSOCIATES

NOTE\$ FROM THE NOTE\$ BOOK

-an occassional column of notes and comments
by Shel Sacks

This column will be a fairly frequent collection of comments, questions, hardware and software ideas; I'll be using it as a place to "jot down" the thoughts and questions of a (CIP) user. A lot of the contents will be fairly basic stuff; the hardware ideas (and plans) will be very basic, and anyone who has ever soldered a wire will be able to do them- my electronics background is very limited, so I tend towards the "brute force" approach-- keep it simple, and don't touch the machine! Oh, yes--they'll be CHEAP!

This is my first time as editor of this rag; it's been a lot easier than I thought it would be, mostly thanks to the efforts of Ugo Re, our president. (Thanks, Ugo.) A few comments:

Maybe we should think about printing a list of names and phones of members; I found it very difficult getting hold of people when I needed information.

Contributors-it would be very helpful if you would give your name and number when you submitted your articles/software- there's a beautiful article in this issue, and I'll be damned if I know whom it's from! Also, on programs: Please include info on which machine it was written on, and what machine it will run on.

When submitting programs on cassette, try recording them at a couple of different volume settings if possible--I had considerable problems loading Pete Schreiber's program in this issue.

You'll notice that I used a slightly different format for printing Pete's and my Programs in this issue- two columns, half a page wide each. When I typed out my program the first time, it was a page and a half, due to the use of less than full lines; in the new format, it fit comfortably in a page, with (a little) space left over. (I'm an ecologist by training and) I hate to see our natural resources wasted. Not to mention the time involved in copying the extra pages. So think about it next time you're typing out a program (I can't imagine how you'd format it for a Printer, though). Also-we don't really need double spacing, do we??

While having trouble loading Pete's program, I decided to fool around with my phone and the computer; I thought I would have to call him and have it transmitted over the phone. I have no modem, so I tried an experiment: Taking two audio patch cables (RCA phone plugs, both ends), four wires with alligator clips on the ends (Radio shack, about \$3 for ten, assembled), I then unscrewed the microphone and speaker from one of my two house phones, and with the alligator clips, attached the terminals in the phone handset directly to the load side of the computer (from the speaker side of the handset); I then hooked up my cassette recorder (monitor/earphone plug) to the microphone side of the other phone. And entered the program from one house phone to the other; it works. Since it's a direct connection, there's no acoustical noise problem; and I don't see a Ma Bell overload as likely because I went through the handset and phones. If anyone can see a problem with my computer arising from this, please let me know; but it seems like an awfully inexpensive way of computer-communication over the phone lines. And if the cassette recorder is necessary (to get the proper amplification of the signal), most recorders can be used as amplifiers by running the mic. side of the recorder to the computer, and the earphone/monitor to the phone. It's a much faster way of sending programs and articles than the mails (and it's probably as cost-effective, too). Comments??

In case anyone wants to demonstrate "STRING*BUG" in 20 bytes, here it is:
5 DIMIS(100):7FRE(X)+----then run it.

If anyone's interested--it's not difficult or expensive to set ourselves up as a non-profit corporation (I've done it in California). This would open up all kinds of opportunities for companies such as M/A-COM, Intel, and others to give us all kinds of things; and Polk's could probably take a write-off on our use of the facilities, copier, etc. Anyone interested??

In browsing through programs in OSItems, Micro, and other OSI software sources, I've noticed statements and functions used in ways I was not familiar with; nowhere in the manuals that come with the machine is there an example of the "Input" statement used like this:10Input"YorN";x\$ (which is equivalent in output to :10?"YorN";Inputx\$). In Pete's program in this issue lines 290,310 and 320 contain the statement"290Ifx then..."which seems to imply "if x is true (=1?) then..." Again I've not seen this explained in any of the (meager) literature I have. Which leads me to wonder... How many other commands am I (and others?) not aware of that would be very useful in our own programming? So...how about a contest (or something) to see who can find/create the most"NON-OSI acknowledged" commands, statement and one line routines,(and uses of existing ones)?? For example, Many BASIC have a command "PRINT AT" which will print anything starting at the location following the "AT"; For OSI BASIC, we have to first make a string <I\$=STR\$(I)>, and then:FOR J=0 TO LEN (I\$)-1:POKE <starting location>+J,ASC(MID\$(I\$,J,1)) I worked this up myself, before discovering that it had been published in OSItems several times. A perfect example of re-inventing the wheel--so what say to contributing all those one liners, etc., and saving ourselves the time and energy for more important things--like the programs!

Since I'm this month's editor, I'd like to take a little time to respond to the letter in this issue.
I agree wholeheartedly with the need to protect the authors of software, in whatever form, for fear that we may,indeed,lose them if we steal their software. But there are shades of grey involved here, also--take the cases of the copying of parts of books and magazines due to the widespread availability and low prices of commercial copiers. This is a problem the Feds haven't worked out yet! When is an article(or a piece of software) merely "information" and thus becomes "disseminable"(is that a word?)? Why is there a difference between my copying a page(or 2 or 3) and someone telling it to me verbally? (the former may be illegal, the latter, never.) I, for example, would love to have a copy of "THE FIRST BOOK OF OSI" (blue version)--but I'll be damned if I'll spend \$16 to buy a"flimsily" bound, 40 or so page xeroxed collection of typewritten pages. So when I need some information, I've been calling someone with the book, and asking them. Perfectly legal AND moral. Seems like a pretty fine line, to me. And, no--I haven't as yet borrowed the book and copied it, although I've been tempted. Since I hope to write software someday, I can't see myself being hypocritical. But I do understand the temptation. I simply cannot afford to buy all the software I'd like to have; I believe the \$10, or so, pieces of software are really quite a bargain, given the time it would take me to write them--that is, even if I could. But what about the \$25 ones, not to mention the \$100 ones?? I don't know. I'm in a real quandary about it myself, and haven't really come to a firm decision yet. At the moment I'm leaning strongly towards doing without, since I can't afford; but I can't say how I'll feel about it in the future--especially if the"BIG Corps."start to try to put the screws on.

COMMENTS???

PROGRAM REVIEWSScale 1 to 10

Author: Robert J. Retelle
 2005 Whittaker Rd.
 Ypsilanti, Michigan, 48197

- 1) "Rebel Gunner" Rev. 12-A \$9.95
 Rating: 8.5 Three Levels
 You're Flying An Xwing Fighter.
 You have to destroy TIE fighters which maneuver very quickly.
 Once you lock on to them, in the Crosshairs, your ship automatically fires four phasers, blowing up the enemy fighter. It's not easy!!
- 2) "AAARRRGGG!!!" with Sound \$7.95
 Rating: 8.0 Five Levels
 Fast Paced. You chase around the screen trying to hit many elusive targets. It's a race against time and you can get a bonus for high score.
- 3) Nike Base Rev. 3 with sound
 Rating 6 \$4.50
 Good Game. Displays Good Graphics.
- 4) Barrier Tank Rev. 5-A \$7.95
 Rating: 7.5 Two Levels

An enemy tank places barriers in various parts of the screen. You have to catch the enemy tank and destroy it in order to keep the the area clear. This is difficult and the spaces keep filling up making it even more difficult. The second level has the enemy tank laying explosives in your path!!

 BILLS MICO SERVICES
 210 South Kenilworth
 Oak Park, Illinois, 60302

I sent for five programs:

#1001 Builder \$5.95
 #1004 Plist \$7.95
 #1008 Renumber 1 \$4.95
 #1016 Trek 111P \$6.95
 #10015 Checking & Disbursements \$7.95

Not one of them worked. I sent them all back requesting a replacement. It's just about one week now. Hopefully I'll have it by the next meeting in February and I'll let you know if it worked out. Bills Mico service sends a contract stating you will not duplicate except for a backup copy.

Warren Modell, V. P. 3133 Rochambeau Avenue, Bx. N.Y.N.Y.10467
 OSUNY

This article is intended as an objective hardware review of SEB, the Super Expansion Board for your C1P. SEB is manufactured by Grafix, 911 Columbia Av. N. Bergen, N.J.

SEB is a graphics board that turns your C1P display into a hi-resolution pixel driven video display. Effective resolution is 256x192 individually programmable dots. This is nearly as good as ATARI(320x192), or APPLE(280x192). SEB uses a Motorola MC6847 video display generator chip (the same one as the APF Imagination Machine). Some of the hardware features are as follows:

- 1) True hi-res in 12 modes
- 2) Directly supports up to 16K of user RAM
- 3) 8 colors (less in the higher res modes)
- 4) 6K of video RAM needed for hi-res (SEB provides 5K)
- 5) On board RF modulator
- 6) No traces to cut. SEB is a board; it is not a hardware mod
- 7) The video output is isolated from your C1 display; you can drive two videos if you have two monitors. Imagine all those neat games that you can do!

As far as the installation of SEB is concerned I am not a hardware person, I had a friend put it in for me. There really are no traces to cut. However the polarities are reversed on the power supply connector, no big deal to fix but annoying until you notice it. SEB fits nicely into your C1 case. My SEB runs on one power supply, you may need another one if you don't have low power memory chips.

I've had SEB for nearly a month now. I've been working on software for SEB graphics for a while now. Objectively I must say that it is quite difficult to do hi-res software. The resolution is very nice but the effort is quite great as compared to normal C1 graphics. If you are impressed by graphics on the ATARI, APPLE, SORCERER, etc... then you will like SEB.

SEB has its own character generator. However characters cannot be mixed with graphics. In the character mode you get 32 characters by 16 lines. In the hi-res mode you can theoretically create your own characters but this must be done by defining the bits for each character. The character mode is good for text. A friend and I have written a fairly simple machine language routine that will drive BASIC display on SEB. Thus I can run hi-res graphics programs and when a system error message comes up the display will switch to character mode.

In summary, I think that SEB is a good way to upgrade your C1. Even if you don't think that graphics is that important, the expansion features (memory, etc..) make the SEB a nice piece of hardware. However, if you are not that good at programming then you will have some trouble working with the documentation, which is hardware oriented. You should be familiar with machine language if you want to greatly benefit from the hi-res graphics which is highly bit oriented. All in all SEB provides a great challenge to he who wants to design the ultimate graphics programs. Good luck!

SOURCE UPDATE: CEGMON

CEGMON is the name of a new monitor chip put out by a British group. Cegmon extends and improves upon the old SYNMON chip, which is supplied with your original computer. Among its features is the elimination of OSI's cursed cursor backspace with a true backspace/delete. This alone will probably prompt some of you to plunk down the \$60 for the chip; but wait: there's much, much more!

CEGMON IN BASIC:

The keyboard under CEGMON is now a "real" typewriter. This means that the only adventure you can play on same is one purchased from a software firm, (try Marooned in Space or Ghost Town by Orion. Trust me.) not the one supplied by OSI. Rather than guessing which shift key does what, when and where, the keyboard is now "normal". Shift P is not a line delete anymore, but shift zero is. (This is a mistake. I always hit that thing when I'm trying for a right parenthese;) Shift-O with the shift lock down is still a backspace. The rubout key is now decoded as a backspace/delete; the one the CEGMON writers advise that you use, because the shift-O only works with the shift lock down. With shift lock up, you will get a capital-O when you hit shift-O.

The so called "screen handler" uses part of page two to define a software "screen" on the true, physical screen. With their new screen handler, you can define windows, bottom of the screen, etc. In reference to the screen handler, several new "commands" are supported. Some of these are:

```
PRINT CHR$(10).....moves cursor down one line (cursor is now free).
PRINT CHR$(11).....moves cursor right one space.
PRINT CHR$(12).....homes cursor to top left of screen.
PRINT CHR$(13).....CR-moves cursor to beginning of same line.
PRINT CHR$(26).....clears entire screen.
PRINT CHR$(30).....clears the defined screen.
```

CHR\$(30) will clear the software defined screen. This leaves the surrounding screen memory uncleared. CHR\$(26) will clear the physical screen, from \$D000 to \$D400. (For C1P.)

CEGMON documentation states that you can access these commands by holding down a control-something. Don't believe it. Those same control keys are masked out by BASIC. They do, however, work perfectly fine in the machine language monitor.

Best of all is a screen editor. With it, you can edit in mid-line. While not like the PET or ATARI screen editors, this system is workable and convenient, and has peculiar strengths of its' own. Pressing control-E will turn on a block at your current cursor position. Using other control keys, you can maneuver this block about the screen, to the point at which you want to copy from the screen into memory. When you press escape, the letter beneath the block is copied into the BASIC line buffer, and the current cursor location; this echo effect keeps you updated. From there, it is treated

as if you had entered it from the keyboard. There you can leave it, or erase it with a backspace. You can also keep moving the special block/edit cursor about, to copy different section of a program or line. With this, you can combine several lines into one by copying into one BASIC line. As an added bonus, OSI's assembler/editor will also possess real backspace and line editing, since it also uses the monitor ROM. (The extended monitor also, but I haven't had a chance to try that out yet.)

CEGMON IN MACHINE:

After working with OSI's six digit display in the machine language monitor, CEGMON is a true pleasure to use.

CEGMON retains the OSI ML Monitor command format, but little else is the same. Entry into data mode is still with a slash, exit with a period, and incrementing the memory location in data mode with a return. In addition, you will have:

- 1) Scrolling
- 2) Memory Move
- 3) Monitor format save and autostart
- 4) Load (with space bar used to exit the mode!)
- 5) Tabular display (hoorah!)
- 6) Breakpoints and break table, restart.
- 7) unlisted others.

These are the major commands. The general ease of use is enough to set you drooling with pleasure. If you are a machine language programmer, get this at once. It is more valuable in machine language than in BASIC.

Memory move is a simple move. It does not recompute JMP's and JSR's for you. Tabular display is an eight byte per line hex display of specified memory locations. The cursor and screen will continue to act in CEGMON BASIC fashion, that is, the cursor is free over the screen, not fixed at the bottom line. However, you do retain the option of no/scroll in monitor data mode; this was retained to retain speed in loading in from tape. (The screen doesn't have to scroll) A line-feed in data mode will scroll down, opening a new memory location, and keeping the old one and it's contents still on the screen.

RANDOM THOUGHTS AND WARNINGS:

CEGMON is great for me, but perhaps not for you. I have an 8K C1P, and already have some problems with software compatibility. CEGMON does use various bits of page zero and page two (the use of page two is only to keep tabs on the new defined screen. POKE 538,149 will bring back the -gack- SYNMON monitor, along with attending fake backspace and fixed cursor. But it will free page two from \$0222 to \$02FA.) Any heavily based ROM program will probably have a bit of difficulty in operating, neither will a program which uses location 538 or various bits of page

zero. For example: WP6502 will not load in (tape version); Aardvark's Maze program will half-work. Earthship's Lunar Lander will not work at all. (It think it uses location 538 for something.) HEXDOS for the C1 with disk will not work. However, some compatability has been maintained, but by no means as high as you would wish.

AHEM. **ORION software runs.** Aardvark's cursed cursor control burns. \$FEED still returns a character at location 531. The Cegmon manual tells you how to construct a non-halting GET statement, how to construct a BASIC trace routine, how to simulate PRINT AT.

Documentation is professional.

I've only run into one minor bug in CEGMON, in the Machine language monitor (BASIC PROGRAMMERS heave a sigh of relief); When you are typing in data mode-ASCII mode, the backspace will not work correctly. To see the bug, enter this mode, then type directly to screen memory. You will see the ASCII characters being directly entered into memory, but if you try to backspace, you will see that it will be entered in the next free or consecutive memory location as the old BASIC underscore symbol, an ASCII 96. (96? I'm working on this at 12:30 AM, the night after a double test, so don't expect me to be perfect. If I'm right, so much the better.)

Hopefully, I have gien you an overview of what CEGMON is and what it does. I feel that it is well worth the \$60 price tag, even though it does have a few drawbacks. (Someone is trying to construct a CEGMON/SYNNON switch, so this too, may come to pass.)

Installation for me was fairly painless; a cut trace, and one soldered jumper (DO NOT LET MICHAEL BASSMAN CUT OR SOLDER YOUR BOARD.) As I understand it, installation on the C4 is a bit more complicated - something to do with extra circuitry with the ROM decoding.

Keep monitoring.

THOMAS CHENG
26 MADISON STREET
APARTMENT 4I
NEW YORK, NEW YORK 10036

A BASICALLY USEFUL BASIC UTILITY

--Shel sacks

I've noticed that there are plenty of routines that would be extremely useful, were it not for the fact that writing the routine would take more time than the actual program might. So we often do it the hard way, each and every time; we'll take care of that utility 'later'. (And rarely do.)

So-how about swapping (thru OSItems) those unique, hardly-ever-useful routines so we don't each get involved in re-inventing the wheel'.

I'll start the ball rolling with a 'hardly-ever-useful' routine that columnizes numeric data, either from a pre-existing array, or from the keyboard. This routine will Right-Justify all integer arrays (like you do when adding); or it will automatically detect decimal points and will line those up (again, like in addition). It will either make all the columns of equal width; or you can determine the column alignment. Line spacing is also variable. One last point--There's hardly any use of strings, so "String-bug" hasn't reared it's ugly head for me (yet??)

```

5  REM-All variables are in FULLCAP
10  REM-Columnizer, copyright 1981,
    Sheldon Sacks. (516)681-2388
30  ForZ=1to50:?:Next
35  Input"#Cols";COLS:Input"Rows";ROWS
    :ELEMENTS=COLS*ROWS:Dim Loc(ELEMENTS)
40  Input"Terminal width";TW:Input"
    Line spacing";SP
50  Input"Auto-cols";A$:IfA$="N"then
    gosub600:goto60
55  Input"Top left,right";LEFT,RIGHT
60  Input"Cursor home location";CL

```

```

100 ?"there are 3 ways to enter the
    data:":?
105 ?"1-already in array named 'DIS
    PLAY' in correct order(LtoR,LFeed)"
110 ?"2-data in array named((YOUR AR
    RAY)) in correct order"
115 ?"3-data to be entered from key
    board in correct order."
120 Input"which";D:IfD=2then400:
125 IfD=3thenDimDISPLAY(ELEMENTS):
    Goto500
130 IfD<1OrD>3then100
200 Rem-display routine:IfA$="N"then
    230
205 LW=RIGHT-LEFT:GW=INT(LW/COLS):
    LOC(1)=LEFT+2
210ForZ=2toELEMENTS
215 If(Z-1)/COLS=INT((Z-1)/COLS)then
    LOC(Z)=LOC(Z-COLS)+SP*TW Goto225
220 LOC(Z)=LOC(Z-1)+GW *TW:
225 Next
230 ForZ=1toELEMENTS:IfDISPLAY(Z)<>
    Int(DISPLAY(Z))Then235:Next:Goto
    350
235 ForZ=1toELEMENTS
240 LOC(Z)=LOC(Z)-Len(Str$(Int(DIS
    PLAY(Z))))
245 Next:Goto350

```

```

350 ForZ=1to50:?:Next:ForZ=1toELEMENTS:
    IS=Str$(DISPLAY(Z))
355 ForZZ=0toLen(IS)-1:PokeLOC(Z)+ZZ,
    Asc(Mid$(IS,ZZ+1,1))
360NextZZ:NextZ:PokeCL,32:ForI=1to
    999999:Next:Rem-maintain screen
400 ?"Jot down for future ref
    erence:"
405 ?"You need to swap your array into
    'DISPLAY(Z)'"
410 ?"so at the end of your program,
    add this statement:"
415 ?"XXX DimDISPLAY(#elements in
    your array):ForZ=1toELEMENTS";
420 ?"DISPLAY(Z)=<your array name>(Z):
    NEXT:Goto5"
425 Stop:Rem-after you have the state-
    ment down, you can just enter"CONT"
430 Goto200
500 Rem-keyboard input routine:?"Enter
    data--":?:?
505 ForZ=1toELEMENTS:Input DISPLAY(Z):
    Next:Goto200
600 ?"Enter start points/decimal
    locs.for 1st row":?
610 ForZ=1toCOLS:InputLOC(Z):Next
615 ForZ=COLS+1toELEMENTS:LOC(Z)=LOC(Z
    COLS)+SP*TW:Next
620 Return
625 END
1000Rem-end of program; what follows is
    an example of a pre-existing array swap
1005DimROOT(20):ForX=1to20:ROOT(X)=
    Sqr(X):NextX
1010Rem-array is 'ROOT(X)'. Now swap as
    per instructs in subr 400:
1015DimDISPLAY(20):ForZ=1to20:DISPLAY
    (Z)=ROOT(Z):Next:Goto5

```

(TO RUN ENTIRE PROGRAM, ENTER "RUN 1000"
AND BE SURE TO START THE DATA-GENERATING
PROGRAM AFTER LINE #625.)

MASTERMIND

by Peter Schreiber

(this game is self-explanatory and will not run on the 1P

```

100 Goto430
110 ForY=1toLen(D$):PokeY+D,Asc(mid$
(D$,Y,1)):NextY:Return
120 Input"Number of colors you want
to play with(6-9)";CL
130 D=53529:Poke11,34:Poke12,2:IfCL
<6orCL>9then120
140 ForX=546to570:ReadM:PokeX,M:Next:
ForX=1to4:ReadA(X):NextX
150 DimA$(CL):X=Usr(X):ReadD$:Gosub
110:ForD=53771to53810step4
160 N=N+1:D$=Str$(N):Gosub110:Next:
ForA=53835to53876:PokeA,159
170 PokeA+320,148:PokeA+896,163:Next:
ForA=53899to54667step64
180 PokeA,161:PokeA+41,161:P=149:IfA
=54155thenP=219
190 ForX=A+4toA+36step4:PokeX,P:Next
X,A:D=54859:F=1
200 ForX=1toCL:ReadA$(X),D$:Gosub110:
IfX/3+Int(Y/3)thenD=D+16
210 D=D+16:Next:Poke11,0:Poke12,253:
NO=49:G=53965:L1=54221
220 ForX=1to4:R=Int(Rnd(1)*CL+1):C(X)
=R:C2(X)=R:NextX
230 PokeL1,NO:PokeL1+1,32:X=Usr(X):P=
Peek(531)-48
240IfP=0andNO<>49thenPokeL1,32:L1=L1-
128:NO=NO-1
250 IfP<1orP>CLthen230
260 D$=A$(P):D=L1-1:Gosub110:C1(NO-48)
=P:L1=L1+128:NO=NO+1
270ForC=1to350:NextC:IfNO<>53then230
280 B=0:ForV=1to4:IfC(V)=C1(V)thenB=
B+1:C(V)=0:C1(V)=0
290 NextV:IfBthenforX=1toB:PokeG+A(X),
66:NextX:IfB=4then350
300 F=F+1:NO=49:W=0orV=1to4:ForQ=1to4
310 IfC(V)thenIfC(V)=C1(Q)thenW=W+1:
C(V)=0:C1(Q)=0
320 NextQ,V:IfWthenForX=4to5-Wstep-1:
PokeG+A(X),87:NextX
330 G=G+4:L1=L1-508:ForX=1to4:C(X)=
C2(X):NextX:IfF<>11then230
340 Print"The computer code was:"C2(1)
C2(2)C2(3)C2(4);:Goto360
350 Print"Congratulations!! You broke
the code in"F"turns!";
360 ForX=1to8000:NextX:Poke11,34:Poke
12,2:X=Usr(X)
370 Print:Input"Again";A$:X=USR(X):
IfLeft$(A$,1)="Y"thenRun120
380 END
390 DATA162,0,134,254,169,208,133,255,
160,0,169,32,145,254,200
400 DATA208,251,230,255,232,224,8,208,
244,96,0,1,64,65
410 DATAMASTERMIND,BL,1-BLUE,BK,2-BLACK,
BR,3-BROWN,GD,4-GOLD
420 DATAGR,5-GREEN,OG,6-ORANGE,RD,7-RED,
WH,8-WHITE,YE,9-YELLOW
440 Print" This is the game of MASTER
MIND."
430 ForX=1to8:Print:NextX
450 Print:Print
460 Print" The object of the
game is to guess the"
470 Print" color code that the
computer has randomly"
480 Print" picked within your nine
allotted colors."
490 Print
500 Print" You will be given
a choice of the number"
510 Print" of colors you want to play
with. The computer"
520 Print" then picks a code from
that number of colors."
530 Print
540 Print" During the game, when
the computer awards"
550 Print" you a black peg, you
have correctly calculated"
560 Print" the position of that color
on the board. When"
570 Print" the computer awards you
a white peg, you have"
580 Print" correctly calculated that
color, but NOT its"
590 Print" position on the board----
600 Print
610 Print" The color / key code
will be displayed at"
620 Print" the bottom of the screen
for reference. The"
630 Print" 0-key can be used to back
up and reenter."
640 Print:Print:Print:Goto120
-----
(---looks like a good game. any
volunteers to convert it for
the 1P, using symbols instead
of colors??? -ed.)

```


EXTENSIONS FOR OS-650 BASIC

by Dan Schwartz

This program provides four extensions to the standard version of OS-650 U3.X Basic.

1) The command NEW (number of bytes). This command eliminates the need for the CHANGE program by allowing you to change the lower limit of the workspace with a single command. For example, NEW 2048 for a 5-inch disk system or NEW 3072 for 8-inch will reserve room for one disk buffer. NEW 0 will return to a normal workspace. One difference between my implementation and that found in OS-650 is that in this version, the command NEW by itself does not change the workspace limits, but just clears the current workspace as usual. Another is that general expressions are allowed after the word NEW, for example NEW 12*256.

2) RESTORE (line number). Restores the data pointer to the beginning of the specified line. The line number can be any expression, but must evaluate to a line actually present in the program to avoid an error message. The line need not actually be a DATA statement. If it isn't, the next data item will be read from the first DATA line after the specified line.

3) CALL (subroutine address). This one has been done before (see "Machine Language Call" by Mike Cohen in OSI-items Vol. II, No. 8 (Aug. 1980)). The only difference between my implementation and Mike's is that I combined the two successive subroutine calls needed to evaluate the address into a single subroutine, so that it can be called from other places.

4) GOTO or GOSUB (expression). Once that subroutine is in place, other routines can call it. By replacing a call to a numeric-only evaluator in the code for GOTO with a call to the new evaluator (within the CALL code), the GOTO statement becomes able to GOTO any expression, for example GOTO 10*X. The GOSUB statement is automatically extended when the GOTO statement is.

Note: the code for NEW XXX as given here also uses this evaluator, so it cannot be used unless CALL is enabled. The code for RESTORE XXX uses it only by way of the GOTO code, so it will work with or without the evaluator in place, but will only be able to use explicit line numbers if it is not.

```

10 REM  EXTENSION PROGRAM FOR OS-650 BASIC
20 REM  DANIEL SCHWARTZ  75 E 190 ST  BRONX NY 10468
30 PRINT
40 PRINT"ENABLING 'NEW XXX', 'RESTORE XXX', CALL AND COMPUTED GOTO/GOSUB"
50 PRINT
60 CC=PEEK(2073):POKE 2073,96:REM  LOCK SYSTEM WHILE CHANGING
70 REM
80 REM  POKE IN CODE FOR 'CALL', WITH EVALUATOR AS SUBROUTINE
90 FOR X=0 TO 11:READ C:POKE 2157+X,C:NEXT
100 REM  POKE IN KEYWORD LETTERS 'C' AND 'A' FOR 'CALL'
110 POKE 709,67:POKE 710,65
120 REM
130 REM  POINT GOTO CODE TO USE NEW EVALUATOR
140 POKE 2215,115:POKE 2216,8
150 REM
160 REM  RESERVE 104 BYTES IN HIGH MEMORY
170 M=PEEK(132)+256*PEEK(133)
180 M=M-104:MH=INT(M/256):ML=M-256*MH
190 POKE 132,ML:POKE 133,MH:POKE 128,ML:POKE 129,MH
200 REM
210 REM  POKE IN CODE FOR 'NEW XXX' AND 'RESTORE XXX'
220 FOR X=0 TO 103:READ C:POKE M+X,C:NEXT
230 REM
240 REM  SET UP NEW DISPATCH ADDRESS FOR 'NEW'
250 ND=M-1:NH=INT(ND/256):NL=ND-256*NH
260 POKE 566,NL:POKE 567,NH
270 REM
280 REM  SET UP NEW DISPATCH ADDRESS FOR 'RESTORE'
290 RD=M+55:RH=INT(RD/256):RL=RD-256*RH
300 POKE 534,RL:POKE 535,RH
310 REM
320 POKE 2073,CC:REM  RESTORE PREVIOUS CONTROL-C STATUS
330 REM
340 PRINT M-12670"BYTES FREE"
345 REM  FOR MINI-FLOPPY USE M-12926 FOR BYTES FREE
350 REM
360 REM  OBJECT CODE FOR 'CALL', WITH EVALUATOR SUBROUTINE
370 DATA 32,115,8,108,25,0,32,185,12,76,114,22
375 REM
380 REM  OBJECT CODE FOR 'NEW XXX'
390 REM  FOR MINI-FLOPPY CHANGE FIFTEENTH DATA ITEM IN LINE 500
400 REM  FROM '49' TO '50'!!
500 DATA 240,50,32,115,8,165,25,24,105,128,170,165,26,105,49,168,138
510 DATA 196,133,208,2,197,132,144,3,76,76,4,233,0,176,2,136,56,133
520 DATA 120,132,121,233,1,176,1,136,133,25,132,26,160,0,152,145,25,24
530 DATA 76,98,6
535 REM
540 REM  OBJECT CODE FOR 'RESTORE XXX'
550 DATA 208,3,76,10,8,170,165,199,72,165,200,72,165,134,72,165,135,72
560 DATA 138,32,166,8,165,199,133,142,165,200,133,143,104,133,135,104
570 DATA 133,134,104,133,200,104,133,199,32,192,0,208,251,96

```


Handshake (CTS) of Quick Printer II DOES work with C1P
by Klaus Ernst

Attracted by the low price and encouraged by a favorable review in AARDVARK JOURNAL vol.1, no.1 p.7 I bought a Radio Shack QUICK PRINTER II.

I had already populated the RS-232 OUTPUT on my 600 board, so all I had to do was, to double the BAUD rate. I studied Aardvark's "600 BAUD CASSETTE +PRINTER CONVERSION" instructions and decided all this messing with capacitors etc. was not for me. So I did the second version on the update sheet instead. Next I poked the values listed on page 8 of AARDVARK JOURNAL vol.1 no.1. The printer worked fine within the limitations described in the Aardvark article. Instead of a handshake (CTS=clear to send) the printer is slowed down by software to avoid the loss of characters.

Then I came across a Quick Printer II Fix by our own Danny Schwartz (yeah Danny!) in AARDVARK JOURNAL vol.1 no.3 p.5. With this loaded up the handshake should work. First I had to hook up the CTS line. The schematic of the printer shows that CTS swings between +5 VDC and -6.2VDC. I did not want to hook up a negative voltage directly to a TTL chip, instead I populated the RS232 INPUT (Q2 et al.), fed CTS thru there (where it also gets inverted) and connected it to CTS (pin 24) of the ACIA (6850). Pin 24 had to be disconnected from GND first (cut trace at W 5). I installed a switch to be able to *CTS either to GND or the CTS line. I loaded the fix, typed SAVE and a PRINT command, hit RETURN and -- the system locked up!

I checked with a couple of knowledgeable people, who said: "Forget it, the handshake won't work!"

Then at the last OSI meeting I checked with Danny, who told me that there is a typo in Aardvark's listing. Line OODA should read OODA 8D1A02 (not 801A02) :

Well, I could not wait to try it out. I loaded the corrected fix -- and it worked!

I wrote a BASIC loader for it, which will selfdestruct after you run it (220 NEW). The fix suppresses NULLs that confuse the printer. Never use CHR\$(0) (NUL) more than once (in a row) or your machine will lock up, because the CTS line will stay low.

* CONNECT

LIST

5 REM QUICK PRINTER II FIX BY D
ANIEL SCHWARTZ

6 REM BASIC LOADER BY KLAUS ERN
ST

10 FOR I=54670560

20 READ A

30 POKE I, A

40 NEXT I

50 FOR J=21670220

60 READ A

70 POKE J, A

80 NEXT J

90 POKE 0, 76

100 POKE 1, 216

110 POKE 2, 80

200 DATA 32, 45, 191, 72, 173, 5, 2, 24

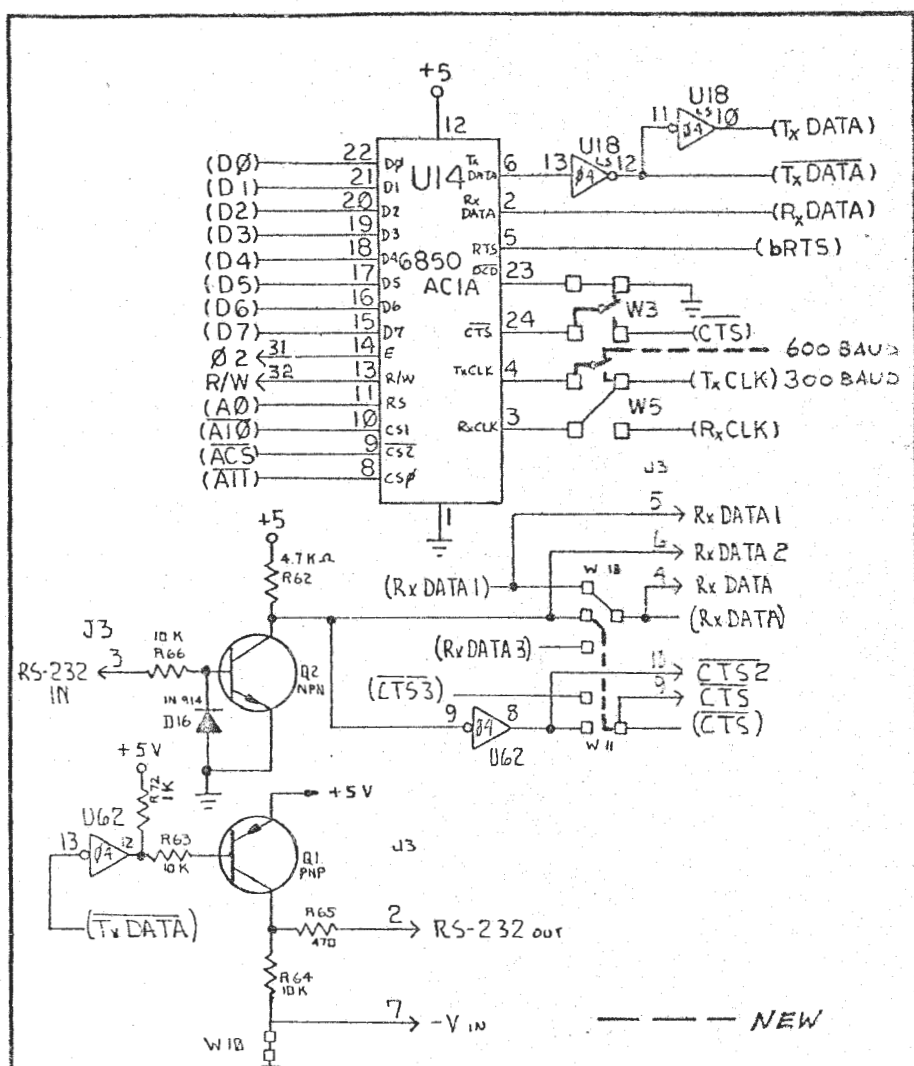
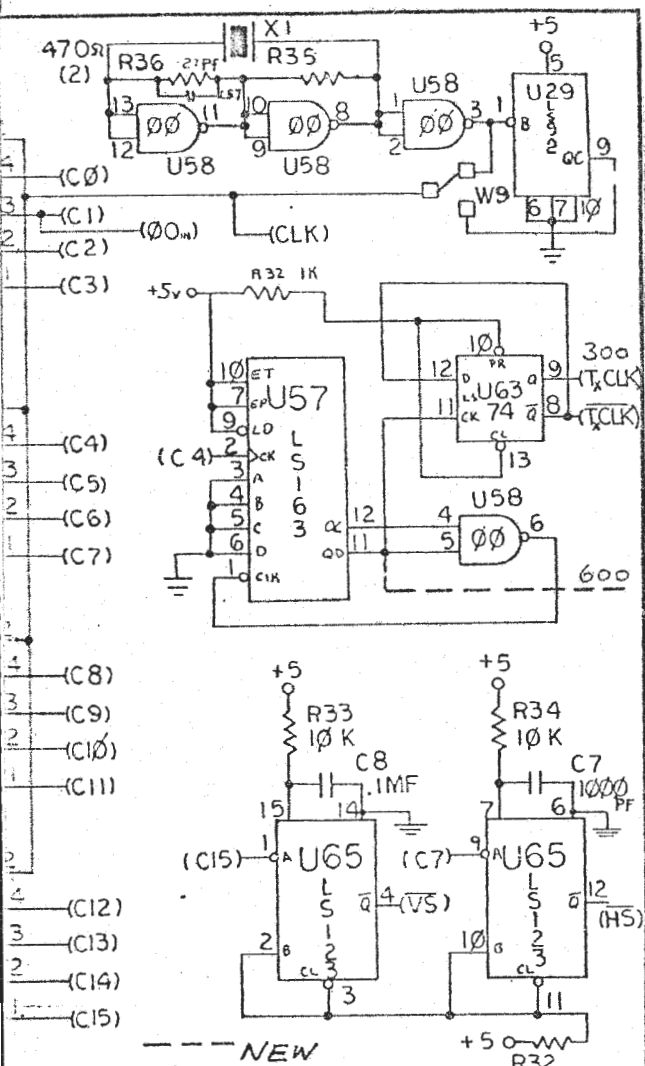
0, 4, 104, 76, 177, 252, 104, 96

210 DATA 169, 34, 141, 26, 2, 169, 2, 1

41, 27, 2, 76, 116, 162

220 NEW

OK



TIFIC		product name/number	
page	status	sheet 9 of 13	

OHIO SCIENTIFIC			product name/number	
date 5 SEP 1978	revision B	page	status	sheet 6 of 13

NOTE 1: THIS IS WHERE THE WIRES GO, PUT SWITCH IN CONVENIENT LOCATION.

HARDWARE MODS FOR 600 BOARD FOR USE OF R/S
HANDSHAKE SWITCH: QUICK PRINTER II

REMOVE: CUT JUMPER BETWEEN GND PAD AND CTS PAD (PIN 24)
AT W3

INSTALL: 1) SPDT SWITCH AT W3 (NOTE 1)

2) POPULATE RS-232 IN

3) ADD JUMPER FROM R_x DATA 2 PAD OF W10
TO CTS PAD OF W11

4) CONNECT CTS LINE OF PRINTER TO RS-232 IN

BAUD RATE SWITCH:

REMOVE: CUT JUMPER BETWEEN T_x CLK PAD AND PAD
CONNECTED TO PIN 4 (T_x CLK)

INSTALL 1) SPDT SWITCH AT W5 (NOTE 1)

2) RUN NEW WIRE FROM ONE SIDE OF SWITCH TO PIN 11 OF U63. (600 BAUD LINE)

SHAPE PLOTTER - Version 2

by Mike Cohen

Here is an update on Sol's shape plotter from November 1980 OSItems. For this one, I have added multiple shape tables and adapted it to a disk system.

Many disk users wonder where to put machine language routines, since page 2 isn't free. Often, the solution is to put it in high memory, but how many times have you forgotten to set top of memory and had your machine code wiped out by strings???

Here's the solution - put the machine code under the basic workspace. Not only doesn't the memory size have to be set, but the machine code rides in and out with the basic program so there's no DATA statements and POKES to set it up.

To set up a program this way, it must be assembled to reside at \$327E, the start of BASIC's workspace. The program must end with 3 bytes of zero and the locations 3279 and 327B must both point to the address of the second zero byte (low byte first as usual). In \$327D you should put the number of tracks the machine language section would take up (remember, on a 5" disk each track is 8 pages).

After the program is properly set up, it should be saved on disk using a PUT command. Then, boot up BASIC and call in the machine language program that was saved and type NEW.

The address that you put in \$3279 and \$327b become BASIC's new start of workspace and the zero byte at that location allows basic to use it. If that first byte isn't zero, "NEW" will give a syntax error and basic may do other strange things. Next, type the BASIC program on top of the machine code and they may be saved on the disk as one file.

Now, here's the program. The machine code program actually ends at \$32BD. Everything after that is BASIC text that was 'disassembled' by the extended monitor.

The BASIC program is similar to the train crash program which appeared in COMPUTE II several months ago. The machine language routine will work on all systems, but the BASIC demo is for C1 only. To use the routine, poke the address of the shape table into 11899 and 11900, low byte first, and the screen address into 11897 and 11898. Then call it with X=USR(X), as it is done in the subroutines at 1000 and 2000, which select alternate shapes.

```

5 TIME=150
7 FORM=1T030:PRINT:NEXT:POKE54117,32
10 POKE8955,126:POKE8956,50
15 FORM=53954T053960:POKEM,128:POKEM+32,209:NEXT
17 FORM=53922T053948STEP3:POKEM,INT(RND(3)*3+13):NEXTM
20 DATA 2,2,167,1,157,1,161,3,167,24
30 DATA 165,1,161,1,161,1,161,1,161,1,155,1,176,1,161,1,161,24
40 DATA 166,1,161,1,161,1,161,1,161,1,128,1,161,1,161,1,161,24
50 DATA 176,1,224,1,225,1,226,3,226,2,226,0
60 DATA 165,3,161,1,156,1,165,2,2,25
70 DATA 161,1,161,1,178,1,155,1,161,1,161,1,161,1,161,1,167,24
80 DATA 161,1,161,1,161,1,128,1,161,1,161,1,161,1,161,1,168,24
90 DATA 226,2,226,3,226,1,224,1,225,1,178,0
100 AD=11897:PT=11899
110 P=11901:P1=P
120 DEFFNH(X)=INT(X/256):DEFFNL(X)=X-FNH(X)*256
130 READ Q:POKE P,Q:P=P+1:IF Q>0 THEN 130
140 P2=P
150 READ Q:POKE P,Q:P=P+1:IF Q>0 THEN 150
160 S1=53910:S2=53891
170 GOSUB1000:GOSUB2000
175 FORT=1T0TIME:NEXTT
180 GOSUB1000:GOSUB2000
190 S1=S1-1:S2=S2+1:IFS1-S2>9THEN170
195 GOSUB1000:GOSUB2000
200 FORM=1T06:POKES1+M,42:POKES2+M,42:POKES1-M,42:POKES2-M,42
210 POKES1+32*M,42:POKES2+32*M,42:POKES1-32*M,42:POKES2-32*M,42:NEXTM
220 FORM=1T06:POKES1+M,32:POKES2+M,32:POKES1-M,32:POKES2-M,32
240 POKES1+32*M,32:POKES2+32*M,32:POKES1-32*M,32:POKES2-32*M,32:NEXTM
300 RUN
999 END
1000 POKEAD,FNL(S1):POKEAD+1,FNH(S1):POKEPT,FNL(P1):POKEPT+1,FNH(P1)
1010 X=USR(X):RETURN
2000 POKEAD,FNL(S2):POKEAD+1,FNH(S2):POKEPT,FNL(P2):POKEPT+1,FNH(P2)
2010 X=USR(X):RETURN

```

:0327E

```

327E AD7B2E LDA $2E7B
3281 85FC STA $FC
3283 AD7C2E LDA $2E7C
3286 85FD STA $FD
3288 AD792E LDA $2E79
328B 85FE STA $FE
328D AD7A2E LDA $2E7A
3290 85FF STA $FF
3292 A000 LDY #$00
3294 B1FE LDA ($FE),Y
3296 AA TAX
3297 B1FC LDA ($FC),Y
3299 91FE STA ($FE),Y
329B BA TXA
329C 91FC STA ($FC),Y
329E E6FC INC $FC
32A0 D002 BNE $32A4
32A2 E6FD INC $FD
32A4 B1FC LDA ($FC),Y
32A6 D001 BNE $32A9

```


32A6 D001 BNE \$32A9--

32A8 60	RTS
32A9 18	CLC
32AA 65FE	ADC \$FE
32AC 85FE	STA \$FE
32AE A5FF	LDA \$FF
32B0 6900	ADC #\$00
32B2 85FF	STA \$FF
32B4 E6FC	INC \$FC
32B6 D0DC	BNE \$3294
32B8 E6FD	INC \$FD
32BA 4C9432	JMP \$3294
32BD 00	BRK
32BE CB	???
32BF 32	???
32C0 0500	ORA \$00
32C2 54	???
32C3 494D	EOR #\$4D
32C5 45AB	EOR \$AB
32C7 3135	AND (\$35),Y
32C9 3000	BMI \$32CB
32CB E532	SBC \$32
32CD 07	???
32CE 00	BRK
32CF 814D	STA (\$4D,X)
32D1 AB	???
32D2 319D	AND (\$9D),Y

256 Byte Precision Factorials

Very much has been said about the lack of precision (8 bits) of the 6502. All of the complaints fail to take into account the provisions for extended precision arithmetic that are inherent in the chip. For example, the usage of the x and y registers in the indexed addressing modes can make the manipulation of large numbers very easy, by sequentially accessing the portions of the numbers, and saving the status of the registers onto the stack.

The following program illustrates the concept of handling very large numbers, in this case, 255 factorial. The program will take the number stored at 00h, then compute and leave it's factorial beginning at \$1D00. This number will extend upwards in memory, in typical 6502 16-byte, hi-byte format. (Actually, it will be lowest-byte, lower-byte, higher-byte, all the way up to the highest byte.)

The multiplication routine has been pulled from Lance Leventhal's book, programming with the 6502, after having been liberally modified to work in higher precision than originally designed for.

THE PROGRAM

Fact is the location holding the number you want the factorial of. Fact1, Fact2, are temporary location used within the body of the program. Temp, inaptly named, is the current precision of the factorial, i.e., \$1D00 + Temp = last byte of factorial (highest byte). Tans is the temporary store for the factorial.* (Answer will be held here Ans is the permanent store for the factorial.* temporarily.)

Lines 90-280 are mainly used for setting up the correct requirements for the program to run. The areas of Tans and Ans are zeroed to prevent interference. The stack is initialized.

Mult is the main routine, which controls the rest of the program, calling upon specific subroutines to perform the extended precision instructions. What I mean by this is that the subroutines will rotate, move, etc., the chunks of memory holding the numbers. For example, subroutine rotate will do a ROL of 256 bytes, as if one command had been used to shift over an entire block of memory. In this fashion, the blocks of memory holding the answer are maneuvered as if they were a single byte. Therefore, nothing will be lost between byte boundaries, such as a carry.

I have purposely not been specific in my documentation, as I believe that studying the program will teach more than one of my lectures. The program written is very limited in usefulness, however, it serves its purpose of illustrating hi-precision arithmetic very well. I do hope that it will be used as an example.

.P

```

10 0000          ; ANSWER LEFT IN $1D00, PRECISION
20 000A          *=10
30 000A          FACT=0
40 000A          FACT1=01
50 000A          FACT2=02
60 000A          TEMP=03
70 000A          ANS=$1D00
80 000A          TANS=$1E00
90 000A A2FF     LDX #$FF
100 000C 9A      TXS
110 000D 201400  JSR INIT
120 0010 203200  JSR MULT
130 0013 00      BRK
140 0014 A200    INIT  LDX #0
150 0016 A900    LDA #0
160 0018 9D001D  STORE STA ANS,X
170 001B 9D001E  STA TANS,X
180 001E E8      INX
190 001F D0F7    BNE STORE
200 0021 A500    LDA FACT
210 0023 8D001D  STA ANS
220 0026 38      SEC
230 0027 E901    SBC #$01
240 0029 8501    STA FACT1
250 002B 8502    STA FACT2
260 002D A900    LDA #00
270 002F 8503    STA TEMP
280 0031 60      RTS
290 0032         ;
300 0032         ;
310 0032         ;
320 0032 A008    MULT  LDY #$08
330 0034 207700  SHIFT JSR ROTATE
340 0037 A200    LDX #$00
350 0039 0602    ASL FACT2
360 003B 901E    BCC NOADD
370 003D 18      CLC
380 003E 08      PHP
390 003F 4C4300  JMP ADD

```

400 0042 E8	NEXT	INX
410 0043 28	ADD	PLP
420 0044 BD001E		LDA TANS,X
430 0047 7D001D		ADC ANS,X
440 004A 9D001E		STA TANS,X
450 004D 08		PHP
460 004E E403		CPX TEMP
470 0050 D0F0		BNE NEXT
480 0052 28		PLP
490 0053 9006		BCC NOADD
500 0055 E603		INC TEMP
510 0057 E8		INX
520 0058 FE001E		INC TANS,X
530 005B 88	NOADD	DEY
540 005C D0D6		BNE SHIFT
550 005E A200		LDX #0
560 0060 BD001E	TRANS	LDA TANS,X
570 0063 9D001D		STA ANS,X
580 0066 A900		LDA #0
590 0068 9D001E		STA TANS,X
600 006B E8		INX
610 006C D0F2		BNE TRANS
620 006E C601		DEC FACT1
630 0070 A501		LDA FACT1
640 0072 8502		STA FACT2
650 0074 D0BC		BNE MULT
660 0076 60		RTS
670 0077 A2FF	ROTATE	LDX #\$FF
680 0079 18		CLC
690 007A 08		PHP
700 007B E8	ENTER	INX
710 007C 28		PLP
720 007D 3E001E		ROL TANS,X
730 0080 08		PHP
740 0081 E403		CPX TEMP
750 0083 D0F6		BNE ENTER
760 0085 28		PLP
770 0086 9006		BCC RET
780 0088 E603		INC TEMP
790 008A E8		INX
800 008B FE001E		INC TANS,X
810 008E 60	RET	RTS

400 0042 E8	NEXT	INX
410 0043 28	ADD	PLP
420 0044 BD001E		LDA TANS,X
430 0047 7D001D		ADC -ANS,X
440 004A 9D001E		STA TANS,X
450 004D 08		PHP
460 004E E403		CPX TEMP
470 0050 D0F0		BNE NEXT
480 0052 28		PLP
490 0053 9006		BCC NOADD
500 0055 E603		INC TEMP
510 0057 E8		INX
520 0058 FE001E		INC TANS,X
530 005B 88	NOADD	DEY
540 005C D0D6		BNE SHIFT
550 005E A200		LDX #0
560 0060 BD001E	TRANS	LDA TANS,X
570 0063 9D001D		STA ANS,X
580 0066 A900		LDA #0
590 0068 9D001E		STA TANS,X
600 006B E8		INX
610 006C D0F2		BNE TRANS
620 006E C601		DEC FACT1
630 0070 A501		LDA FACT1
640 0072 8502		STA FACT2
650 0074 D0BC		BNE MULT
660 0076 60		RTS
670 0077 A2FF	ROTATE	LDX #\$FF
680 0079 18		CLC
690 007A 08		PHP
700 007B E8	ENTER	INX
710 007C 28		PLP
720 007D 3E001E		ROL TANS,X
730 0080 08		PHP
740 0081 E403		CPX TEMP
750 0083 D0F6		BNE ENTER
760 0085 28		PLP
770 0086 9006		BCC RET
780 0088 E603		INC TEMP
790 008A E8		INX
800 008B FE001E		INC TANS,X
810 008E 60	RET	RTS