PEEK[65]

The Unofficial OSI Journal

Inside This Month:

Column One

Hello! Remember me? Yes, it has been an extraordinarily long time between issues. The reasons are many, some of them are my fault, and some of them aren't. The bottom line is that despite all of the problems I've had this year, PEEK[65] remains alive and well. It is becoming clear that despite my best efforts, it is very difficult to publish on a monthly basis and I may have to finally give in and work on bi-monthly issues. For the record, I will be crediting everyone's subscription for missed publications. The address labels on this issue already reflect a two-month credit that I posted when I first tried to publish earlier this year. Beyond that, I will be adding more credits, but they will be offset to a certain extent by subsequent issues which will be over-sized as this one is.

Before I leave the topic of the state of PEEK[65], I do want to assure you that I intend to keep publishing as long as there are people out there who want to read PEEK[65] and are willing to support it. That support must come in two forms. First of all, there is financial support. PEEK[65] has traditionally depended on advertising to pay a large amount of the costs. However, there hasn't been any demand for commercial ad space in PEEK for some time and that means subscriptions and sales of software, books, and other items must pay for everything. So far, revenues are just about keeping up, but I do ask that you check your expiration date and if it's close, sign up again. For my part, I intend to work on expanding the scope of the magazine to make it a more viable advertising media.

Secondly, and every bit as important as the dollars, is the never-ending need for articles. Each and every one of you, whether you realize it or not, has something to contribute. Each of us has his own special way of using his system, and sharing that experience can bring useful information to the entire community of PEEK readers. Most of you are not techinically inclined, and from the letters I receive, most of you don't consider yourselves to be good programmers, but don't let that stop you from writing. The quality of a letter or article isn't so much the level of technical excellence as it is the fact that you have solved a problem. So, if you're capable of submitting an article with a program as sophisticated as this month's "TRACER", please do. But if you're just a weekend hacker, I'd still like to see that little utility you wrote. Another thing I don't see enough of is letters describing how to connect OSI's to different peripherals, and just simple descriptions of how people use their systems as a part of their businesses. Right now, I have enough material to take me through 1-1/2 more oversized issues. After that, the library is dry, so PLEASE contribute!

Okay, enough of that. On to more pleasant topics. This issue contains two block-buster articles. The first is an excellent utility from Carl Eidbo, the OS-65U Machine Code Emulator/Tracer, which allows you to single-step through machine code. If you want to get into the guts of OS-65U, this little ditty will do the job. Second, John Hepner has given us his system for recording student's gardes. It's an excellent example of simple software doing a much-needed chore. I've added a brief article on the technical side of the new 65816 and 65802 chips and how they can really add power to current systems. Finally, Jack Noble submits the saga of the little gremlin that almost got away.

Have a Happy Holiday Season, and thank you for your past and continuing support of PEEK[65]!

The 65816/65802: Down & Dirty

by Richard L. Trethewey

It's been a hard struggle, but I've finally gotten passingly familiar with the new 16-bit microprocessors, the 65816 and the 65802. A quick glance through the reference materials is enough to start your mouth watering, but I think it's about time we got down to some specifics around here. This article will describe some of the benefits of the new chips in some detail and will also discuss some of the issues facing us.

Let me begin by acknowledging the reference book I used in my work, "65816/65802 Assembly Language Programming" and "The Apple IIgs Technical Reference", both by Michael Fischer, published by Osborne/McGraw-Hill. Fischer is a well-known writer on Apple II subjects and he does a fine job of making a dry subject exciting and understandable. The books cost about $20.00 each and are worth that much for the tables and examples alone. The first book is based on using an Apple IIe with some popular assemblers for that system, but it is by no means Apple-specific. The second is, naturally, specific to the Apple IIgs, but it is worth reading in order to gain an appreciation for the Apple IIgs operating system, ProDOS-16.

The 65816 and 65802 (from now on, when I say 65816, you can assume I also mean the 65802 unless otherwise specified) are CMOS microprocessors that work with 16-bit data and use 24 bits for addressing. They both feature a 6502 emulation mode which lets them run all software for that CPU. Fortunately, they "wake up" in this emulation mode so that we can boot up with standard OSI software without a hitch. Both of the new chips contain a vastly expanded instruction set which makes programming them a joy for anyone frustrated by their predecessor. However, it is a bit inaccurate to describe the emulation mode as simply "emulation" as all of the native mode instructions still function in this mode, albeit with some natural limitations.

The difference between the 65816 and the 65802 is that the 65816 has the full 24-bits available for addressing. The 65802 is a pin-compatible replacement for the 6502 and is thus limited to 16 bits in the real world. However, this does mean you can simply replace your 6502 with the '802 and take advantage of some of the speed and power of the new chip with a minimum investment. I just bought a 65802 from Jameco Electronics for $19.95 (plus a couple dollars for shipping and tax) and am pleased to report no problems with it in my C8P-DF. The one caveat that comes to mind is the OSI Assembler. Rumor has it that the OSI Assembler used unimplimented opcodes of the 6502. If so, that program will not run on the 65802 or the 65816. I haven't tested this yet, but it's something to watch out for.

For those who want to really explore new ground, Paul Chidley of TOSIE fame, and David Livesay both offer 65816 CPU boards that can hold a lot of memory and do so at very reasonable prices. And of course, let's not forget the DevTech Inc. troops back in Denver who have the DB-2 system available.

Table 1 shows the instruction set of the new chips, with the new instructions marked with an asterisk. As you can see, one of the primary gains with the new chips is to fill in the gaps in the instruction set of the 6502, by adding in everything from the 65C02, and topping it all off with its own enhancements.

The Hardware

To begin, most of your assumptions about the 65816 are true. The Accumulator, X and Y registers, and the Stack Pointer are now a full 16-bits wide. Yes, that's right, you can move the stack anywhere you like when running in the native mode, making it easy to set up separate stacks for separate tasks. The Program Counter is still only 16-bits, but the 658xx family has an additional register called the Program Bank register which adds the upper 8-bits needed to obtain the full 24-bit address range.

The heart of changing between modes with the 65816 lies in the Processor Status Register. In the 65816 and 65802, the Processor Status register is (hold onto your hats, boys and girls) a 9-bit byte. I'm joking, of course, but only slightly. The Status Register is still only 8-bits wide, but the CPU stores an additional bit internally called the Emulation Bit ("E"). Since the value of this bit is crucial in both the native and emulation modes, the Carry Bit is used for double-duty. The instruction XCE (Exchange Carry and Emulation) swaps the current contents of the Carry and Emulation bits. By preceding this command with either CLC or SEC, you alter the mode the CPU is running in.

When in the native mode, bit 5 of the Processor Status byte is the Memory Select flag. When set (1), operations are performed on 8-bits of data. When clear (0), memory and Accumulator operations use 16-bits. If you look in Table 1, you'll see that there are no instructions that set or clear this flag. Instead, the instructions SEP and REP are used to alter the Processor Status Register as a whole.

Again when in the native mode, the Break bit ("B" or bit 4) has a new function and is referred to as the Index Register Select Bit ("X"). This bit affects the X and Y registers as the Memory Select bit works on the Accumulator, which is to say switching between 8 and 16-bits. When used in indexed addressing modes,

the X and Y registers can be used to work on a full 64K range of data instead of the single page (256 byte) range on the 6502.

In addition to the Program Bank Register, a second new register is available on the 65816 called the Direct Register. The Direct Register controls the use of the addressing mode called (who would have guessed?) the Direct Mode. Simply put, the direct mode
is a faster way of addressing memory in the first bank from $000000 to $00FFFF, and is similar to the Zero Page mode on the 6502. The Direct Register is 16-bits wide and consists of two 8-bit halves, the DH for the high 8-bits and the DL for the low 8-bits. The direct mode operates differently when in the emulation mode than when in the native mode, and we'll discuss this more later.

The Accumulator takes on a special characteristic in the native mode. It can be used as a single 16-bit register (often called the "C" register), or as two distinct 8-bit registers called "A" and "B". But, while the "B" register can be exchanged with the "A" register to temporarily store data, the value in the "B" register cannot be changed directly without involving the "A" register. For now, don't be too concerned with "A", "B", and "C". Understanding that they exist is enough to get started.

Addressing Modes

As noted above, the 65816 also brings with it several extensions to the addressing mode capabilities of the 6502. These addressing modes are both natural extensions for the 24-bit world of the 65816 and are also welcomed aid for all of us harried programmers. Rather than document all of the modes available, let's just discuss the new modes available on the 65816 in broad terms.

As you'd expect, many of the new modes involve the use of 24-bit addresses and the additional registers available, but one of the biggest steps forward made by these chips are the many instructions that are now Program Counter Relative (or PC Relative). That is, they work based on the value of the Program Counter Register (ie. where the instruction currently being executed is located in memory). This allows writing what is called "position independent code" and can bring us many benefits.

On the 6502, the PC Relative instructions (BCC, BCS, BEQ, BMI, etc.) are limited to a 256 byte range and then they are limited to only +127 / -128 bytes in either direction from the current PC value. Thus, to send the program any further away in memory, you must either know the absolute 16-bit address of your destination in advance, or go through some convoluted gyrations to figure out where you are now and how far away you want to go. It's possible, but not pretty. With the 65816, several PC-relative instructions have a range of +32767 / -32768. The PER instruction allows for easy access to relative effective addresses for data storage. Branches are similarly extended, but are also enhanced with the addition of the BSL instruction to execute subroutines.

By using position independent instructions, it is possible to vastly expand the capabilities of our systems. You can call code into memory anywhere you have room, instead of the situation we have now where the code must reside precisely where it is written for, or it will crash. One use of this would be the ability to break large programs up into smaller pieces, whereby code is brought into memory from disk as needed and then discarded once its task is completed. Another possibility is to have small utilities reside in reserved sections of memory, ala the Macintosh Desk Accessory or the MS-DOS TSR software, where they can remain ready until needed and yet be independent of the primary program being executed.

The other new addressing modes made available make it possible to write more efficient programs, in that they'll take up less memory and run faster. Whereas with the 6502 it is often necessary to use either the X or the Y register with a particular instruction, most indexed modes on the 65816 allow using either one, filling in many of the obvious gaps in the 6502 instruction set.

Of all the new instructions available in the new chip, as a programmer I am most grateful for MVN and MVP (for MOVE BLOCK NEGATIVE and MOVE BLOCK POSITIVE). These instructions make it possible to move large chunks of data around in memory with a minimum of programming and code space, and do so much faster than would be possible using discrete code. Shrewdly, WDC made these instructions able to move blocks of data across bank boundaries.

I've been running a 65C802 in my C8P-DF for over a month now and have only had one program show any signs of failure. I've run it for long sessions in order to make sure that problems won't arise from OSI parts and design pitfalls from creeping in once the system gets hot. All in all, I'm now very confident in this configuration, however, I would suggest that anyone installing the 65C802 carefully remove the original 6502 so that it can be replaced should it become necessary.

The Future

OK, so where does all of this wonderous hardware leave us? Frankly, we've only scratched the surface. The biggest hurdle remains before us, namely, the operating system. There are a couple of possibilities here.

First, we can modify OS-65D. This would be the easiest and most practical thing to do in the near term. Those of us writing software on an

independent basis are most familiar with OS-65D and can make it jump through hoops as we please. Further, the rest of the world is largely based on this operating system and will make the transition easier.

Second, we can wait to see what the factory does with OS-65U. That's where the big boys play and where most of the more sophisticated software is designed to run. DevTech has been working for many months in this area, and while they haven't made any formal announcements that I'm aware of, they will certainly be doing so in the future. I believe that OS-65U will continue to be the operating system of choice for the business users and we should strive to be in a position to take advantage of whatever the factory does with it. Otherwise, we will be left in the cold once again.

Third, there are a few sources of commercially available operating systems and languages. Paul Chidley is working with the generic 6502 p-System code from Pecan Software. This is the core of the imfamous UCSD p-System that never really ran well on OSI. Of course, SofTech never made much of an effort to understand us, so that wasn't much of a surprise. Pecan Software bought out the p-System a little over a year ago and has been making great strides in micro world. Should Paul be able to adapt the p-System to OSI, it would make several real-world class tools available at a reasonable price, and save us from having to write as much on our own. Whatever else happens, I am very hopeful that Paul can get this up and running.

Finally, we can develop our own operating system. I have made a lot of noise about this in the past 18 months, but there has been little enthusiasm for it. It would be a gargantuan task, and my optimism as to the practicalities of it rise and fall with the tides. I want to do it, but not unless there is some reasonable chance that (a) it will be used and, (b) PEEK can make a few dollars on it. To a certain extent (a) and (b) are diametrically opposed, but I believe that if the software provides a distinct benefit, a modest charge for the OS will not be a problem. In any event, I don't see it exceeding $50 in its initial versions and I will likely charge far less than that just to get the ball rolling.

I hope that this article and the information on the following pages gets you as excited about your system as I am about mine. You can start out small by popping in a 65C802, or you can go whole-hog and buy one of the new 65816 systems from Paul Chidley or Dave Livesay. A lot of people are abandoning their OSI systems entirely, and that's understandable. However, if you're considering such a move, don't overlook the amount of time and trouble it will take to become familiar with any new system as opposed to how easily you can evolve your current set-up into a system that can hold its own against any micro.

Published monthly
Editor: Richard L. Trethewey

Subscription Rates	Air	Surface
US		$22
Canada & Mexico		$30
Europe	$42	$40
Other Foreign	$47	$40

All subscriptions are for one year and are payable in advance in US dollars. For back issues, subscriptions, or other information, write to:

PEEK[65]
P.O. Box 586
Pacifica, CA 94044
415-359-5708

Where Are They Now?

Two very active OSI'ers have been busily making new CPU boards that use the new 65816 chip, but since PEEK last mentioned them, they both have new addresses.

Paul Chidley manufactures the CxP system, which is available in two states of assembly. $60 Cdn (for bare and $199 Cdn for the lazy board, which has the more difficult soldering done for you. Please check current exchange rates for U.S. prices. His new address is:

Paul Chidley
P.O. Box 435
Pickering, Ontario
Canada L1V 2R7

Dave Livesay makes his own CPU board with a 65816, and many external interface options. Dave recently moved back to the U.S. and his address is:

David Livesay
2748 Camino Del Rey
San Jose, CA 95132

Please contact them for details on availability and current prices.

Letters to the Editor

Dear Editor;

First a quick resume' - I started with a used C4P-MF about 5 years ago, which got little use until I latched onto a copy of (OS-65D) v3.3, bought a second C4P-MF, and heard about PEEK[65]. This was about late 1984 and I successfully put everything together to give me a dual-disk, 48K system to which I added an OKI 82 printer. Apart from the learning aspect and a bit of self-satisfaction, I still had little use for the beast until I hit on the idea of using it as an aid for my work, keeping records in order. I manage a technical group in NZ Telecom which is involved with the installation and maintenance of telex, data terminals, modems, and the like.

I had a copy of MDMS which had been butchered, but with Leo Jankowski's help I managed to patch it up. I installed a few POKEs here and there and finally got it up and running with a full-duplex 1200bps, dial-up modem which I access from the office, where I have a Freedom 110 dumb terminal and Siemens ink-jet printer. This was about the beginning of 1986, but a further three months was lost due to something like eight dud 2114's and power supplies which died or became sick with alarming frequency - each time meaning a trip home to re-boot, as it invariably happened when monthly reports were due! Then I lost one disk drive and nothing I did seemed to help, so development stopped and I carried on with the limited space available on a single disk.

Luck smiled about six months ago and I acquired, very cheaply, three Panasonic JB3031, single-sided 40-track drives and it only took a few hours to interface the hardware. The software was a bit tricky, but a PEEK article by David Kuhn, March 1984, solved the problem of slow step rate, though I took it a bit further as the step rate seemed just too slow. By tinkering with the values suggested, I finished up with a reliable rate that is about mid-way between MPI and Siemens. Incidently, I don't exactly understand what, why, or how, but it does work.

By researching for just the information that Eddie (Gieske) gave readers in the December 1986 PEEK, I am now running the system on a dedicated 4-wire circuit at 4800 bps using line drivers and have cranked up the ink-jet printer to work at the same speed. The printer has a large line buffer so it is possible for me to initiate further action using the terminal while the printer is still printing out the results of the previous report.

My first data base only contained 60 records of 13 fields with about 100 bytes per record, but sorting was a pain and got worse as the file grew until it was easier to do manually! More research, more ideas from Leo and PEEK, a little help from one of my staff with a better grasp of BASIC than I have, and "hey presto!", a 150 record sort came down to 4-5 minutes. Admittedly, it is less sophisticated than the original, as it doesn't pack strings with "spaces", but I can't say that I ever notice any problems in my application.

I have eliminated many of the options from the MDMS utilities because I don't need them and had to save disk space. I also had to find a way around the lack of an MDMS Aux disk and its Key File Utility, but all in all, things work very well. As I said at the beginning, Ive been inventing the wheel all this time, so it is doubtful if I have much to offer your readers. Just in case there is someone out there who might be interested, here is what I have done:

(1) I have patched MDMS onto OS-65D v3.3, as this was the only way I knew of being able to backspace and correct typos. An added advantage was that I could install TRAPs in each utility so that they would default to BEXEC* and the main menu and re-set the necessary parameters from the ACIA.

(2) I delete records by filling them with semi-colons, which I can search for when editing and then over-write with new data. The choice of this character is based on its position on my keyboards and because of its ASCII value being greater than any numeral (see next paragraph).

(3) I have a "Re-pack" utility, which I use at the end of each day to sort all the records anew in the order required (essentially by some past date, numerically coded) and stuff them backl on the disk without printing. Records filled with semi-colons are pushed to the end of the file and, when there are too many, I chop them off by using a program that alters the last record number stored in record 0.

(4) All utilities, except a search by field contents using "EDITOR", ignore semi-colons.

(5) I have a new utility based on "GSOSRT", which I call "PTRSRT". This provides a sorted print-out without modifying disk contents and sorts can be done on two fields instead of one.

(6) I am working on a new "REPORT" (about 6 months now) which will produce sorted reports on one or two fields for all, or part of a file. It may not work, but it's fun trying!

(7) One of my files - I have three now - contains 667 records of seven fields in which the first field is a Stock List number from 333 to 999 which coincides with the record number and cannot be modified. The Stock List contains all the numbers and I

only have about 120 of the items listed, but could stock any of them at any time in the future. This file cannot be sorted and empty records have a "0" in field 7 which normally contains a 2-character description of the items of equipment (used for selective reports and by "STATAN" for counting purposes). The whole idea was to gain faster access to stock numbers through the record number, without having to search with "EDITOR" which would have been painfully slow. I use a modified DEF FNA(X) to do the job. Perhaps this principle may be of use to someone? On the other hand, there is probably a proper way of doing it - it's that "wheel" again! I scratched my head when it came to creating the file because I couldn't face typing in all the data when I only needed about 120 actual records, so finished up writing a program based on "EDITOR" to write zeroes into all the records. Then I only had to edit the ones I actually needed.

Now some questions which someone may be able to help me with:

I chose 4800 bps for the serial system as it was easy to play with MDMS at home by simply POKEing 64512 with a different value and then my OKI printer would still work - its highest speed is 1200 bps and it doesn't have a line buffer. But I ran into trouble with WP6502 which I use all the time and had to install a switch - of course I frequently forget to change to 4800 before going to work, so I have to change speed on the terminal and printer there. Can anyone please tell me what I can do to WP650, perhaps in its BEXEC* or INSTAL utility, so that it will divide by 4 when printing? Of course I had problems with the step rate after installing the new drives, but Leo and Paul Chidley's October 1985 article solved that - once again I'm none too sure of the how and the why, but it worked!

I am interested in Home Control, AC Control, and the like, purely for my own satisfaction, but there are great gaps in my knowledge and reference material. Can anyone help with copies of OSI manuals, software, and circuit diagrams? I know roughly what AC Control is all about, but suspect the BSR hardware would not be directly useable here - with diagrams I can modify and build my own, but I would not be capable of writing the software from scratch. I have the HC2 disk - is there a manual? - but this seems to be no more than a demo.

I am also, very slowly, working on building up from bare boards, a CA-20 and CA-22 8-port I/O and A/D, D/A interface. I have manuals and software listings for both, but am unable to get ahold of four chips for the CA-22. They are the DAC-80, ADC-80, DG508, and SHC 298AM? Can anyone help in obtaining these?

Attatched find a listing of the sort routine that I have patched into MDMS - its the only thing that I can think might be of interest for the moment. It features a moving cursor simply to indicate to the remote user that things are happening - the system was so unreliable initially with its frequent crashes - they make no significant difference to transmission time at 4800 bps nor to record access time, which is slow anyway. Hence the next question which I suspect may be unanswerable! Is it possible to do anything to MDMS or DOS which would allow the loading into memory of all the records contained on one track, in one revolution of the disk, and then operating on the data in memory in any way one wishes - more specifically, perhaps loading a whole file before doing anything? I realize that it is likely that OS-65D and MDMS were designed to utilize a very small amount of RAM in the days when much more was unthinkable and the disk represented by far the largest storage area, so any modifications is probably not worth the effort. No doubt OS-65U has all the answers but that is out of the question for now and my need will disappear in the near future in any case I think.

I have looked at the economics and utility of accessing CompuServe by various means, but none are too practicable from this side of the world - its a great thought, but I can see no need for instant communication, particularly where a subscription is involved. If it were merely a matter of paying only for line time I'd probably be a starter, but it would come down to what I get out of it, as I have so little to offer and not a lot of spare time either.

The whole business is very much a part time hobby for me, but I've never regretted buying OSI. Even though it can be frustrating for lack of info, it's fascinating to play around with and to learn about - an end in itself rather than a means to an end. Who wants to buy an IBM that does everything, but slowly, when there is no use for it?

I've enjoyed our chat, albeit one-sided, and hope you may find something of interest in the above. If my questions are not answerable, it won't matter too much as I'll get there in the end. I enjoy reading PEEK enormously and usually find at least one valuable fact which, given a good index at the end of the year, I am assured of being able to find when the need arises. Keep up the good work.

Ray Osborn
9a Nairn Road
Rotorua, New Zealand

Subroutine for faster MDMS sort. Variable names are those used in MDMS (other than those used entirely within this routine) except for "SF" which is the second field number for a two-field sort.

```
300 PRINT: PRINT "LOAD-
ING..."
```

```
310 FOR I = 1 TO EN: DISK
GET,FNA(I)

320 FOR J = 1 TO NF:
INPUT #6, S$(I,J): NEXTJ

330 L$(I) = S$(I,SE) +
S$(I,SF):R(I)=I:
PRINT " ";: NEXT I

340 PRINT: PRINT
"SORTING-"

370 K% - (2^INT(LOG(EN)/
LOG(2))) - 1

380 K%=K%/2: FOR Q = 1 TO
EN - K%: T$ = L$(Q+K%):
R=R(Q+K%): FOR I = Q TO 1
STEP -K%

390 IF L$(I) > T$ THEN
L$(I+K%) = L$(I): R(I+K%)
= R(I):PRINT " ";: NEXT I

400 L$(I+K%) = T$:
R(I+K%) = R: PRINT " ";:
NEXT Q: PRINT " ";: IF K%
> 1 GOTO 380

410 PRINT: PRINT "SORT
COMPLETE-"
```

THE LITTLE GREMLIN THAT COULD

A FAIRY TALE by Jack Noble

Once upon a time, long ago and far away in a land called Ohio, a very little gremlin was born to a computer company by the name of Ohio Scientific. He was such a very little thing but as he awoke and became aware of his surroundings he was very happy because he realized that he was going to be part of a computer and everyone knows that gremlins in computers have a lot more fun than gremlins in cars or lawn mowers or just about anything else. Here he was, just a whiskerof wire, barely large enough to see, but what made him know that he was destined for greatness was not what he was, but where he was, for he was safely hidden under a molex connector where no one could possibly see him, and he was very sure that he could wreak a lot of havoc because, after all, he was going to be part of a computer.

As he sat on the shelf waiting to be assembled he had all sorts of visions of what dastardly deeds he could perform in the final product, and he was sure that he would be a very sucessful gremlin and he could hardly wait to get started. But then a frightening thing happened — He was sent to the quality control department for inspection. "OH NO!" he cried "THEY'LL FIND ME FOR SURE". But to his amazement when the checks were made they totally ignored his home. "Whew" he sighed "that was pretty close and I'm sure glad it's over". Every gremlin knows that once he's made it past quality control, the rest is easy. And now he knew exactly what he was. "That sounds pretty important to me" he thought "I'm backplane #106646 for a C2-4P and thank you very much, WES, for passing me through".

Then he was put on the assembly line and he felt assured of a sucessful gremlin life as they first assembled him onto a power supply and then into a case, but as they began inserting the boards he became a bit worried. The first board they installed didn't use either one of the contacts that he was bridging. "Well, so much for the 540 board" he thought. But as they installed the 527 board and finally the 505 board, he was painfully aware of why the QC people didn't check for him. The sad truth was that he was bridging pins 13 and 14 of the backplane and none of the boards used these pins. "DRAT" he sulked "I may have to wait a while, but someday I'll get one of these humans 'real good'".

Things got pretty grim after that. The computer company sold the C2(and him) to a fellow in Texas who used it daily for five years but never once opened the case. Then he was sold to a guy in Montana who worked the machine constantly but never even concidered hooking anything up to buss pins 13 or 14. Another sale and he was off to eastern Washington and another owner who was not interested in using his flawed connectors. All these years of unfulfilled aspirations were beginning to erode his confidence and then when he thought that his morale could go no lower, the worst of all possible things happened. He was sold to the Seattle OSI club for spare parts. The computer was stripped of everything but 540 board, power supply, and keyboard and he was summarilly tossed into an unused corner of the basement.

"I'M DOOMED" he wailed "I'll die of old age in this musty old basement and when I get to gremlin heaven I won't have any stories to tell about how I tortured those nasty humans with my black magic". "Everyone will know that I failed as a gremlin". The gremlin became very depressed as he lay in that musty basement with nothing to occupy his time but the monthly meetings of the club, which never held and promise of using him for anything and so only added to his feelings of failure.

Then one day the gremlin was suddenly awakened as someone at long last dragged him out of the corner. He was encouraged as the cobwebs were vacuumed away and even more so as his backplane connectors were cleaned to new condition. His excitement rose as he saw a new board about to be installed and reached fever pitch when he saw just what board it was. "MY TIME HAS COME!" he cried (quietly, so as not to alert the victim) for what he saw was that he was about to receive none other than Paul Chidley's new CxP board. "This is going to great" he gleamed "not only does the CxP use pins 13 and 14,

Continued on page 17

Software Spectacular!

C1P/Superboard Cassettes

OSI Invaders	Hangman	Star Trek
Biorhythm	Zulu 9	Racer
SpaceWar	Add Game	Advertisement
Basic Math	High Noon	Tiger Tank
Hectic	Annuity I	Math Intro.
Cryptography	Sampler	

Assortment of 10 for just $20.00!

Specify your preferences, but due to limited quantities, some substitutions will be made.

C4P/C8P Cassettes

Statistics I	Frustration	Space War	Battleship
Annuity II	Mastermind	Trig. Tutor	Powers
Bomber	Loan Finance	Star Trek	Zulu 9
Stock Market	Annuity I	Math Intro	Mathink
Metric Tutor	A.C. Control	Blackjack	High Noon
Electronics Equ.	Star Wars	Math Blitz	Calendar
Prgmble. Calc.	Checking Acct.		

Sargon II Chess Software

Disk version for C8, C4, or C1 (specify)
Regular $34.95 Sale Price $15.00

Cassette version for C8, C4, or C1 (specify)
Regular $29.95 Sale Price $10.00

Extended Monitor

Cassette version for all systems
Regular $50.00

Sale Price $15.00

OS-65U Machine Code Emulator/Tracer

by Carl Eidbo

This program was originally conceived of as a result of my attempts to disassemble OS-65U. I found that, while some subroutines were easy to decode, others were extremely difficult, since the exact path of the program was determined by parameters passed from outside the routine. This program will report the activities of the 6502 processor, allowing much easier understanding of the purpose of the code. I have also used the program several times to debug assembly language routines that I had written.

I have no formal computer education, so I'm sure that many parts of my program could have been more efficiently written. If you have any suggestions for improvements in the program, I would be happy to hear about them - the more specific, the better. I would also be happy to hear any comments you might have on the program itself, as well as any uses you have put it to. If you do any disassembly work on OS-65U or other proprietary operating systems, please report your results to PEEK[65] so that more may benefit and save time by your efforts.

First things first, you may want to put the EDITOR or EXTENDED INPUT program on your working disk. Run it/them from BEXEC* and then have the user end up in MENU. I recommend this procedure. That done, you're ready to begin.

Theory of Operation

The Emulator/Tracer consists of two distinct parts, the machine code which does all of the actual TRACEing, and the BASIC part which allows use under OS-65D and also allows the label files to be moved in and out of memory.

Likewise, the machine code portion can be looked at in separate sections. The biggest part of the code consists of three tables or lists. Table #1 contains a list of all 6502 machine code instructions, the mnemonic, the number of bytes each one occupies in memory, and the type of instruction of addressing mode, broken down into thirteen catagories. Table #2 contains a list of all non-sequential 6502 instructions, and the address of the subroutine that emulates the execution of each. Table #3 contains a list of the thirteen addressing modes and the address of the subroutine that controls the print-out of each.

The rest of the machine code is concerned with setting up the initial operating conditions, moving the instructions to be TRACEd, one at a time, to an area within the TRACER for execution or emulation, and then reporting to an active output devices.

More specifically, the operation of the program goes something like this:

(1) The controlling parameters must be entered into the program.

(2) The program checks to see if a control character is waiting to be entered.

(3) The program moves the first (or next) instructions to be TRACEd into the emulation execution area.

(4) The program first compares the instruction to Table #1 to make sure it's a legal instruction.

(5) Then program next compares the instruction to Table #2 to see if the instruction is sequential or not. If the instruction is sequential, control is passed to the instruction. If the instruction is non-sequential, the instruction is analyzed, and its execution is emulated.

(6) After execution or emulation of the instruction, control is passed to the report portion. Here, the output is sent to any active device(s), according to the type of instruction.

(7) The program loops back to step #2 above.

The actual TRACER machine code (and the Label file in use) reside in high memory starting at $A900. The upper memory limit is changed in the first lines of TBASIC, the BASIC support program. You will not be able to load any programs that take up more than $4900 (18688 or $A900-$6000) bytes of memory until you boot up again.

Installation

[Editor's Note: the installation code is mine, not Carl's. Carl originally sold this package on a commercial basis, but he has graciously allowed me to publish the work. Therefore, he had no need to supply this information. I am supplying a "generic" machine code installation procedure which I hope will suffice. In any event, any problems introduced by this are my fault and not Carl's]

To install TRACER, begin by creating a 10 track file named "SRC1" on an OS-65D diskette to hold the assembly language source code shown in Listing 2. Create a 3 track file named "SRC2" to hold the assembly language program in Listing 3. Create two 1 track files named "CODE1" and "CODE2" on the same diskette. Note the track number where "CODE1" and "CODE2" reside on your disk. Enter the assembly progams with your favorite assembler, save them in the files "SRC1" and "SRC2" created above. Assemble "SRC1" in memory. Get to OS-65D's "A*" prompt and enter:

```
SA TT,1=A900/C
```

where "TT" above is the track number where "CODE1" resides on your disk. Assemble "SRC2" in memory with an offset of $4000. Get to the "A*" prompt again and enter:

SA TT,1=A000/2

where "TT" above is the track number where "CODE2" resides on your disk.

Now reboot your system with OS-65U. Create a BASIC program file named "TBASIC". Make it a full 24576 bytes, just for safety. Create a second file of type "OTHER" named "TRACEM" with "READ" access rights and a password of "PASS". Run "DIR" and write down the disk address where "TRACEM" resides. Run the program "LOAD48". Replace the disk in drive A with the OS-65D disk that holds "CODE1" and "CODE2". At the "A*" prompt, enter:

C6000TT

again, where "TT" is the track number for "CODE1". Note that LOAD48 will fill in some characters as you enter the above in order to make your command look like a standard OS-65D command. Replace the disk in drive A with your OS-65U diskette. At the "OK" prompt in BASIC, enter:

```
NEW 3100
10 REM
SAVE "TBASIC"
```

This does three things. (1) Clears the workspace and sets up a 3100 byte buffer to protect the machine code we just read into memory with LOAD48. (2) Enter a dummy first line of the program. (3) Permanently saves the machine code on disk as a part of the program file "TBASIC" for good keeping in case problems arise when you enter the program from the listing.

You may now enter the BASIC program from Listing 1. When you've finished, again enter SAVE"TBASIC" to store the real program.

Run LOAD48 again. Insert your OS-65D diskette in drive A. Enter:

CA900TT

where "TT" above is the track number where "CODE2" resides. From here, things get sticky. You need a disk read/write utility to write the contents of memory at $A900 to the OS-65U file "TRACEM". See the program in Listing 4.

To Run the EMULATOR/TRACER

(1) Select the EMULATOR/TRACER from the menu.

(2) Select (or don't select) a label file to use.

(3) Select the Mode of Operation:

STANDARD MODE:

You must first enter the starting address of the code that you want to TRACE. You must enter the starting values of all the microprocessor registers, except for the Stack Pointer. Finally, select the Output Device(s), and TRACEing will begin.

EXPLODE MODE:

As above, you must enter the starting address of the code you want to TRACE, then enter the Output Device(s). The next time the microprocessor executes the code at the point you have specified, TRACEing will begin. As an example, the SYNTAX ERROR routine begins at $0E1E. Enter this as the starting address, and after you are returned to the immediate mode of BASIC, enter some garbage and you will see what happens when a SYNTAX ERROR occurs.

Please Note: Once you have entered the machine code portion, you no longer have editing functions such as destructive backspace, so enter the data carefully!

TRACER and OS-65D

Once the TRACER and any desired label file have been loaded into memory by OS-65U, the program is completely self-contained. You can reboot your system with OS-65D and run TRACER by following these steps:

(1) Boot up and set up TRACER normally under OS-65U.

(2) Reset the computer.

(3) Boot up OS-65D.

(4) Get to BASIC's "OK" prompt.

(5) Enter "EXIT" to get to 65D's "A*" prompt.

(6) Enter "EM" to invoke the Extended Monitor.

(7) To run in Standard Mode, enter "GA900".

(8) To run in Explode Mode, enter "GA904".

Control Characters

Several control characters have been utilized by the TRACER to help manage the huge volume of data the program can generate.

(1) Control-S:
The first <CTRL>'S' that is received will stop the TRACER after it prints the line it is on. Each successive <CTRL>'S' will cause one line of code to be output (single step).

(2) Control-Q:
<CTRL>'Q' will turn the TRACER back on after a <CTRL>'S'.

(3) Control-C:
The TRACER will transfer control to the reset routine (ie. "H/D/M?") upon receipt of <CTRL>'C'. The reason for resetting the computer is that I have no way of knowing what you are TRACEing or what has happened during the TRACE. Many times, you can re-enter the immediate mode of BASIC by pressing "M" and then "G", following the reset prompt. Usually, this will return control to you, just as if you had entered a <CTRL>'C' during the normal running of the program.

(4) Control-P:
<CTRL>'P' is a printer control character. The printer will be toggled on and off as each <CTRL>'P' is received.

Error Conditions

If any errors are detected during TRACEing, the program reacts just as if a <CTRL>'C' had been entered. The only error that you are likely to encounter is "illegal command". That is, the TRACER has detected an unlisted command at the next address it is to emulate. There is no identifying message printed when an error occurs, just the reset prompt.

Using a Label File

The TRACER may be run with or without a Label file in use. The following is a brief description of what happens in the program when a Label file is in use.

Each time the TRACER fetches an instruction, it searches through the entire Label file for a start address that is the same as the address of the instruction. If a match is not found, the TRACER prints out the machine code relating to that instructions, and goes on to the next part of the interpretation. If a match is found, the TRACER prints out the name of the Label in place of the machine code. The TRACER then looks at the end address for that Label. If the end address is zero, the TRACER continues on normally. If the end address is not zero, the TRACER stores the end address, finishes printing the line it is on, and then shuts down the output. The output will not be turned on again until the end address is encountered during no-output operation.

Please Note: The TRACER runs almost identically when the output is off as compared to when the output is on. About the only difference is that the Labels are not searched for, and the output subroutines are bypassed within the program. Also note that since this program executes thousands of microprocessor instructions for each line that is TRACEd, it may take a while for the output to be turned on again, depending on the number of instructions executed during the subroutine!

You may increase (or decrease) the number of Label files in use if you (1) change the subroutine at line 42000 in the program TBASIC. (2) Create (or delete) new files with the same attributes (size, type, access rights) as those already on the disk, and then copy an old file to the new one. The new filename must have a number in it corresponding to the selection on the menu in TBASIC. The maximum number of Labels allowed in any file is 255.

Bugs and Limitations

Unfortunately, with any undertaking of this size, there are usually shortcomings. I have eliminated all of the bugs I could find, but I still do not claim that this program set is perfect. The following is a list of "shortcomings" which are not necessarily bugs, but could cause the user difficulty if they were to be discovered accidentally.

If you should find any defects, or just have suggestions for improvements, please let me know. I will attempt to correct any serious bugs, and I may incorporatesuggestions into updated versions of the system.

(1) When running the TRACER, you will notice that the BRK bit in the condition code register is always "1". I have not been able to determine why this is. I believe it really should be "0".

(2) I am not 100% sure that the emulation codes for the BRK and RTI instructions are correct. I have talked to several supposed experts, searched several publications, and have come up with more than one answer.

(3) Since the only way I can get at the condition code register is by means of the PHP or PLP instructions, which use the stack, some operations involving direct access of page one may not be emulated properly. Consider this example:

```
*=$6000

LDA     #$7F
TSX
PHA
PLA
LDA     $0100,X
```

At the end, the Accumulator obviously should contain the same value that it was given in the beginning. This would be the case if you were to run this "program" normally. But since I need to access the condition code register for each line to be TRACEd, and I must use PHP, the stack location that was relinquished by the PLA instruction will be filled by PHP (as well as other possible stack uses within TRACER). If for some reason you wanted to recover that location later, it would no longer contain the same value as was stored by PHA.

(4) Time-dependent programs (subroutines) will not work during TRACEing. The TRACER executes thousands of lines of its own code for each line it prints to the output devices, thus throwing off the timing of the program being TRACEd. During a disk I/O routine, the microprocessor looks for a signal from the disk drive that the index hole on the disk has passed through a light detecting circuit. When it receives this signal, it then goes into a loop that has been calculated to take a certain amount of time (equal to a certain distance on the rotating disk). When the loop is finished, the actual I/O to the magnetic surface of the disk begins. If you were TRACEing this routine, the I/O would begin much, much later than it was supposed to, if it occurs at all. The operating system would probably detect an error. This means that

you would probably end up TRACEing some error-handling routine instead of the disk handling code you intended to TRACE.

```
10 REM ^^^ 3080 Bytes of Machine Code and File Space above ^^^
15 DIM LR(20) : SP=0 : REM Prepare User Stack
17 :
18 REM  *** ************ ***
20 REM  ***  File Name:  ***
25 REM  ***    TBASIC    ***
30 REM  *** ************ ***
35 REM  ***   Author:    ***
40 REM  ***  CARL EIDBO  ***
45 REM  *** ************ ***
50 REM
55 REM  *** ************ ***
60 REM  ***   Started:   ***
65 REM  ***   11/09/82   ***
70 REM  *** ************ ***
75 REM  *** Latest Rev.: ***
80 REM  ***   06/15/83   ***
90 REM  *** ************ ***
95 :
99 X=43264:Y=INT(X/256):POKE133,Y:Y=X-Y*256:POKE132,Y:CLEAR
100 REM This is the BASIC support program for all features of TRACER.
110 :
130 LR(SP)=140:SP=SP+1:GOTO40000
140 GOTO2000
600 :
610 REM Substitute for RETURN
620 SP=SP-1 : GOTO LR(SP)
700 :
2000 REM Menu 1
2010 T1=22:PRINTSC$
2020 PRINTTAB(T1);"TRACER menu 1"
2030 PRINTTAB(T1);"------":PRINT:PRINT
2040 PRINTTAB(T1);"(01) Examine/Edit a LABEL file"
2050 PRINTTAB(T1);"(02) Load TRACER into operating position"
2060 PRINT
2065 PRINTTAB(T1);"( /) Return to Main Menu"
2066 PRINT
2070 PRINTTAB(T1-15);"Your Selection";:INPUTM1$:M1=VAL(M1$)
2075 IFM1$="/"THENRUN"MENU"
2076 IFM1$="STOP"THENSTOP
2080 ON M1 GOTO 2500,2110
2090 PRINTCHR$(7):GOTO2010
2100 :
2110 REM Move TRACER machine code to High Memory
2120 CH=1:F$(CH)="TRACEM":P$(CH)="PASS":DV$(CH)=DV$(0)
2125 LR(SP)=2130:SP=SP+1:GOTO48000
2130 DA=FS(CH):NB=3584:AA=AD(2):CLOSECH
2140 RW=0:LR(SP)=2146:SP=SP+1:GOTO47000
2146 GOTO 8000
2150 :
```

Listing 1

```
2500 REM Edit LABEL file
2510 PRINTSC$:PRINTTAB(T1);"LABEL file editor"
2520 PRINTTAB(T1);"--------":PRINT:PRINT
2540 PRINTTAB(T1-5);"Which LABEL file would you like to Examine/Edit?"
2550 PRINT:LR(SP)=2555:SP=SP+1:GOTO42000
2555 IFLF<10RLF>LF%THENPRINTBP$:GOTO2000
2560 RA=AD(1):RW=0:LR(SP)=2570:SP=SP+1:GOTO43000:REM Move LABEL file
2570 PRINTSC$:PRINTTAB(T1);"Edit LABEL file #";LF
2575 PRINTTAB(T1);"---------":PRINT:PRINT
2580 PRINTTAB(T1);"(01) Display & Edit LABEL file"
2590 PRINTTAB(T1);"(02) List  LABEL file to printer"
2600 PRINTTAB(T1);"(03) Erase LABEL file"
2630 PRINT
2640 PRINTTAB(T1);"(99) Save modified LABEL file"
2650 PRINTTAB(T1);"     and return to Menu 1"
2660 PRINTTAB(T1);"( /) Return to Menu 1 (don't save file)"
2670 PRINT
2680 PRINTTAB(T1-15);"Your Selection";:INPUTX$
2690 IFX$="/"THEN2000
2700 LE=VAL(X$):IFLE=99THEN4000
2710 IFLE<10RLE>3THENPRINTBP$:GOTO2570
2720 ONLEGOTO3000,4500,5000
2730 :
3000 REM Display Contoller
3002 LR(SP)=3005:SP=SP+1:GOTO20700
3005 PG=1
3010 LP%=RN%/DL+1:IFLP%=00RLP%=1THENPG=1
3050 BG=(PG-1)*DL:EN=BG+DL-1
3054 IF RN%=0 THEN BG=0 : EN=0 : GOTO3070
3055 IFBG=>RN%THEN2570
3060 IFEN>RN%-1THENEN=RN%-1
3070 PRINTSC$:PRINTTAB(0);"Examine/Edit LABEL file #";LF;
3080 PRINTTAB(35);"Page";PG:FORX=1TO43:PRINT"-";:NEXT:PRINT
3100 X=BG:Y=EN:LR(SP)=3105:SP=SP+1:GOTO20500
3105 PRINT
3110 PRINTTAB(0);"Enter:  <Record #>  to Edit; <-Record #> to Delete"
3120 PRINTTAB(08);"<+> to Add a Record; <CR> to Continue Display";
3130 PRINTSPC(5);:INPUTX$
3140 IFX$<>""THEN3160
3150 IFPG<LP%THENPG=PG+1:GOTO3050
3155 GOTO2570
3160 Y$=LEFT$(X$,1):IFY$<>"+"THEN3300
3165 :
3170 IFRN%=>255THENPRINTBP$:GOTO3070
3173 PRINTSC$:PRINTTAB(15);"Add a Record (#";RN%+1;")"
3174 PRINTTAB(15);"----------":PRINT:PRINT
3175 LR(SP)=3180:SP=SP+1:GOTO20800
3180 ED=RN%+1:LR(SP)=3185:SP=SP+1:GOTO20000
3185 IFER<>0THEN3070
3190 RN%=ED:POKER4,RN%-1:IFED=ENTHENPG=PG+1
3200 GOTO3010
3250 :
3260 REM Delete a record
3300 IFY$<>"-"THEN3500
3310 ED=VAL(RIGHT$(X$,LEN(X$)-1)):IFED<10RED>RN%THENPRINTBP$:GOTO3070
3320 PK=RA+(ED-1)*10+4:POKEPK,255:LR(SP)=3325:SP=SP+1:GOTO20600
3325 RN%=RN%-1:IFRN%<BGANDPG>1THENPG=PG-1
3330 GOTO3010
3499 :
```

Listing 1 continued

```
3500 REM EDIT a record
3510 ED=VAL(X$):IFED<1ORED>RN%THENPRINTBP$:GOTO2570
3520 PRINTSC$:PRINTTAB(10);"EDIT Record #";ED:PRINT
3530 PRINT"Old Record:":PRINT
3540 X=ED-1:Y=X:LR(SP)=3550:SP=SP+1:GOTO20500
3550 LR(SP)=3560:SP=SP+1:GOTO20000
3560 IFER=1THEN3070
3999 GOTO3070
4000 :
4010 REM Save TRACER File
4020 RW=1:LR(SP)=2000:SP=SP+1:GOTO43000
4030 RW=1:GOSUB43000:GOTO2000
4500 :
4510 REM List to Printer
4520 PRINT#LP,TAB(5);"Start";TAB(15);"End";TAB(25);"Label"
4530 PRINT#LP,TAB(4);"Address";TAB(13);"Address"
4540 FOR L1=1TO35:PRINT#LP,"-";:NEXTL1:PRINT#LP:PRINT#LP
4550 FOR L1=1TORN%:PK=RL%*(L1-1)+AD(1):X=PEEK(PK)+PEEK(PK+1)*B1
4555 LR(SP)=4560:SP=SP+1:GOTO21010
4560 L$=STR$(L1):L$=RIGHT$(L$,LEN(L$)-1):L$=RIGHT$("00"+L$,3)
4570 PRINT#LP,TAB(0);L$;")";TAB(5);X$;:X=PEEK(PK+2)+PEEK(PK+3)*B1
4580 LR(SP)=4583:SP=SP+1:GOTO21010
4583 IFX$="$0000"THENX$="$--"
4585 PRINT#LP,TAB(15);X$;:FORL2=4TO9:PRINT#LP,TAB(25);
4590 PRINT#LP,CHR$(PEEK(PK+L2));:NEXTL2:PRINT#LP:NEXTL1
4600 IFPEEK(15908)<>PEEK(14457)ANDLP=5THENPRINT#LP:GOTO4600
4980 GOTO2570
4990 :
5000 REM Erase LABEL File
5010 PRINTBP$:PRINTSC$:PRINTTAB(15);"Erase Entire LABEL File"
5015 PRINTTAB(15);"-----------"
5016 PRINT:PRINT"Are You Sure? If so, enter 'YES'";:INPUTX$
5017 IFX$="YES"THENRN%=0:GOTO2570
5020 GOTO2570
5030 :
8000 REM Menu 2
8010 PRINTSC$
8020 PRINTTAB(T1);"TRACER menu 2"
8030 PRINTTAB(T1);"------":PRINT:PRINT
8040 PRINTTAB(T1-5);"Would you like to use a LABEL file with TRACER?"
8050 PRINT
8060 LR(SP)=8070:SP=SP+1:GOTO42000
8070 IFLF=0THEN8220
8120 IFLF<1ORLF>LF%THENPRINTBP$:GOTO2000
8130 RA=PEEK(AD(4))+256*PEEK(AD(4)+1):RW=0
8135 LR(SP)=8140:SP=SP+1:GOTO43000
8140 POKE(AD(5)),RN%:GOTO9000
8200 :
8210 REM No LABEL file
8220 RN%=0:GOTO8140
9000 :
9010 REM TRACER all set up, now what?
9020 PRINTSC$:PRINTTAB(T1);"TRACER ready to run"
9030 PRINTTAB(T1);"---------":PRINT:PRINT
9040 PRINTTAB(T1);"(01)  Run TRACER directly"
9050 PRINTTAB(T1);"(02)  Set TRACER to EXPLODE mode"
9070 PRINT
```

Listing 1 continued

```
9080 PRINTTAB(T1);"( /)  Start Over"
9090 PRINT:PRINTTAB(T1-15);"Your selection";:INPUTX$
9100 SE=VAL(X$):IFX$="/"THEN2000
9110 IFSE<1ORSE>3THENPRINTBP$:GOTO9010
9120 ONSEGOTO9220,9320
9200 :
9210 REM Go to Direct Mode
9220 RA=AD(2):GOTO9900
9300 :
9310 REM Go to EXPLODE Mode
9320 RA=AD(3):GOTO9900
9900 :
9910 REM GO
9920 HB%=RA/B1:LB%=RA-HB%*B1:POKE8778,LB%:POKE8779,HB%:X=USR(0)
9930 END
20000 :
20010 REM Input a Record Sub
20020 ER=0:PRINT:PRINT"Enter New Record : "
20030 PRINT:PRINTTAB(1);"Start Addr.";
20040 INPUTE1$:IFLEFT$(E1$,1)<>"$"THENE1$="$"+E1$
20050 X$=E1$:LR(SP)=20052:SP=SP+1:GOTO21100
20052 E1=X::IFX=00RX=1E9THENPRINTBP$:GOTO20170
20055 PRINT:PRINTTAB(1);"End Address? (If any)";
20060 INPUTE2$:IFLEFT$(E2$,1)<>"$"THENE2$="$"+E2$
20065 IFE2$="--"THENE2$="$0000"
20070 X$=E2$:LR(SP)=20072:SP=SP+1:GOTO21100
20072 E2=X:IFX=1E9THENPRINTBP$:GOTO20170
20075 PRINT:PRINTTAB(26);"Label";
20080 INPUTE3$:IFE3$="/"ORE3$=""THENPRINTBP$:GOTO20170
20085 PRINT
20090 PRINT:PRINT"Is this Correct? (<CR> = Yes)";:INPUTX$
20100 IFX$<>""THENPRINTBP$:GOTO20170
20110 PK=RA+(ED-1)*10
20120 HB%=E1/B1:LB%=E1-HB%*B1:POKEPK+0,LB%:POKEPK+1,HB%
20130 HB%=E2/B1:LB%=E2-HB%*B1:POKEPK+2,LB%:POKEPK+3,HB%
20140 IFLEN(E3$)<6THENE3$=E3$+" ":GOTO20140
20150 FORY=1TO6:X=ASC(MID$(E3$,Y,1)):POKEPK+3+Y,X:NEXT
20160 GOTO20180
20170 ER=1:REM Error
20180 GOTO 620 :REM Return
20190 :
20500 :
20501 REM Call Display Routine
20510 REM X = 1st  Record to Display
20520 REM Y = Last Record to Display
20523 LR(SP)=20525:SP=SP+1:GOTO20800
20525 IFRN%=0THENPRINTTAB(15);"File Empty":GOTO 620
20530 HB%=S3/B1:LB%=S3-HB%*B1:POKE8778,LB%:POKE8779,HB%
20540 POKE24589,X:POKE24590,Y:Z=USR(0):GOTO 620
20550 :
20600 REM Call Delete Routine
20605 IFRN%=0THEN620
20606 LR(SP)=20610:SP=SP+1:GOTO43100
20610 HB%=S1/B1:LB%=S1-HB%*B1:POKE8778,LB%:POKE8779,HB%
20620 Z=USR(0):GOTO 620
20630 :
```

Listing 1 continued

```
20700 REM Call Sort Routine
20705 IFAN%<2THEN620
20706 LA(SP)=20710:SP=SP+1:GOTO43100
20710 HB%=S2/B1:LB%=S2-HB%*B1:POKE8778,LB%:POKE8779,HB%
20720 Z=USR(0):GOTO 620
20730 :
20800 REM Print Header
20810 PRINTTAB(3);"Rec. #";TAB(13);"Start";
20820 PRINTTAB(24);"End";TAB(33);"LABEL":PRINT:GOTO620
20830 :
21000 REM Convert dec to hex (X to X$)
21010 X$="$":FORL9=1TO4:EX(2)=16:EX(3)=4-L9
21013 LA(SP)=21015:SP=SP+1:GOTO22020
21015 Y%=X/EX(1):X=X-Y%*EX(1)
21020 Y%=Y%+48-7*(Y%>9):X$=X$+CHR$(Y%):NEXTL9:GOTO620
21030 :
21100 REM Convert hex to dec (X$ to X)
21110 X$=RIGHT$(X$,LEN(X$)-1):X=0:IFX$=""THEN620
21115 Z=LEN(X$)
21120 FORL9=1TOZ:Y=ASC(MID$(X$,L9,1)):Y=Y-48+7*(Y>57)
21125 IFY<00RY>15THENX=1E9:GOTO620
21130 EX(2)=16:EX(3)=Z-L9:LA(SP)=21140:SP=SP+1:GOTO22020
21140 X=X+Y*EX(1):NEXTL9:GOTO620
22000 :
22010 REM Exponential Replacement  EX(1)=EX(2)^EX(3)
22020 IF EX(3)=0 THEN EX(1)=1 : GOTO620
22030 EX(1)=EX(2) : IF EX(3)=1 THEN GOTO620
22040 FOR LO=2 TO EX(3) : EX(1)=EX(1)*EX(2) : NEXT LO : GOTO620
40000 REM  Set Status
40010 CLOSE
40020 B1=256:B2=256*B1:B3=256*B2
40030 X=PEEK(9832):IFX>127THENX=X-124
40040 DV$(0)=CHR$(X+65)
40050 FLAG9:FLAG1:FLAG21:POKE2888,0
40055 POKE 2972,13 : POKE 2976,13 : REM Allow : , in Input
40070 DL=16: REM Max # of vertical display lines
40080 OPEN"SET-UP",1:INPUT#1,X:LI$=CHR$(X):REM Lead-in Char.
40090 INPUT#1,X:CLOSE:SC$=LI$+CHR$(X):REM Screen Clear Char.
40100 BP$=CHR$(7)
40110 DS$="----------"
40120 CB=9889:NB=3584
40122 S1=24576:REM Delete  Routine
40123 S2=24579:REM Sort    Routine
40124 S3=24582:REM Display Routine
40125 R4=24588 : REM # of records in LABEL file.
40140 DIMAD(5)
40145 AD(1)=25088 : REM LABEL file start address
40150 AD(2)=43264 : REM TRACER start address in high memory ($A900)
40160 AD(3)=AD(2)+4 : REM TRACER (EXPLODE mode) address
40170 AD(4)=43288 : REM Address of Label File for TRACER
40180 AD(5)=43291 : REM # of Records in Label File in use
40200 LP=5 : REM Printer output to device #5
40990 GOTO620
40999 :
41000 REM  Return Status
41010 CLOSE:DEVDV$(0)
41020 FLAG10:FLAG2
41030 POKE8778,208:POKE8779,16:REM Restore USR(X) = FC ERR
41040 GOTO620
41050 :
```

Listing 1 continued

```
42000 REM LABEL file listing
42010 LF%=3:REM Present # of allowed LABEL files
42020 RL%=10:REM Record Length
42030 PRINTTAB(T1);"(01) OS-65U Ver 1.42 Floppy DOS"
42040 PRINTTAB(T1);"(02) Not in Use"
42050 PRINTTAB(T1);"(03) Not in Use"
42055 PRINT:PRINTTAB(T1);"( /) None of the above"
42060 PRINT:PRINTTAB(T1-15);"Your Selection";:INPUTX$
42070 LF=INT(VAL(X$)):GOTO620
42080 :
43000 REM Move LABEL file
43010 CH=1:F$(CH)="TRACE":P$(CH)="PASS":DV$(CH)=DV$(0)
43020 X$=STR$(LF):X$=RIGHT$(X$,LEN(X$)-1):F$(CH)=F$(CH)+X$
43030 LR(SP)=43032:SP=SP+1:GOTO48000
43032 INDEX<CH>=0:IFRW=1THEN43040
43035 INPUT#1,RN%:LR(SP)=43038:SP=SP+1:GOTO43100
43038 GOTO43050
43040 PRINT#1,RN%
43050 CLOSECH:IFRN%=0THEN43070
43060 DA=FS(CH)+10:NB=RN%*RL%:LR(SP)=43070:SP=SP+1:GOTO47000
43070 GOTO620
43080 :
43100 POKER4,(RN%-1)-(RN%=0):GOTO620:REM Poke # of Records
43110 :
47000 REM USR(X) I/O SUB
47010 POKE8778,192:POKE8779,36
47020 POKE9432,243:POKE9433,40:POKE9435,232:POKE9436,40
47030 D3=INT(DA/B3):D0=DA-D3*B3:D2=INT(D0/B2):D0=D0-D2*B2
47040 D1=INT(D0/B1):D0=D0-D1*B1:POKECB+1,D0:POKECB+2,D1
47050 POKECB+3,D2:POKECB+4,D3:POKECB+5,NB-INT(NB/B1)*B1
47060 POKECB+6,INT(NB/B1):POKECB+7,RA-INT(RA/B1)*B1
47070 POKECB+8,INT(RA/B1): ER=USR(RW) :IFER<>0THEN47090
47080 GOTO620
47090 PRINTCC$(0):PRINTCC$(1):REM USR(X) Error
47100 PRINT"******** USR(";RW;") ERROR";ER;
47105 PRINT"AT ADDRESS";DA;" ********"
47110 STOP:GOTO47080
47120 :
48000 REM Get a File's Params from File Buffer
48010 DEVDV$(CH):OPENF$(CH),P$(CH),CH:X=CB+10+CH*8
48020 FS(CH)=B1*PEEK(X+0)+B2*PEEK(X+1)+B3*PEEK(X+2)+16:REM Start
48030 FH(CH)=B1*PEEK(X+3)+B2*PEEK(X+4)+B3*PEEK(X+5):REM Length
48040 GOTO620
48050 :
```

Listing 1 (end)

---

but it uses 13 for RESET and 14 for BANK0, so whenever bank0 is selected I'll cause a reset to occur and reset always selects bank0 so the computer will be in a constant state of reset". "Not only that, this poor slob plugging it in has just finished building it and has no idea if it works or not and I'll be the verylast thing in the world that he would ever suspect of causing the problem".

"I LOVE IT! I LOVE IT! I LOVE IT!"

The author believes that the events that transpired over the next few hours should not be included in such a gentle story but will assure you that the backplane gremlin has been dispatched to gremlin heaven where I am sure that he is very happy telling all the horrid details that have been omitted here.

CxP COMES TO SEATTLE — MORE LATER

```
                ;       TRACEP   Version 1.00 .01
                ;
                ;    Machine Language Trace Utility
                ;    ==============================
                ;    Copyright 1983  by  Carl Eidbo
                ;
                ;      Date Started    : 04/12/82
                ;      Latest Revision : 07/11/83
                ;                        01/04/84   1.01  Line 9820
                ;
                ;================================================
                ;
                ;                EQUATES
                ;                =======
                ;
F400=           PRNTR   = $F400 ; Parallel Interface Address
FC00=           CON     = $FC00
                ;
A900            *       = $A900 ; Sits in High Memory
                ;
                ; ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
                ;
A900 A900       TRACE1  LDA #00 ; Enter here from BASIC for Manual Mode
A902 F002               BEQ TRACE
A904 A901       TRACE2  LDA #01 ; Enter here from BASIC for EXPLODE Mode
A906 8D2DA9     TRACE   STA TRMODE
A909 4C37A9             JMP START1
                ;
                ; ***   ***   ***   ***   ***   ***   ***   ***   ***   ***   ***
                ;
A90C 60B1       BOD1    .WORD TABLE1
A90E 06         RL1     .BYTE 06
A90F 96         RN1     .BYTE 150
A910 39B0       BOD2    .WORD TABLE2
A912 03         RL2     .BYTE 03
A913 0E         RN2     .BYTE 14
A914 00AF       BOD3    .WORD TABLE3
A916 02         RL3     .BYTE 02
A917 0D         RN3     .BYTE 13
A918 7FB6       BOD4    .WORD TABLE4
A91A 0A         RL4     .BYTE 10
A91B 00         RN4     .BYTE 00 ; To be filled in by BASIC , if = 0 then
                                   don't use LABEL file.
                ;
A91C 00         TEMPA   .BYTE 00
A91D 00         TEMPX   .BYTE 00
A91E 00         TEMPY   .BYTE 00
A91F 00         TEMPP   .BYTE 00
A920 00         TEMPA1  .BYTE 00
A921 00         TEMPX1  .BYTE 00
A922 00         TEMPY1  .BYTE 00
A923 00         TEMPA2  .BYTE 00
A924 0000       TEMPPC  .DBYTE 00
A926 00         LINECT  .BYTE 00
A927 2E         TABCHR  .BYTE '.'
A928 00         SSTEP   .BYTE 00 ; 1=Single Step on ; 0=off
A929 00         NSEQSP  .BYTE 00 ; Print blank line if Non-Seq Inst.  0=No
A92A 01         OUTSW1  .BYTE 01 ; 0=Off, 1=On
A92B 00         OUTSW2  .BYTE 00
A92C 00         OUTDEV  .BYTE 00 ; 0=Console, 1=Printer, 2=Both
A92D 00         TRMODE  .BYTE 00 ; 0 = Manual Mode, 1 = Auto mode
A92E 4C15AA     AUTOST  JMP START2 ; Auto-Start Vector
A931 0000       TMPREC  .DBYTE 00,00,00
A933 0000
A935 0000
                ;
                ;================================================
                ;
                ;                START
                ;                =====
                ;
                ;
A937 20EBAE     START1  JSR SETCC
A93A A201               LDX #01
A93C 8E2AA9             STX OUTSW1
A93F 8E2BA9             STX OUTSW2
A942 68                 PLA ; Pull USR(X) Ret to BAS addr from stack
A943 8D13AA             STA RETBAS+1
A946 68                 PLA
A947 8D14AA             STA RETBAS+2
                ;
A94A EE13AA             INC RETBAS+1
A94D D003               BNE MESSG1
A94F EE14AA             INC RETBAS+2
A952 A200       MESSG1  LDX #00
A954 8E28A9             STX SSTEP
A957 8E26A9             STX LINECT
A95A 8E8AAE             STX STEPCT
A95D 8E8BAE             STX STEPCT+1
A960 8E8CAE             STX STEPCT+2
A963 BDEAB4     NXTMG1  LDA MSG1,X
A966 C9FF               CMP #$FF
A968 F006               BEQ GETADR
A96A 2038AD             JSR CONOUT
A96D E8                 INX
A96E D0F3               BNE NXTMG1
A970 A202       GETADR  LDX #02
A972 2076AD             JSR GETHEX
A975 9D23A9             STA TEMPPC-1,X
A978 CA                 DEX
A979 D0F7               BNE GETADR+2
A97B A000               LDY #00
A97D AE2DA9             LDX TRMODE ; Skip register contents if in Auto-Mode
```

```
 990 A980 D021           BNE GETOUT
1000 A982 A200           LDX #00
1010 A984 BD5DB5 INIZRG  LDA MSG3,X
1020 A987 E8             INX
1030 A988 C9FF           CMP #$FF
1040 A98A F006           BEQ INZREG
1050 A98C 2038AD         JSR CONOUT
1060 A98F 4C84A9         JMP INIZRG
1070 A992 2076AD INZREG  JSR GETHEX
1080 A995 C003           CPY #$03
1090 A997 D002           BNE SAVREG
1100 A999 29EF           AND #%11101111 ; Restrict P register contents
1110 A99B 991CA9 SAVREG  STA TEMPA,Y
1120 A99E C8             INY
1130 A99F C004           CPY #04
1140 A9A1 D0E1           BNE INIZRG
1150 A9A3 2054AD GETOUT  JSR CRLF
1160 A9A6 A200           LDX #00  ; Get Output Device
1170 A9A8 BD0AB5 PTMSG2  LDA MSG2,X
1180 A9AB C9FF           CMP #$FF
1190 A9AD F006           BEQ ANS
1200 A9AF 2038AD         JSR CONOUT
1210 A9B2 E8             INX
1220 A9B3 D0F3           BNE PTMSG2
1230 A9B5 202CAD ANS     JSR CONIN
1240 A9B8 38             SEC
1250 A9B9 E931           SBC #'1
1260 A9BB F00A           BEQ OK      ; 0 = Console only
1270 A9BD C901           CMP #01
1280 A9BF F006           BEQ OK      ; 1 = Printer only
1290 A9C1 C902           CMP #02
1300 A9C3 F002           BEQ OK      ; 2 = Both
1310 A9C5 D0EE           BNE ANS
1320 A9C7 8D2CA9 OK      STA OUTDEV
1330 A9CA 18             CLC
1340 A9CB 6931           ADC #'1
1350 A9CD 2038AD         JSR CONOUT
1360 A9D0 2054AD         JSR CRLF
1370 A9D3 AE2DA9         LDX TRMODE
1380 A9D6 F04B           BEQ BEGIN
1390 A9D8 A202           LDX #$02 ; Move Auto-Start vector to Work Area
1400 A9DA BD2EA9 MOVST1  LDA AUTOST,X
1410 A9DD 9DABAB         STA WKAREA,X
1420 A9E0 CA             DEX
1430 A9E1 10F7           BPL MOVST1
1440 A9E3 AD24A9         LDA TEMPPC
1450 A9E6 8DFEA9         STA STADD1+1
1460 A9E9 8D05AA         STA STADD2+1
1470 A9EC 8D1EAA         STA STADD3+1
1480 A9EF AD25A9         LDA TEMPPC+1
1490 A9F2 8DFFA9         STA STADD1+2
1500 A9F5 8D06AA         STA STADD2+2
1510 A9F8 8D1FAA         STA STADD3+2
1520 A9FB A202           LDX #$02
1530 A9FD BDFFFF STADD1  LDA $FFFF,X ; Swap (Activation Point & WKAREA)
1540 AA00 48             PHA
1550 AA01 BDABAB         LDA WKAREA,X
1560 AA04 9DFFFF STADD2  STA $FFFF,X
1570 AA07 68             PLA
1580 AA08 9DABAB         STA WKAREA,X
1590 AA0B CA             DEX
1600 AA0C 10EF           BPL STADD1
1610 AA0E A900           LDA #$00
1620 AA10 AA             TAX
1630 AA11 A8             TAY
1640 AA12 4CFFFF RETBAS  JMP $FFFF ; Return to BASIC, after Auto-mode configuratic
1650 AA15 20AFAD START2  JSR PRSRVE ; Start here for "Auto-Start"
1660 AA18 A202           LDX #$02 ; Move Bytes back to Activation Point.
1670 AA1A BDABAB MOVST3  LDA WKAREA,X
1680 AA1D 9DFFFF STADD3  STA $FFFF,X
1690 AA20 CA             DEX
1700 AA21 10F7           BPL MOVST3
1710                     ;  ===   ===   ===
1720 AA23 20EBAE BEGIN   JSR SETCC
1730 AA26 2054AD         JSR CRLF
1740 AA29 206EAE         JSR HEADER
1750 AA2C A900    NEXT   LDA #00
1760 AA2E 8D29A9         STA NSEQSP
1770 AA31 F8             SED ; Bump Step Counter (in decimal)
1780 AA32 A202           LDX #$02
1790 AA34 18      BMPCNT CLC
1800 AA35 A901           LDA #$01
1810 AA37 7D8AAE         ADC STEPCT,X
1820 AA3A 9D8AAE         STA STEPCT,X
1830 AA3D 9003           BCC BMPDON
1840 AA3F CA             DEX
1850 AA40 10F2           BPL BMPCNT
1860 AA42 D8      BMPDON CLD ; Back to Hex
1870 AA43 AE2BA9         LDX OUTSW2 ; Is output shut off for a sub?
1880 AA46 F077           BEQ CQOUT  ; Yes.
1890 AA48 AE28A9         LDX SSTEP  ; Is Single Step on?
1900 AA4B D068           BNE WAITCQ ; Yes.
1910 AA4D A200           LDX #00
1920 AA4F AD00FC LOOP    LDA CON    ; Is there input waiting?
1930 AA52 4A             LSR A
1940 AA53 B005           BCS CNTCHR
1950 AA55 E8             INX
1960 AA56 D0F7           BNE LOOP
1970 AA58 F065           BEQ CQOUT
1980 AA5A AD01FC CNTCHR  LDA CON+1
1990 AA5D 297F           AND #$7F
```

```
2000 AA5F C903    CHKCC   CMP #03     ; Is it a C/C?
2010 AA61 D003            BNE CHKCA   ; No.
2020 AA63 4CFDAE          JMP EXIT    ; Yes, Exit.
2030 AA66 C901    CHKCA   CMP #01     ; Is it C/A?
2040 AA68 D003            BNE CHKCB   ; No.
2050 AA6A 4C00A9          JMP TRACE1
2060 AA6D C902    CHKCB   CMP #02     ; Is it C/B?
2070 AA6F D009            BNE CHKCD   ; No.
2080 AA71 20F0AE          JSR EXTRC   ; Yes.
2090 AA74 2004A9          JSR TRACE2
2100 AA77 4C81AA          JMP EXTRAC
2110 AA7A C904    CHKCD   CMP #04     ; Is it C/D?
2120 AA7C D009            BNE CHKCP   ; No.
2130 AA7E 20F0AE          JSR EXTRC
2140 AA81 20C3AD  EXTRAC  JSR RCOVER
2150 AA84 4C0000  RTMACH  JMP $0000   ; Filled in above, (EXIT VECTOR).
2160 AA87 C910    CHKCP   CMP #16     ; Is it C/P?
2170 AA89 D011            BNE CHKCS   ; No.
2180 AA8B AD2CA9          LDA OUTDEV  ; Yes ...
2190 AA8E 4902            EOR #$02    ;     ... Toggle Printer Byte.
2200 AA90 8D2CA9          STA OUTDEV  ;     ... and Save.
2210 AA93 4A              LSR A
2220 AA94 4A              LSR A       ; Turning Printer On or Off?
2230 AA95 9005            BCC CHKCS   ; Off.
2240 AA97 A90A            LDA #10     ; On ... So output LF to printer.
2250 AA99 2045AD          JSR PRTOUT
2260 AA9C C913    CHKCS   CMP #19     ; Is it a C/S?
2270 AA9E D00C            BNE CHKCQ   ; No.
2280 AAA0 AE89AE          LDX CONTON  ; Is C/S available?
2290 AAA3 F01A            BEQ CQOUT   ; No.
2300 AAA5 A201            LDX #01     ; Yes ...
2310 AAA7 8E28A9          STX SSTEP   ;     ... Record it.
2320 AAAA D013            BNE CQOUT
2330 AAAC AE28A9  CHKCQ   LDX SSTEP   ; 0=off  1=on
2340 AAAF F00E            BEQ CQOUT   ; Single Step not on, no need for C/Q.
2350 AA81 C911            CMP #17     ; Is it a C/Q?
2360 AAB3 F005            BEQ YESCQ   ; Yes.
2370 AAB5 202CAD  WAITCQ  JSR CONIN   ; No ... wait for Console input.
2380 AAB8 D0A5            BNE CHKCC   ;    ... what Char?
2390 AABA A200    YESCQ   LDX #00
2400 AABC 8E28A9          STX SSTEP   ;     ... Record it.
2410 AABF AD24A9  CQOUT   LDA TEMPPC
2420 AAC2 8DDCAA          STA GETINS+1
2430 AAC5 AD25A9          LDA TEMPPC+1
2440 AAC8 8DDDAA          STA GETINS+2
2450 AACB AD0CA9          LDA BOD1
2460 AACE 8DDFAA          STA CHKINS+1
2470 AAD1 AD0DA9          LDA BOD1+01
2480 AAD4 8DE0AA          STA CHKINS+2
2490 AAD7 A000            LDY #00
2500 AAD9 A200            LDX #00
2510 AADB ADFFFF  GETINS  LDA $FFFF ; To be filled in for "artificial" indirect ad
2520 AADE CDFFFF  CHKINS  CMP $FFFF ;
2530 AAE1 F022            BEQ MOVREC
2540 AAE3 EC0FA9          CPX RN1
2550 AAE6 D003            BNE SMORE1
2560 AAE8 4CFDAE          JMP EXIT ; Error
2570 AAEB E8      SMORE1  INX
2580 AAEC 48              PHA
2590 AAED AD0EA9          LDA RL1
2600 AAF0 18              CLC
2610 AAF1 6DDFAA          ADC CHKINS+1
2620 AAF4 8DDFAA          STA CHKINS+1
2630 AAF7 9008            BCC MORE1
2640 AAF9 EEE0AA          INC CHKINS+2
2650 AAFC D003            BNE MORE1
2660 AAFE 4CFDAE          JMP EXIT ; Error
2670 AB01 68      MORE1   PLA
2680 AB02 18              CLC
2690 AB03 90D9            BCC CHKINS
2700 AB05 ADDFAA  MOVREC  LDA CHKINS+1
2710 AB08 8D14AB          STA FLNDX1+1
2720 AB0B ADE0AA          LDA CHKINS+2
2730 AB0E 8D15AB          STA FLNDX1+2
2740 AB11 A000            LDY #00
2750 AB13 B9FFFF  FLNDX1  LDA $FFFF,Y ; To be filled in for "artificial" indirect
2760 AB16 9931A9          STA TMPREC,Y
2770 AB19 C8              INY
2780 AB1A CC0EA9          CPY RL1
2790 AB1D D0F4            BNE FLNDX1
2800 AB1F A9EA            LDA #$EA    ; NOP
2810 AB21 8DA8AB          STA WKAREA
2820 AB24 8DACAB          STA WKAREA+1
2830 AB27 8DADAB          STA WKAREA+2
2840 AB2A AD35A9          LDA TMPREC+4  ; Determine address of next instruction
2850 AB2D 18              CLC
2860 AB2E 6D24A9          ADC TEMPPC
2870 AB31 8D24A9          STA TEMPPC
2880 AB34 9003            BCC MOVINS
2890 AB36 EE25A9          INC TEMPPC+1
2900 AB39 ADDCAA  MOVINS  LDA GETINS+1
2910 AB3C 8D48AB          STA INS+1
2920 AB3F ADDDAA          LDA GETINS+2
2930 AB42 8D49AB          STA INS+2
2940 AB45 A000            LDY #00
2950 AB47 B9FFFF  INS     LDA $FFFF,Y ;
2960 AB4A 99ABAB          STA WKAREA,Y
2970 AB4D C8              INY
2980 AB4E CC35A9          CPY TMPREC+4
2990 AB51 D0F4            BNE INS
3000 AB53 AD10A9  SRCH2   LDA BOD2
3010 AB56 8D65AB          STA CPTB2+1
```

```
3020 AB59 AD11A9          LDA BOD2+01
3030 AB5C 8D66AB          STA CPTB2+2
3040 AB5F A200            LDX #00
3050 AB61 ADABAB CHKNON   LDA WKAREA
3060 AB64 CDFFFF CPTB2    CMP $FFFF ,
3070 AB67 F01A            BEQ NONSEQ
3080 AB69 E8              INX
3090 AB6A EC13A9          CPX RN2
3100 AB6D F039            BEQ XQT ; Not non-sequential
3110 AB6F AD12A9          LDA RL2
3120 AB72 18              CLC
3130 AB73 6D65AB          ADC CPTB2+1
3140 AB76 8D65AB          STA CPTB2+1
3150 AB79 90E6            BCC CHKNON
3160 AB7B EE66AB          INC CPTB2+2
3170 AB7E D0E1            BNE CHKNON
3180 AB80 4CFDAE          JMP EXIT ; Error
3190 AB83 AD65AB NONSEQ   LDA CPTB2+1
3200 AB86 8D92AB          STA NONS+1
3210 AB89 AD66AB          LDA CPTB2+2
3220 AB8C 8D93AB          STA NONS+2
3230 AB8F A001            LDY #01
3240 AB91 B9FFFF NONS     LDA $FFFF,Y ;
3250 AB94 99A5AB          STA SIMXQT,Y
3260 AB97 C8              INY
3270 AB98 C003            CPY #03
3280 AB9A D0F5            BNE NONS
3290 AB9C A901            LDA #01
3300 AB9E 8D29A9          STA NSEQSP
3310 ABA1 20C3AD          JSR RCOVER
3320 ABA4 D8              CLD
3330 ABA5 4C0000 SIMXQT   JMP $0000      ; Non-Sequential Instruction
3340 ABA9 20C3AD XQT      JSR RCOVER
3350 ABA8 EA     WKAREA   .BYTE $EA,$EA,$EA
3350 ABAC EA
3350 ABAD EA
3360 ABAE 20AFAD          JSR PRSRVE
3370                      ;
3380 ABB1 20EBAE REPORT   JSR SETCC
3390 ABB4 8D26A9          STA LINECT
3400 ABB7 A924            LDA #'$
3410 ABB9 209FAD          JSR CHROUT
3420 ABBC ADDDAA          LDA GETINS+2
3430 ABBF 8D38AC          STA CODE+2
3440 ABC2 8D87AE          STA SCHSTR+1
3450 ABC5 2088AD          JSR HEXOUT
3460 ABC8 ADDCAA          LDA GETINS+1
3470 ABCB 8D37AC          STA CODE+1
3480 ABCE 8D86AE          STA SCHSTR
3490 ABD1 2088AD          JSR HEXOUT
3500 ABD4 AC2AA9 ENSRCH   LDY OUTSW1   ; End Address Search
3510 ABD7 D018            BNE TAB06    ; Output already on.
3520 ABD9 AC86AE          LDY SCHSTR
3530 ABDC CC84AE          CPY ENADDR
3540 ABDF D013            BNE TAB06
3550 ABE1 AC87AE          LDY SCHSTR+1
3560 ABE4 CC85AE          CPY ENADDR+1
3570 ABE7 D00B            BNE TAB06
3580 ABE9 A001            LDY #01
3590 ABEB 8C2AA9          STY OUTSW1   ; Turn output on, right now!
3600 ABEE 8C2BA9          STY OUTSW2
3610 ABF1 8C89AE          STY CONTON ; Turn on Control Chars
3620 ABF4 A206   TAB06    LDX #06
3630 ABF6 20D2AD          JSR TAB
3640 ABF9 AC2AA9 CDSRCH   LDY OUTSW1 ; Code Search
3650 ABFC F036            BEQ CODE:  ; Output already turned off.
3660 ABFE 208DAE          JSR SRCH04
3670 AC01 AC88AE          LDY FOUND:
3680 AC04 F030            BEQ CODE   ; Not found
3690 AC06 ADC4AE          LDA SCH04B+1
3700 AC09 8D15AC          STA SCH04C+1
3710 AC0C ADC5AE          LDA SCH04B+2
3720 AC0F 8D16AC          STA SCH04C+2
3730 AC12 A002            LDY #02
3740 AC14 B9FFFF SCH04C   LDA $FFFF,Y
3750 AC17 9982AE          STA ENADDR-2,Y
3760 AC1A C8              INY
3770 AC1B C004            CPY #04
3780 AC1D D0F5            BNE SCH04C
3790 AC1F AC84AE          LDY ENADDR
3800 AC22 D005            BNE SHUTOF
3810 AC24 AC85AE          LDY ENADDR+1
3820 AC27 F019            BEQ TAB14
3830 AC29 A000   SHUTOF   LDY #00
3840 AC2B 8C2BA9          STY OUTSW2 ; Shut off output after line is printed.
3850 AC2E 8C89AE          STY CONTON ; Shut off Control Chars
3860 AC31 4C42AC          JMP TAB14
3870 AC34 A000   CODE:    LDY #00
3880 AC36 B9FFFF CODE     LDA $FFFF,Y ;
3890 AC39 2088AD          JSR HEXOUT
3900 AC3C C8              INY
3910 AC3D CC35A9          CPY TMPREC+4
3920 AC40 D0F4            BNE CODE
3930 AC42 A20E   TAB14    LDX #14
3940 AC44 20D2AD          JSR TAB
3950 AC47 A001            LDY #01
3960 AC49 B931A9 MNEOUT   LDA TMPREC,Y
3970 AC4C 209FAD          JSR CHROUT
3980 AC4F C8              INY
3990 AC50 C004            CPY #04
4000 AC52 D0F5            BNE MNEOUT
4010 AC54 A214            LDX #20
4020 AC56 20D2AD          JSR TAB
```

```
4030 AC59 AD14A9          LDA BOD3
4040 AC5C 8D82AC          STA FLNDX3+1
4050 AC5F AD15A9          LDA BOD3+01
4060 AC62 8D83AC          STA FLNDX3+2
4070 AC65 AE36A9 ADV3     LDX TMPREC+5
4080 AC68 CA              DEX
4090 AC69 F014            BEQ FND3
4100 AC6B AD16A9          LDA RL3
4110 AC6E 18              CLC
4120 AC6F 6D82AC          ADC FLNDX3+1
4130 AC72 8D82AC          STA FLNDX3+1
4140 AC75 90F1            BCC ADV3+03
4150 AC77 EE83AC          INC FLNDX3+2
4160 AC7A D0EC            BNE ADV3+03
4170 AC7C 4CFDAE          JMP EXIT ; Error
4180 AC7F A000   FND3     LDY #00
4190 AC81 B9FFFF FLNDX3   LDA $FFFF,Y ;
4200 AC84 C8              INY
4210 AC85 998CAC          STA TBL3,Y
4220 AC88 C002            CPY #02
4230 AC8A D0F5            BNE FLNDX3
4240 AC8C 200000 TBL3     JSR $0000    ; To be filled in later
4250 AC8F A000            LDY #00
4260 AC91 A227            LDX #39
4270 AC93 20D2AD          JSR TAB
4280 AC96 B91CA9 REGOUT   LDA TEMPA,Y
4290 AC99 2088AD          JSR HEXOUT
4300 AC9C A901            LDA #01
4310 AC9E 18              CLC
4320 AC9F 6D26A9          ADC LINECT
4330 ACA2 AA              TAX
4340 ACA3 20D2AD          JSR TAB
4350 ACA6 C8              INY
4360 ACA7 C003            CPY #03
4370 ACA9 D0EB            BNE REGOUT
4380 ACAB A000   POUT      LDY #00
4390 ACAD AD1FA9          LDA TEMPP
4400 ACB0 0A     SHIFT    ASL A
4410 ACB1 8D23A9          STA TEMPA2
4420 ACB4 900C            BCC P0
4430 ACB6 C002            CPY #02 ; Third bit always 1
4440 ACB8 D004            BNE P1
4450 ACBA A978            LDA #'x
4460 ACBC D006            BNE OUTP
4470 ACBE A931   P1        LDA #'1
4480 ACC0 D002            BNE OUTP
4490 ACC2 A930   P0        LDA #'0
4500 ACC4 209FAD OUTP     JSR CHROUT
4510 ACC7 C007            CPY #07
4520 ACC9 F00D            BEQ STKOUT
4530 ACCB A92D            LDA #'-
4540 ACCD 209FAD          JSR CHROUT
4550 ACD0 AD23A9          LDA TEMPA2
4560 ACD3 C8              INY
4570 ACD4 C008            CPY #08
4580 ACD6 D0D8            BNE SHIFT
4590 ACD8 A902   STKOUT    LDA #02
4600 ACDA 18              CLC
4610 ACDB 6D26A9          ADC LINECT
4620 ACDE AA              TAX
4630 ACDF 20D2AD          JSR TAB
4640 ACE2 BA              TSX
4650 ACE3 8A              TXA
4660 ACE4 2088AD          JSR HEXOUT
4670 ACE7 AD2AA9          LDA OUTSW1
4680 ACEA F00B            BEQ SWSWAP
4690 ACEC 2061AD          JSR LNCRLF
4700 ACEF AD29A9          LDA NSEQSP
4710 ACF2 F003            BEQ SWSWAP
4720 ACF4 2041AE          JSR HEADR2
4730 ACF7 AD2BA9 SWSWAP   LDA OUTSW2
4740 ACFA CD2AA9          CMP OUTSW1
4750 ACFD F006            BEQ GONEXT
4760 ACFF 2061AD          JSR LNCRLF
4770 AD02 8D2AA9          STA OUTSW1
4780 AD05 4C2CAA GONEXT   JMP NEXT
4790              ;
4800              ;
4810              ;              Subroutines
4820              ;-----------------------------------------
4830              ;
4840 AD08 202CAD HEXIN    JSR CONIN
4850 AD0B 8D23A9          STA TEMPA2
4860 AD0E C930            CMP #'0
4870 AD10 30F6            BMI HEXIN
4880 AD12 C93A            CMP #':
4890 AD14 300B            BMI YESHEX
4900 AD16 C941            CMP #'A
4910 AD18 30EE            BMI HEXIN
4920 AD1A C947            CMP #'G
4930 AD1C 10EA            BPL HEXIN
4940 AD1E 18              CLC
4950 AD1F E906            SBC #06
4960 AD21 290F   YESHEX   AND #$0F   ; Mask to Low four bits
4970 AD23 48              PHA
4980 AD24 AD23A9          LDA TEMPA2
4990 AD27 2039AD          JSR CONOUT
5000 AD2A 68              PLA
5010 AD2B 60              RTS
5020              ;
5030 AD2C AD00FC CONIN    LDA CON
5040 AD2F 4A              LSR A
```

```
5050 AD30 90FA           BCC CONIN
5060 AD32 AD01FC         LDA CON+1
5070 AD35 297F           AND #$7F   ; Mask to 7 bits
5080 AD37 60             RTS
5090                     ;
5100 AD38 48     CONOUT  PHA ; Serial (ACIA) Console Output
5110 AD39 AD00FC         LDA CON
5120 AD3C 4A             LSR A
5130 AD3D 4A             LSR A
5140 AD3E 90F9           BCC CONOUT+1
5150 AD40 68             PLA
5160 AD41 8D01FC         STA CON+1
5170 AD44 60             RTS
5180                     ;
5190 AD45 48     PRTOUT  PHA ; Parallel Printer Output
5200 AD46 AD00F4         LDA PRNTR
5210 AD49 4A             LSR A
5220 AD4A B0FA           BCS PRTOUT+1
5230 AD4C 68             PLA
5240 AD4D 8D02F4         STA PRNTR+2
5250 AD50 2C20F4         BIT PRNTR+$20
5260 AD53 60             RTS
5270                     ;
5280 AD54 48     CRLF    PHA
5290 AD55 A90D           LDA #13
5300 AD57 2038AD         JSR CONOUT
5310 AD5A A90A           LDA #10
5320 AD5C 2038AD         JSR CONOUT
5330 AD5F 68             PLA
5340 AD60 60             RTS
5350                     ;
5360 AD61 48     LNCRLF  PHA
5370 AD62 A90D           LDA #13
5380 AD64 209FAD         JSR CHROUT
5390 AD67 A90A           LDA #10
5400 AD69 209FAD         JSR CHROUT
5410 AD6C 20E0AD         JSR LINOUT
5420 AD6F A900           LDA #00
5430 AD71 8D26A9         STA LINECT
5440 AD74 68             PLA
5450 AD75 60             RTS
5460                     ;
5470 AD76 2008AD GETHEX  JSR HEXIN
5480 AD79 0A             ASL A
5490 AD7A 0A             ASL A
5500 AD7B 0A             ASL A
5510 AD7C 0A             ASL A
5520 AD7D 8D20A9         STA TEMPA1
5530 AD80 2008AD         JSR HEXIN
5540 AD83 18             CLC
5550 AD84 6D20A9         ADC TEMPA1
5560 AD87 60             RTS
5570                     ;
5580 AD88 48     HEXOUT  PHA
5590 AD89 29F0           AND #$F0
5600 AD8B 4A             LSR A
5610 AD8C 4A             LSR A
5620 AD8D 4A             LSR A
5630 AD8E 4A             LSR A
5640 AD8F 2095AD         JSR HXOUT
5650 AD92 68             PLA
5660 AD93 290F           AND #$0F  ; Mask to four Low bits
5670 AD95 18     HXOUT   CLC
5680 AD96 6930           ADC #'0
5690 AD98 C93A           CMP #':
5700 AD9A 3003           BMI CHROUT
5710 AD9C 18             CLC
5720 AD9D 6907           ADC #07 ; Difference between 'A' and '9'
5730 AD9F 8E21A9 CHROUT  STX TEMPX1
5740 ADA2 AE26A9         LDX LINECT
5750 ADA5 9D2FB6         STA LINE,X
5760 ADA8 AE21A9         LDX TEMPX1
5770 ADAB EE26A9         INC LINECT
5780 ADAE 60             RTS
5790                     ;
5800 ADAF 08     PRSRVE  PHP
5810 ADB0 8D1CA9         STA TEMPA
5820 ADB3 8E1DA9         STX TEMPX
5830 ADB6 8C1EA9         STY TEMPY
5840 ADB9 68             PLA
5850 ADBA 48             PHA
5860 ADBB 8D1FA9         STA TEMPP
5870 ADBE 28             PLP
5880 ADBF 20EBAE         JSR SETCC
5890 ADC2 60             RTS
5900                     ;
5910 ADC3 AD1FA9 RCOVER  LDA TEMPP
5920 ADC6 48             PHA
5930 ADC7 AD1CA9         LDA TEMPA
5940 ADCA AE1DA9         LDX TEMPX
5950 ADCD AC1EA9         LDY TEMPY
5960 ADD0 28             PLP
5970 ADD1 60             RTS
5980                     ;
5990 ADD2 EC26A9 TAB     CPX LINECT ; Tab in X
6000 ADD5 3008           BMI TBDONE
6010 ADD7 AD27A9         LDA TABCHR
6020 ADDA 209FAD         JSR CHROUT
6030 ADDD 10F3           BPL TAB
6040 ADDF 60     TBDONE  RTS
6050                     ;
```

```
6060 ADE0 A200   LINOUT  LDX #00
6070 ADE2 BD2FB6         LDA LINE,X
6080 ADE5 AC2CA9         LDY OUTDEV
6090 ADE8 F007           BEQ CNONLY
6100 ADEA 2045AD         JSR PRTOUT
6110 ADED C001           CPY #$01
6120 ADEF F003           BEQ NXTCHR
6130 ADF1 2038AD CNONLY  JSR CONOUT
6140 ADF4 E8     NXTCHR  INX
6150 ADF5 EC26A9         CPX LINECT
6160 ADF8 D0E8           BNE LINOUT+02
6170 ADFA 60             RTS
6180             ;
6190 ADFB 8C22A9 EFADR1  STY TEMPY1 ; Skip SRCH04
6200 ADFE AD1BA9         LDA RN4
6210 AE01 8D20A9         STA TEMPA1
6220 AE04 A900           LDA #$00
6230 AE06 8D1BA9         STA RN4
6240 AE09 201BAE         JSR EFFADR+05
6250 AE0C AD20A9         LDA TEMPA1
6260 AE0F 8D1BA9         STA RN4
6270 AE12 AC22A9         LDY TEMPY1
6280 AE15 60             RTS
6290             ;
6300 AE16 A21F   EFFADR  LDX #31
6310 AE18 20D2AD         JSR TAB
6320 AE1B ADACAB         LDA WKAREA+01
6330 AE1E 8D86AE         STA SCHSTR
6340 AE21 ADADAB         LDA WKAREA+02
6350 AE24 8D87AE         STA SCHSTR+01
6360 AE27 208DAE         JSR SRCH04
6370 AE2A AD88AE         LDA FOUND:
6380 AE2D D011           BNE EFFDON
6390 AE2F A924           LDA #'$
6400 AE31 209FAD         JSR CHROUT
6410 AE34 ADADAB         LDA WKAREA+02
6420 AE37 2088AD         JSR HEXOUT
6430 AE3A ADACAB         LDA WKAREA+01
6440 AE3D 2088AD         JSR HEXOUT
6450 AE40 60     EFFDON  RTS
6460             ;
6470 AE41 A200   HEADR2  LDX #00
6480 AE43 8E26A9         STX LINECT
6490 AE46 AD27A9         LDA TABCHR
6500 AE49 48             PHA
6510 AE4A A92D           LDA #'-
6520 AE4C 8D27A9         STA TABCHR
6530 AE4F A210           LDX #16
6540 AE51 20D2AD         JSR TAB
6550 AE54 A200           LDX #$00
6560 AE56 BD8AAE CNTOUT  LDA STEPCT,X ; Print out current Step Count
6570 AE59 2088AD         JSR HEXOUT
6580 AE5C E8             INX
6590 AE5D E003           CPX #$03
6600 AE5F D0F5           BNE CNTOUT
6610 AE61 A226           LDX #38
6620 AE63 20D2AD         JSR TAB
6630 AE66 68             PLA
6640 AE67 8D27A9         STA TABCHR
6650 AE6A E8             INX
6660 AE6B 4C73AE         JMP HEADER+05
6670             ;
6680 AE6E A200   HEADER  LDX #00
6690 AE70 8E26A9         STX LINECT
6700 AE73 BDE6B5         LDA MSG4,X
6710 AE76 C9FF           CMP #$FF
6720 AE78 F006           BEQ HEADDN
6730 AE7A 209FAD         JSR CHROUT
6740 AE7D E8             INX
6750 AE7E D0F3           BNE HEADER+5
6760 AE80 2061AD HEADDN  JSR LNCRLF
6770 AE83 60             RTS
6780             ;
6790 AE84 0000   ENADDR  .DBYTE 00
6800 AE86 0000   SCHSTR  .DBYTE 00
6810 AE88 00     FOUND:  .BYTE 00 ; 0 = Not Found, 1 = Found
6820 AE89 01     CONTON  .BYTE 01 ; 0 = Off, 1 = Check for Control Char
6830 AE8A 00     STEPCT  .BYTE 00,00,00 ; Step Counter
6830 AE8B 00
6830 AE8C 00
6840             ;
6850 AE8D A000   SRCH04  LDY #00
6860 AE8F 8C89AE         STY FOUND:
6870 AE92 AD1BA9         LDA RN4
6880 AE95 F053           BEQ DON04A
6890 AE97 AD18A9         LDA BOD4
6900 AE9A 8DA9AE         STA SCH04A+01
6910 AE9D AD19A9         LDA BOD4+1
6920 AEA0 8DAAAE         STA SCH04A+02
6930 AEA3 A201   GTADDR  LDX #01
6940 AEA5 BD86AE         LDA SCHSTR,X
6950 AEA8 DDFFFF SCH04A  CMP $FFFF,X
6960 AEAB D025           BNE NXT04A
6970 AEAD CA             DEX
6980 AEAE F0F5           BEQ GTADDR+02
6990 AEB0 A901   YESFND  LDA #01
7000 AEB2 8D88AE         STA FOUND:
7010 AEB5 ADA9AE         LDA SCH04A+1
7020 AEB8 8DC4AE         STA SCH04B+1
7030 AEBB ADAAAE         LDA SCH04A+02
7040 AEBE 8DC5AE         STA SCH04B+02
7050 AEC1 A204           LDX #04
7060 AEC3 BDFFFF SCH04B  LDA $FFFF,X
```

```
7070 AEC6 209FAD          JSR CHROUT
7080 AEC9 E8              INX
7090 AECA EC1AA9          CPX RL4
7100 AECD F01B            BEQ DON04A
7110 AECF 4CC3AE          JMP SCH04B
7120 AED2 CC1BA9 NXT04A   CPY RN4
7130 AED5 F013            BEQ DON04A
7140 AED7 18              CLC
7150 AED8 AD1AA9          LDA RL4
7160 AEDB 6DA9AE          ADC SCH04A+01
7170 AEDE 8DA9AE          STA SCH04A+01
7180 AEE1 9003            BCC *+05
7190 AEE3 EEAAAE          INC SCH04A+02
7200 AEE6 C8              INY
7210 AEE7 4CA3AE          JMP GTADDR
7220 AEEA 60      DON04A  RTS
7230                      ;
7240 AEEB A900    SETCC   LDA #$00 ; Clear CC Reg. for TRACE1 use
7250 AEED 48              PHA
7260 AEEE 28              PLP
7270 AEEF 60              RTS
7290 AEF0 AD24A9 EXTRC    LDA TEMPPC
7300 AEF3 8D85AA          STA RTMACH+1
7310 AEF6 AD25A9          LDA TEMPPC+1
7320 AEF9 8D86AA          STA RTMACH+2
7330 AEFC 60              RTS
7340                      ;
7350 AEFD 6CFCFF EXIT     JMP ($FFFC) ; BOOT-UP Vector.
7360                      ;
7370                      ;================================================
7380                      ;                   TABLE3
7390                      ;
7400 AF00 1AAF    TABLE3  .WORD TYPE01
7410 AF02 28AF            .WORD TYPE02
7420 AF04 32AF            .WORD TYPE03
7430 AF06 3EAF            .WORD TYPE04
7440 AF08 43AF            .WORD TYPE05
7450 AF0A 44AF            .WORD TYPE06
7460 AF0C 76AF            .WORD TYPE07
7470 AF0E ADAF            .WORD TYPE08
7480 AF10 E2AF            .WORD TYPE09
7490 AF12 E6AF            .WORD TYPE10
7500 AF14 1580            .WORD TYPE11
7510 AF16 1C80            .WORD TYPE12
7520 AF18 81AF            .WORD TYPE13
7530                      ;
7540                      ;================================================
7550                      ;
7560 AF1A A923    TYPE01  LDA #'#   ; Immediate
7570 AF1C 209FAD          JSR CHROUT
7580 AF1F A924            LDA #'$
7590 AF21 209FAD          JSR CHROUT
7600 AF24 ADACAB          LDA WKAREA+1
7610 AF27 2088AD          JSR HEXOUT
7620 AF2A 60              RTS
7630                      ;
7640 AF2B 20FBAD TYPE02   JSR EFADR1   ; Absolute
7650 AF2E 2016AE          JSR EFFADR
7660 AF31 60              RTS
7670                      ;
7680 AF32 201FAF TYPE03   JSR TYPE01+5   ; Zero Page
7690 AF35 A900            LDA #00
7700 AF37 8DADAB          STA WKAREA+2
7710 AF3A 2016AE          JSR EFFADR
7720 AF3D 60              RTS
7730                      ;
7740 AF3E A941    TYPE04  LDA #'A   ; Accumulator
7750 AF40 209FAD          JSR CHROUT
7760 AF43 60      TYPE05  RTS       ; Implied
7770                      ;
7780 AF44 A928    TYPE06  LDA #'(   ; (Ind,X)
7790 AF46 201CAF          JSR TYPE01+02
7800 AF49 A92C            LDA #',
7810 AF4B 209FAD          JSR CHROUT
7820 AF4E A958            LDA #'X
7830 AF50 209FAD          JSR CHROUT
7840 AF53 A929            LDA #')
7850 AF55 209FAD          JSR CHROUT
7860 AF58 AD1DA9          LDA TEMPX
7870 AF5B 18              CLC
7880 AF5C 6DACAB          ADC WKAREA+1
7890 AF5F 8D69AF          STA INDXL+1
7900 AF62 8D6EAF          STA INDXH+1
7910 AF65 EE6EAF          INC INDXH+1
7920 AF69 A500    INDXL   LDA $00  ; Will contain indirect PZ address
7930 AF6A 8DACAB          STA WKAREA+1
7940 AF6D A500    INDXH   LDA $00
7950 AF6F 8DADAB          STA WKAREA+2
7960 AF72 2016AE          JSR EFFADR
7970 AF75 60              RTS
7980                      ;
7990 AF76 A928    TYPE07  LDA #'(   ; (Ind),Y
8000 AF78 201CAF          JSR TYPE01+02
8010 AF7B A929            LDA #')
8020 AF7D 209FAD          JSR CHROUT
8030 AF80 A92C            LDA #',
8040 AF82 209FAD          JSR CHROUT
8050 AF85 A959            LDA #'Y
8060 AF87 209FAD          JSR CHROUT
8070 AF8A ADACAB          LDA WKAREA+1
8080 AF8D 8D97AF          STA INDYH+1
8090 AF90 8D9CAF          STA INDYL+1
8100 AF93 EE97AF          INC INDYH+1
```

```
8110 AF96 A500   INDYH   LDA $00
8120 AF98 8DADAB         STA WKAREA+02
8130 AF9B A500   INDYL   LDA $00
8140 AF9D 18             CLC
8150 AF9E 6D1EA9         ADC TEMPY
8160 AFA1 8DACAB         STA WKAREA+1
8170 AFA4 9003           BCC *+05
8180 AFA6 EEADAB         INC WKAREA+2
8190 AFA9 2016AE         JSR EFFADR
8200 AFAC 60             RTS
8210             ;
8220 AFAD A000   TYPE08  LDY #00         ; (Zero Page,X)
8230 AFAF F002           BEQ *+04
8240 AFB1 A001   TYPE13  LDY #01         ; (Zero Page,Y)
8250 AFB3 A924           LDA #'$
8260 AFB5 2021AF         JSR TYPE01+07
8270 AFB8 A92C           LDA #',
8280 AFBA 209FAD         JSR CHROUT
8290 AFBD A958           LDA #'X
8300 AFBF C000           CPY #00
8310 AFC1 F002           BEQ *+04
8320 AFC3 A959           LDA #'Y
8330 AFC5 209FAD         JSR CHROUT
8340 AFC8 AD1DA9         LDA TEMPX
8350 AFCB C000           CPY #00
8360 AFCD F003           BEQ *+05
8370 AFCF AD1EA9         LDA TEMPY
8380 AFD2 18             CLC
8390 AFD3 6DACAB         ADC WKAREA+01
8400 AFD6 8DACAB         STA WKAREA+01
8410 AFD9 A900           LDA #00
8420 AFDB 8DADAB         STA WKAREA+02
8430 AFDE 2016AE         JSR EFFADR
8440 AFE1 60             RTS
8450             ;
8460 AFE2 A000   TYPE09  LDY #00    ; Absolute,X
8470 AFE4 F002           BEQ ABSOUT
8480 AFE6 A001   TYPE10  LDY #01    ; Absolute,Y
8490 AFE8 20FBAD ABSOUT  JSR EFADR1
8500 AFEB A92C           LDA #',
8510 AFED 209FAD         JSR CHROUT
8520 AFF0 A958           LDA #'X
8530 AFF2 C000           CPY #00
8540 AFF4 F002           BEQ XYOUT
8550 AFF6 A959           LDA #'Y
8560 AFF8 209FAD XYOUT   JSR CHROUT
8570 AFFB AD1DA9         LDA TEMPX
8580 AFFE C000           CPY #00
8590 B000 F003           BEQ XYADD
8600 B002 AD1EA9         LDA TEMPY
8610 B005 18     XYADD   CLC
8620 B006 6DACAB         ADC WKAREA+01
8630 B009 8DACAB         STA WKAREA+01
8640 B00C 9003           BCC ABDONE
8650 B00E EEADAB         INC WKAREA+02
8660 B011 2016AE ABDONE  JSR EFFADR
8670 B014 60             RTS
8680             ;
8690 B015 20FBAD TYPE11  JSR EFADR1 ; Relative
8700 B018 2016AE         JSR EFFADR
8710 B01B 60             RTS
8720             ;
8730 B01C A928   TYPE12  LDA #'(          ; Indirect
8740 B01E 209FAD         JSR CHROUT
8750 B021 20FBAD         JSR EFADR1
8760 B024 A929           LDA #')
8770 B026 209FAD         JSR CHROUT
8780 B029 AD24A9         LDA TEMPPC
8790 B02C 8DACAB         STA WKAREA+1
8800 B02F AD25A9         LDA TEMPPC+1
8810 B032 8DADAB         STA WKAREA+2
8820 B035 2016AE         JSR EFFADR
8830 B038 60             RTS
8840             ;
8850             ;        TABLE2
8860             ;
8870             ;  This is a list of all non-sequential
8880             ; 6502 commands, and their interpreter
8890             ; addresses.
8900             ;
8910             ;
8920             ;================================================
8930             ;
8940             ;         Fields:
8950             ;            (1) OP CODE
8960             ;            (2) ADDRESS OF SIMULATED EXECUTION
8970             ;
8980             ;================================================
8990 B039 00     TABLE2 .BYTE $00     ; BRK
9000 B03A 3AB1          .WORD XQTBRK
9010 B03C 10            .BYTE $10     ; BPL
9020 B03D 63B0          .WORD XQTBPL
9030 B03F 20            .BYTE $20     ; JSR
9040 B040 0AB1          .WORD XQTJSR
9050 B042 30            .BYTE $30     ; BMI
9060 B043 68B0          .WORD XQTBMI
9070 B045 40            .BYTE $40     ; RTI
9080 B046 51B1          .WORD XQTRTI
9090 B048 4C            .BYTE $4C     ; JMP Absolute
9100 B049 D2B0          .WORD XQTJPA
9110 B04B 50            .BYTE $50     ; BVC
9120 B04C 6DB0          .WORD XQTBVC
```

```
 9130 B04E 60            .BYTE $60      ; RTS
 9140 B04F 27B1          .WORD XQTRET
 9150 B051 6C            .BYTE $6C      ; JMP Indirect
 9160 B052 E1B0          .WORD XQTJPI
 9170 B054 70            .BYTE $70      ; BVS
 9180 B055 72B0          .WORD XQTBVS
 9190 B057 90            .BYTE $90      ; BCC
 9200 B058 77B0          .WORD XQTBCC
 9210 B05A B0            .BYTE $B0      ; BCS
 9220 B05B 7CB0          .WORD XQTBCS
 9230 B05D D0            .BYTE $D0      ; BNE
 9240 B05E 81B0          .WORD XQTBNE
 9250 B060 F0            .BYTE $F0      ; BEQ
 9260 B061 86B0          .WORD XQTBEQ
 9270                ;
 9280                ;=======================================
 9290                ;
 9300 B063 102A    XQTBPL   BPL YBRNCH
 9310 B065 4C88B0           JMP NBRNCH
 9320 B068 3025    XQTBMI   BMI YBRNCH
 9330 B06A 4C88B0           JMP NBRNCH
 9340 B06D 5020    XQTBVC   BVC YBRNCH
 9350 B06F 4C88B0           JMP NBRNCH
 9360 B072 701B    XQTBVS   BVS YBRNCH
 9370 B074 4C88B0           JMP NBRNCH
 9380 B077 9016    XQTBCC   BCC YBRNCH
 9390 B079 4C88B0           JMP NBRNCH
 9400 B07C B011    XQTBCS   BCS YBRNCH
 9410 B07E 4C88B0           JMP NBRNCH
 9420 B081 D00C    XQTBNE   BNE YBRNCH
 9430 B083 4C88B0           JMP NBRNCH
 9440 B086 F007    XQTBEQ   BEQ YBRNCH
 9450 B088 A000    NBRNCH   LDY #00
 9460 B08A 8C29A9           STY NSEQSP
 9470 B08D F002             BEQ BRCALC
 9480 B08F A001    YBRNCH   LDY #01
 9490 B091 AD25A9 BRCALC    LDA TEMPPC+1
 9500 B094 8DADAB           STA WKAREA+2
 9510 B097 ADACAB           LDA WKAREA+1
 9520 B09A 18               CLC
 9530 B09B 0A               ASL A
 9540 B09C 9012             BCC BRFOR ; Forward or backward branch?
 9550 B09E ADACAB BRBAC     LDA WKAREA+1
 9560 B0A1 18               CLC
 9570 B0A2 6D24A9           ADC TEMPPC
 9580 B0A5 8DACAB           STA WKAREA+1
 9590 B0A8 B015             BCS BRDONE
 9600 B0AA CEADAB           DEC WKAREA+2
 9610 B0AD 4CBFB0           JMP BRDONE
 9620 B0B0 AD24A9 BRFOR     LDA TEMPPC
 9630 B0B3 18               CLC
 9640 B0B4 6DACAB           ADC WKAREA+1
 9650 B0B7 8DACAB           STA WKAREA+1
 9660 B0BA 9003             BCC BRDONE
 9670 B0BC EEADAB           INC WKAREA+2
 9680 B0BF C000    BRDONE   CPY #00
 9690 B0C1 F00C             BEQ DONEBR
 9700 B0C3 ADACAB           LDA WKAREA+1
 9710 B0C6 8D24A9           STA TEMPPC
 9720 B0C9 ADADAB           LDA WKAREA+2
 9730 B0CC 8D25A9           STA TEMPPC+1
 9740 B0CF 4CB1AB DONEBR    JMP REPORT
 9750                ;
 9760 B0D2 ADACAB XQTJPA    LDA WKAREA+01
 9770 B0D5 8D24A9           STA TEMPPC
 9780 B0D8 ADADAB           LDA WKAREA+02
 9790 B0DB 8D25A9           STA TEMPPC+01
 9800 B0DE 4CB1AB           JMP REPORT
 9810 B0E1 ADACAB XQTJPI    LDA WKAREA+01
 9820 B0E4 8DFCB0           STA LOBITE+01
 9830 B0E7 8D02B1           STA HIBITE+01
 9840 B0EA ADADAB           LDA WKAREA+02
 9850 B0ED 8DFDB0           STA LOBITE+02
 9860 B0F0 8D03B1           STA HIBITE+02
 9870 B0F3 EE02B1           INC HIBITE+01
 9880 B0F6 9003 D003    BNE BEQ LOBITE    ; 1.01, was BCC
 9890 B0F8 EE03B1           INC HIBITE+02
 9900 B0FB ADFFFF LOBITE    LDA $FFFF      ; To be filled in later
 9910 B0FE 8D24A9           STA TEMPPC
 9920 B101 ADFFFF HIBITE    LDA $FFFF      ; To be filled in later
 9930 B104 8D25A9           STA TEMPPC+01
 9940 B107 4CB1AB           JMP REPORT
 9950 B10A AC25A9 XQTJSR    LDY TEMPPC+1
 9960 B10D AE24A9           LDX TEMPPC
 9970 B110 D001             BNE DCLOW
 9980 B112 88               DEY
 9990 B113 CA       DCLOW   DEX
10000 B114 98               TYA
10010 B115 48               PHA
10020 B116 8A               TXA
10030 B117 48               PHA
10040 B118 ADACAB           LDA WKAREA+1
10050 B11B 8D24A9           STA TEMPPC
10060 B11E ADADAB           LDA WKAREA+2
10070 B121 8D25A9           STA TEMPPC+1
10080 B124 4CB1AB           JMP REPORT
10090 B127 68       XQTRET  PLA
10100 B128 8D24A9           STA TEMPPC
10110 B12B 68               PLA
10120 B12C 8D25A9           STA TEMPPC+1
10130 B12F EE24A9           INC TEMPPC
```

```
10140 B132 D003           BNE RETDON
10150 B134 EE25A9         INC TEMPPC+1
10160 B137 4CB1AB RETDON  JMP REPORT
10170 B13A ADFEFF XQTBRK  LDA $FFFE
10180 B13D 8D24A9         STA TEMPPC
10190 B140 ADFFFF         LDA $FFFF
10200 B143 8D25A9         STA TEMPPC+1
10210 B146 AD1FA9         LDA TEMPP
10220 B149 0914           ORA #%00010100 ; Set B and I flags
10230 B14B 8D1FA9         STA TEMPP
10240 B14E 4CB1AB         JMP REPORT
10250 B151 68      XQTRTI PLA
10260 B152 8D1FA9         STA TEMPP
10270 B155 68             PLA
10280 B156 8D24A9         STA TEMPPC
10290 B159 68             PLA
10300 B15A 8D25A9         STA TEMPPC+1
10310 B15D 4CB1AB         JMP REPORT
10320      TABLE1
10330
10340  This is a data file containing the complete set of
10350  6502 Op Codes.
10360
10370 ================================================
10380
10390      Fields:
10400         (1) OP CODE VALUE,   1 BYTE
10410         (2) MNEMONIC,        3 BYTES
10420         (3) LEN OF INSTR,    1 BYTE
10430         (4) ADR MODE #,      1 BYTE
10440                   (01) = IMMEDIATE
10450                   (02) = ABSOLUTE
10460                   (03) = ZERO PAGE
10470                   (04) = ACCUMULATOR
10480                   (05) = IMPLIED
10490                   (06) = (INDIRECT,X)
10500                   (07) = (INDIRECT),Y
10510                   (08) = ZERO PAGE,X
10520                   (09) = ABSOLUTE,X
10530                   (10) = ABSOLUTE,Y
10540                   (11) = RELATIVE
10550                   (12) = INDIRECT
10560                   (13) = ZERO PAGE,Y
10570
10590 ========================================
10590TABLE1   .BYTE $00,'BRK',1,05
10600         .BYTE $01,'ORA',2,06
10610         .BYTE $05,'ORA',2,03
10620         .BYTE $06,'ASL',2,03
10630         .BYTE $08,'PHP',1,05
10640         .BYTE $09,'ORA',2,01
10650         .BYTE $0A,'ASL',1,04
10660         .BYTE $0D,'ORA',3,02
10670         .BYTE $0E,'ASL',3,02
10680         .BYTE $10,'BPL',2,11
10690         .BYTE $11,'ORA',2,07
10700         .BYTE $15,'ORA',2,08
10710         .BYTE $16,'ASL',2,08
10720         .BYTE $18,'CLC',1,05
10730         .BYTE $19,'ORA',3,10
10740         .BYTE $1D,'ORA',3,09
10750         .BYTE $1E,'ASL',3,09
10760         .BYTE $20,'JSR',3,02
10770         .BYTE $21,'AND',2,06
10780         .BYTE $24,'BIT',2,03
10790         .BYTE $25,'AND',2,03
10800         .BYTE $26,'ROL',2,03
10810         .BYTE $28,'PLP',1,05
10820         .BYTE $29,'AND',2,01
10830         .BYTE $2A,'ROL',1,04
10840         .BYTE $2C,'BIT',3,02
10850         .BYTE $2D,'AND',3,02
10860         .BYTE $2E,'ROL',3,02
10870         .BYTE $30,'BMI',2,11
10880         .BYTE $31,'AND',2,07
10890         .BYTE $35,'AND',2,08
10900         .BYTE $36,'ROL',2,08
10910         .BYTE $38,'SEC',1,05
10920         .BYTE $39,'AND',3,10
10930         .BYTE $3D,'AND',3,09
10940         .BYTE $3E,'ROL',3,09
10950         .BYTE $40,'RTI',1,05
10960         .BYTE $41,'EOR',2,06
10970         .BYTE $45,'EOR',2,03
10980         .BYTE $46,'LSR',2,03
10990         .BYTE $48,'PHA',1,05
11000         .BYTE $49,'EOR',2,01
11010         .BYTE $4A,'LSR',1,04
11020         .BYTE $4C,'JMP',3,02
11030         .BYTE $4D,'EOR',3,02
11040         .BYTE $4E,'LSR',3,02
11050         .BYTE $50,'BVC',2,11
11060         .BYTE $51,'EOR',2,07
11070         .BYTE $55,'EOR',2,08
11080         .BYTE $56,'LSR',2,08
11090         .BYTE $59,'CLI',1,05
11100         .BYTE $59,'EOR',3,10
11110         .BYTE $5D,'EOR',3,09
11120         .BYTE $5E,'LSR',3,09
11130         .BYTE $60,'RTS',1,05
11140         .BYTE $61,'ADC',2,06
11150         .BYTE $65,'ADC',2,03
11160         .BYTE $66,'ROR',2,03
11170         .BYTE $68,'PLA',1,05
11180         .BYTE $69,'ADC',2,01
11190         .BYTE $6A,'ROR',1,04
11200         .BYTE $6C,'JMP',3,12
11210         .BYTE $6D,'ADC',3,02
11220         .BYTE $6E,'ROR',3,02
11230         .BYTE $70,'BVS',2,11
11240         .BYTE $71,'ADC',2,07
11250         .BYTE $75,'ADC',2,08
11260         .BYTE $76,'ROR',2,08
11270         .BYTE $78,'SEI',1,05
11280         .BYTE $79,'ADC',3,10
11290         .BYTE $7D,'ADC',3,09
11300         .BYTE $7E,'ROR',3,09
11310         .BYTE $81,'STA',2,06
11320         .BYTE $84,'STY',2,03
11330         .BYTE $85,'STA',2,03
11340         .BYTE $86,'STX',2,03
11350         .BYTE $88,'DEY',1,05
11360         .BYTE $8A,'TXA',1,05
11370         .BYTE $8C,'STY',3,02
11380         .BYTE $8D,'STA',3,02
11390         .BYTE $8E,'STX',3,02
11400         .BYTE $90,'BCC',2,11
11410         .BYTE $91,'STA',2,07
11420         .BYTE $94,'STY',2,08
11430         .BYTE $95,'STA',2,08
11440         .BYTE $96,'STX',2,13
11450         .BYTE $98,'TYA',1,05
11460         .BYTE $99,'STA',3,10
11470         .BYTE $9A,'TXS',1,05
11480         .BYTE $9D,'STA',3,09
11490         .BYTE $A0,'LDY',2,01
11500         .BYTE $A1,'LDA',2,06
11510         .BYTE $A2,'LDX',2,01
11520         .BYTE $A4,'LDY',2,03
11530         .BYTE $A5,'LDA',2,03
11540         .BYTE $A6,'LDX',2,03
11550         .BYTE $A8,'TAY',1,05
11560         .BYTE $A9,'LDA',2,01
11570         .BYTE $AA,'TAX',1,05
11580         .BYTE $AC,'LDY',3,02
11590         .BYTE $AD,'LDA',3,02
11600         .BYTE $AE,'LDX',3,02
11610         .BYTE $B0,'BCS',2,11
11620         .BYTE $B1,'LDA',2,07
11630         .BYTE $B4,'LDY',2,08
11640         .BYTE $B5,'LDA',2,09
```

```
11650          .BYTE $B5,'LDX',2,13
11660          .BYTE $B8,'CLV',1,05
11670          .BYTE $B9,'LDA',3,10
11680          .BYTE $BA,'TSX',1,05
11690          .BYTE $BC,'LDY',3,09
11700          .BYTE $BD,'LDA',3,09
11710          .BYTE $BE,'LDX',3,10
11720          .BYTE $C0,'CPY',2,01
11730          .BYTE $C1,'CMP',2,06
11740          .BYTE $C4,'CPY',2,03
11750          .BYTE $C5,'CMP',2,03
11760          .BYTE $C6,'DEC',2,03
11770          .BYTE $C8,'INY',1,05
11780          .BYTE $C9,'CMP',2,01
11790          .BYTE $CA,'DEX',1,05
11800          .BYTE $CC,'CPY',3,02
11810          .BYTE $CD,'CMP',3,02
11820          .BYTE $CE,'DEC',3,02
11830          .BYTE $D0,'BNE',2,11
11840          .BYTE $D1,'CMP',2,07
11850          .BYTE $D5,'CMP',2,08
11860          .BYTE $D6,'DEC',2,08
11870          .BYTE $D8,'CLD',1,05
11880          .BYTE $D9,'CMP',3,10
11890          .BYTE $DD,'CMP',3,09
11900          .BYTE $DE,'DEC',3,09
11910          .BYTE $E0,'CPX',2,01
11920          .BYTE $E1,'SBC',2,06
11930          .BYTE $E4,'CPX',2,03
11940          .BYTE $E5,'SBC',2,03
11950          .BYTE $E6,'INC',2,03
11960          .BYTE $E8,'INX',1,05
11970          .BYTE $E9,'SBC',2,01
11980          .BYTE $EA,'NOP',1,05
11990          .BYTE $EC,'CPX',3,02
12000          .BYTE $ED,'SBC',3,02
12010          .BYTE $EE,'INC',3,02
12020          .BYTE $F0,'BEQ',2,11
12030          .BYTE $F1,'SBC',2,07
12040          .BYTE $F5,'SBC',2,08
12050          .BYTE $F6,'INC',2,08
12060          .BYTE $F8,'SED',1,05
12070          .BYTE $F9,'SBC',3,10
12080          .BYTE $FD,'SBC',3,09
12090          .BYTE $FE,'INC',3,09
12100;
12110;   End of TABLE1
12120MSG1      .BYTE 13,10,'INPUT HEX STARTING ADDRESS: $'
12130          .BYTE $FF
12140MSG2      .BYTE 13,10,'OUTPUT TO:'
12150          .BYTE 13,10,'          (1) CONSOLE'
12160          .BYTE 13,10,'          (2) PRINTER'
12170          .BYTE 13,10,'          (3) BOTH'
12180          .BYTE 13,10,'         ?',$FF
12190MSG3      .BYTE 13,10,'Initial A Register Contents: $',$FF
12200          .BYTE 13,10,'Initial X Register Contents: $',$FF
12210          .BYTE 13,10,'Initial Y Register Contents: $',$FF
12220          .BYTE 13,10,'Initial Condition Code Contents : $',$FF
12230MSG4      .BYTE 'ADDR   CODE MNEMONIC OPERAND  EFF ADDR'
12240          .BYTE '   A    X    Y   N-V-x-B-D-I-Z-C   SP',$FF
12250LINE      .BYTE 00
12260*         = LINE+$50
12270TABLE4    .END
```

```
  10                ;
  20                ;                       TBASIC machine code
  30                ;
  40                ;                     12/04/82    Carl Eidbo
  50                ;                     05/23/83  Modified Output Sub
  60                ;
  70                ;=================================================
  80 0010=          INDEX1  = $0010
  90 0012=          INDEX2  = INDEX1+02
 100 0014=          CURRC1  = INDEX2+02
 110 0014=          NUMDEL  = CURRC1    ; CURRC1 not used in DELETE.
 120 0015=          CURRC2  = CURRC1+01
 130 0016=          DECNUM  = CURRC2+01 ; Temp for decimal number.
 140 6200=          TABLE4  = $6200
 150 2501=          BASOD   = $2501
 160 2D33=          OUTD:   = $2D33
 170                ;
 180 6000           *       = $6000
 190                ;
 200 6000 4C1A60            JMP DELETE
 210 6003 4C5B60            JMP SORT
 220 6006 4CAB60            JMP DSPLAY
 230                ;
 240 6009 0062      BOD4    .WORD TABLE4
 250 600B 0A        RL4     .BYTE $0A
 260 600C 00        RN4     .BYTE $00 ; To be filled in by BASIC
 270 600D 00        STREC   .BYTE $00 ; First record to be displayed.
 280 600E 00        ENREC   .BYTE $00 ; Last  record to be displayed.
 290 600F 2E        TABCHR  .BYTE '.'
 300 6010 FF        DELCHR  .BYTE $FF
 310 6011 0000      TEMPPZ  .DBYTE $00,$00,$00,$00 ; Temp. to save PZ contents.
 310 6013 0000
 310 6015 0000
 310 6017 0000
 320 6019 00        TEMPY   .BYTE $00 ; Temp Y reg.
 330                ;
 340                ;=================================================
 350                ;
```

```
 360 601A 203861 DELETE  JSR INIZ
 370 601D A004   CHKDEL  LDY #$04
 380 601F B112           LDA (INDEX2),Y
 390 6021 CD1060         CMP DELCHR
 400 6024 D004           BNE DELNO
 410 6026 E614           INC NUMDEL
 420 6028 D015           BNE DELYES ; Branch Always
 430 602A A414   DELNO   LDY NUMDEL
 440 602C F00C           BEQ MOVDON
 450 602E A000   DLMOVE  LDY #$00
 460 6030 B112           LDA (INDEX2),Y
 470 6032 9110           STA (INDEX1),Y
 480 6034 C8             INY
 490 6035 CC0B60         CPY RL4
 500 6038 D0F6           BNE DLMOVE+02
 510 603A A200   MOVDON  LDX #$00
 520 603C 20C861         JSR BUMP
 530 603F A202   DELYES  LDX #$02
 540 6041 20C861         JSR BUMP
 550 6044 A415           LDY CURRC2
 560 6046 CC0C60         CPY RN4
 570 6049 F004           BEQ DELDON
 580 604B E615           INC CURRC2
 590 604D D0CE           BNE CHKDEL
 600 604F 38     DELDON  SEC
 610 6050 A514           LDA NUMDEL
 620 6052 ED0C60         SBC RN4
 630 6055 8D0C60         STA RN4
 640 6058 4C2C61         JMP ALLDON
 650             ;
 660             ;================================================
 670             ;
 680 605B 203861 SORT    JSR INIZ
 690 605E A514   SRSTRT  LDA CURRC1
 700 6060 8515           STA CURRC2
 710 6062 A510           LDA INDEX1
 720 6064 8512           STA INDEX2
 730 6066 A511           LDA INDEX1+01
 740 6068 8513           STA INDEX2+01
 750 606A A202   BUMP:2  LDX #$02
 760 606C 20C861         JSR BUMP
 770 606F E615           INC CURRC2
 780 6071 A002           LDY #$02
 790 6073 88     NXBYTE  DEY
 800 6074 3019           BMI CHNXRC
 810 6076 B112           LDA (INDEX2),Y ; Get (b)
 820 6078 300C           BMI GT127      ; (b) < 0, [ >127]
 830 607A B110           LDA (INDEX1),Y ; Get (a) , (b) > 0 , [ <128]
 840 607C 300E           BMI SWAP:1     ; (a) < 0 , [ >127 ] ,, (a) > (b)
 850 607E D112   CMPARE  CMP (INDEX2),Y ; (a),(b) > 0 , [ <128]
 860 6080 F0F1           BEQ NXBYTE     ; (a) = (b)
 870 6082 300B           BMI CHNXRC     ; (a) < (b)
 880 6084 1006           BPL SWAP:1     ; (a) > (b)
 890 6086 B110   GT127   LDA (INDEX1),Y
 900 6088 1005           BPL CHNXRC     ; (a) < (b)
 910 608A 30F2           BMI CMPARE
 920 608C 20D761 SWAP:1  JSR SWAP
 930 608F A415   CHNXRC  LDY CURRC2 ; CHeck NeXt ReCord.
 940 6091 CC0C60         CPY RN4
 950 6094 D0D4           BNE BUMP:2
 960 6096 A414   BUMP:1  LDY CURRC1
 970 6098 C8             INY
 980 6099 CC0C60         CPY RN4
 990 609C F00A           BEQ SRDONE
1000 609E A200           LDX #$00
1010 60A0 20C861         JSR BUMP
1020 60A3 E614           INC CURRC1
1030 60A5 4C5E60         JMP SRSTRT
1040 60A8 4C2C61 SRDONE  JMP ALLDON
1050             ;
1060             ;================================================
1070             ;
1080 60AB 203861 DSPLAY  JSR INIZ
1090 60AE AC0D60         LDY STREC
1100 60B1 F00A           BEQ DSSTRT
1110 60B3 A200           LDX #$00
1120 60B5 20C861 BUMP:   JSR BUMP
1130 60B8 E614           INC CURRC1
1140 60BA 88             DEY
1150 60BB D0F8           BNE BUMP:
1160             ;
1170 60BD AE0E60 DSSTRT  LDX ENREC
1180 60C0 E414           CPX CURRC1
1190 60C2 3068           BMI ALLDON
1200 60C4 A203   CNTNU1  LDX #$03
1210 60C6 205961         JSR SPACE
1220 60C9 A928           LDA #'(
1230 60CB 207B61         JSR OUTCHR
1240 60CE A614           LDX CURRC1
1250 60D0 E8             INX
1260 60D1 8A             TXA
1270 60D2 208C61         JSR HEXDEC
1280 60D5 A929           LDA #')
1290 60D7 207B61         JSR OUTCHR
1300
1310 60DA A000           LDY #$00
1320 60DC A205   ADDR01  LDX #$05
1330 60DE 205961         JSR SPACE
1340 60E1 C004           CPY #$04
1350 60E3 F029           BEQ LBLOUT
1360 60E5 A924           LDA #'$  ; Output Start & End Addresses.
1370 60E7 207B61         JSR OUTCHR
```

```
1380 60EA B110           LDA (INDEX1),Y
1390 60EC 48             PHA
1400 60ED C8             INY
1410 60EE B110           LDA (INDEX1),Y
1420 60F0 D012           BNE ADDR04 ; Check for Address = $0000.
1430 60F2 68             PLA
1440 60F3 D00C           BNE ADDR03
1450 60F5 A204           LDX #$04   ; Yes, Address = $0000,
1460 60F7 A92D           LDA #'-    ;       so print $----.
1470 60F9 207B61 ADDR02  JSR OUTCHR
1480 60FC CA             DEX
1490 60FD D0FA           BNE ADDR02
1500 60FF F00A           BEQ ADDR05
1510 6101 48     ADDR03  PHA
1520 6102 A900           LDA #$00
1530 6104 206461 ADDR04  JSR HEXOUT
1540 6107 68             PLA
1550 6108 206461         JSR HEXOUT
1560 610B C8     ADDR05  INY
1570 610C D0CE           BNE ADDR01
1580
1590 610E B110   LBLOUT   LDA (INDEX1),Y
1600 6110 207B61         JSR OUTCHR
1610 6113 C8             INY
1620 6114 CC0B60         CPY RL4
1630 6117 D0F5           BNE LBLOUT
1640
1650 6119 A90D   NXTREC  LDA #$0D ; Print CR/LF at end of line.
1660 611B 207B61         JSR OUTCHR
1670 611E A90A           LDA #$0A
1680 6120 207B61         JSR OUTCHR
1690 6123 A200           LDX #$00
1700 6125 20C861         JSR BUMP
1710 6128 E614           INC CURRC1
1720 612A D091           BNE DSSTRT
1730
1740 612C A007   ALLDON  LDY #$07
1750 612E B91160         LDA TEMPPZ,Y
1760 6131 991000         STA INDEX1,Y
1770 6134 88             DEY
1780 6135 10F7           BPL ALLDON+02
1790 6137 60             RTS ; Return to BASIC.
1800
1810             ;================================================
1820             ;
1830             ;                 Subroutine
1840             ;
1850 6138 D8     INIZ    CLD       ; Common initialization sub.
1860 6139 A007           LDY #$07
1870 613B B91000         LDA INDEX1,Y ; Preserve Page Zero Contents.
1880 613E 991160         STA TEMPPZ,Y
1890 6141 88             DEY
1900 6142 10F7           BPL INIZ+03
1910 6144 AC0960         LDY BOD4
1920 6147 8410           STY INDEX1
1930 6149 8412           STY INDEX2
1940 614B AC0A60         LDY BOD4+01
1950 614E 8411           STY INDEX1+01
1960 6150 8413           STY INDEX2+01
1970 6152 A200           LDX #$00
1980 6154 8614           STX CURRC1
1990 6156 8615           STX CURRC2
2000 6158 60             RTS
2010
2020 6159 A920   SPACE   LDA #$20 ; Space.
2030 615B CA             DEX       ; Pass # of spaces in X.
2040 615C 3005           BMI SPDONE
2050 615E 207B61         JSR OUTCHR
2060 6161 D0F8           BNE SPACE+02
2070 6163 60     SPDONE  RTS
2080
2090 6164 48     HEXOUT  PHA
2100 6165 29F0           AND #$F0 ; Mask out lower 4 bits.
2110 6167 4A             LSR A
2120 6168 4A             LSR A
2130 6169 4A             LSR A
2140 616A 4A             LSR A
2150 616B 207161         JSR HXOUT
2160 616E 68             PLA
2170 616F 290F           AND #$0F ; Mask out upper 4 bits.
2180 6171 18     HXOUT   CLC
2190 6172 6930           ADC #'0
2200 6174 C93A           CMP #':
2210 6176 3003           BMI OUTCHR
2220 6178 18             CLC
2230 6179 6907           ADC #$07 ; Difference between 'A' and '9'.
2240 617B 08     OUTCHR  PHP
2250 617C 8C1960         STY TEMPY
2260 617F A000           LDY #$00
2270 6181 8C0125         STY BASOD
2280 6184 20332D         JSR OUTD:
2290 6187 AC1960         LDY TEMPY
2300 618A 28             PLP
2310 618B 60             RTS
2320
2330 618C 48     HEXDEC  PHA ; Hex to Dec Converter.
2340 618D F8             SED
2350 618E A200           LDX #$00
2360 6190 8616           STX DECNUM
2370 6192 8617           STX DECNUM+01
2380 6194 4A             LSR A
2390 6195 4A             LSR A
```

```
2400 6196 4A             LSR A
2410 6197 4A             LSR A
2420 6198 F006           BEQ ONES
2430 619A AA             TAX : Add up 16's.
2440 619B A916   ADD16S  LDA #$16
2450 619D 20B761         JSR ADDER
2460 61A0 68     ONES    PLA
2470 61A1 290F           AND #$0F
2480 61A3 F006           BEQ DECOUT
2490 61A5 AA             TAX
2500 61A6 A901   ADD01S  LDA #$01
2510 61A8 20B761         JSR ADDER
2520 61AB D8     DECOUT  CLD
2530 61AC A517           LDA DECNUM+01
2540 61AE 206F61         JSR HXOUT-02
2550 61B1 A516           LDA DECNUM
2560 61B3 206461         JSR HEXOUT
2570 61B6 60             RTS
2580                     ;
2590 61B7 48     ADDER   PHA
2600 61B8 18             CLC
2610 61B9 68             PLA
2620 61BA 48             PHA
2630 61BB 6516           ADC DECNUM
2640 61BD 8516           STA DECNUM
2650 61BF 9002           BCC NXTADD
2660 61C1 E617           INC DECNUM+01
2670 61C3 CA     NXTADD  DEX
2680 61C4 D0F2           BNE ADDER+01
2690 61C6 68             PLA
2700 61C7 60             RTS
2710                     ;
2720 61C8 48     BUMP    PHA : Save A
2730 61C9 AD0B60         LDA RL4
2740 61CC 18     BUMP1   CLC
2750 61CD 7510           ADC INDEX1,X
2760 61CF 9510           STA INDEX1,X
2770 61D1 9002           BCC NXT1
2780 61D3 F611           INC INDEX1+01,X
2790 61D5 68     NXT1    PLA
2800 61D6 60             RTS
2810                     ;
2820 61D7 A000   SWAP    LDY #$00
2830 61D9 B110           LDA (INDEX1),Y
2840 61DB 48             PHA
2850 61DC B112           LDA (INDEX2),Y
2860 61DE 9110           STA (INDEX1),Y
2870 61E0 68             PLA
2880 61E1 9112           STA (INDEX2),Y
2890 61E3 C8             INY
2900 61E4 CC0B60         CPY RL4
2910 61E7 D0F0           BNE SWAP+02
2920 61E9 60             RTS
2930
```

OS-65U Grade Recording System

by John Hepner
9500 Huffman Road
Farmersville, OH 45325

[Editor's Note: John sent this software on disk without an accompanying article. I am using the text of his letter to help document things, but don't let that dissuade you from trying it. The software is well-documented internally, and is fairly straightforward, so it should be easy for you to modify to suit your needs. One thing to watch out for is that John wrote this for his serial-based OSI. Video system owners should be aware that the programs sometimes alter the console device number with POKEs while asking for passwords and at other prompts. In addition, the program assumes a particular paralell printer installed as device #4, and occasionally sends special formatting codes. I would recommend carefully altering all "PRINT#4" statements by defining a printer device number in the variable "PD%" to reflect your own setup, and use "PRINT#PD%" instead. You'll also want to REM-out all of these special printer commands until you're comfortable with the software.]

This is the most recent version of my grade recording program. The programs have grown over the years as I found need of new abilities, or just liked the way someone else's program did something.

In its present form, no distinction is made among different types of grades. Tests and homework are all treated equally. It makes for a simple file structure.

START calls all of the other generally used programs. INITL sets up the student file structure. STUDNT allows adding students to the file. CHNGE allows entering of grades. EDIT lets the file be modified in any desired way. REPORT, PROGRS, and FINAL are various types of reports needed for various school purposes. HELP gives hints as to when and how to use the various functions.

Passwords are "PASS" for the report writers, and "QWERTY" for the ones that can alter the data file contents. Note that "QWERTY" never appears on the screen. [Editor's Note: this is one of the places where the console device number gets altered in all of the programs. If you see "QWERTY", you'll need to pay attention to that line of the program.] Possibly someone else has used this method to supress printing, but I'll claim it as my only true original part of the system.

In actual use, I keep the data on a disk in the "B" drive (and in my briefcase), and programs on a disk in the "A" drive (near the computer). This method of data storage isn't especially efficient in terms of disk space, but it is FAST! It runs about 5 times faster than a similar MECC (Minnesota Educational Computer Consortium) program on an Apple IIe (partly because the program doesn't stop to sort the file on every use).

I'm currently working on a version that allows optimally sorted lists by using record pointers (not key files) to indicate the next & previous records. When new records are added, the pointers will be updated then, and the list stays "sorted" if that is needed.

It also would be nice to add comments to the grades, and there has been a request for "typing" or weighting of the grades (by "TEST", "QUIZ", "HOMEWORK", etc.) so some types of grades can receive more credit. If your readers are interested, I'll send the newer versions along periodically when they work well. [Editor's Note: If you send a request to John for an updated copy of this software, be courteous enough to send a blank disk, a fresh mailer, and enough cash or stamps to cover return postage]

Write for PEEK[65]!

Share the experience!

```
10 REM -START
20 REM -School Records
30 :
40 TRAP 20000
50 :
100 POKE133,122:CLEAR
110 POKE2888,0:POKE8722,0:POKE2073,96
140 :
150 :
160 PRINT CHR$(12):REM CLS/HOME
170 IF PEEK(8993)=1 THEN PRINT CHR$(126);CHR$(28):REM Hazeltine CLR
180 FOR N=0 TO 5: NEXT :REM time for the terminal to clear
200 :
2000 REM --------------------------
2020 PRINT:PRINT:PRINT"  GRADES MANAGEMENT program
2030 PRINT"     1>     Initialize Data files
2035 PRINT"     2>     Enter student names / add new student
2040 PRINT:REM"     3>     Delete student name (not implimented yet)
2045 PRINT"     4>     Enter grades / scores
2047 PRINT
2050 PRINT"     5>     Print class reports in column form.
2055 REM:PRINT"     6>     Calculate final class grades, averages, ";
2057 REM :          PRINT"distributions"
2060 PRINT"     7>     Print Report Cards
2063 PRINT
2065 PRINT"     8>     Edit any record
2070 PRINT:REM"     9>     List directory of Class files
2075 PRINT"     0>     RETURN TO MAIN MENU
2080 PRINT:PRINT"     HELP!
2085 :
2090 PRINT:PRINT:PRINT
2100 INPUT"Enter the number of your choice-";I$
2110 :
2120 :
2130 IF I$="1" THEN RUN"INITL
2135 IF I$="2" THEN RUN"STUDNT
2140 REM: IF I$="3" THEN RUN"DELETE
2145 IF I$="4" THEN RUN"CHNGE
2150 IF I$="5" THEN RUN"REPORT
2155 IF I$="6" THEN RUN"FINAL
2160 IF I$="7" THEN RUN"PROGRS
2165 IF I$="8" THEN RUN"EDIT
2170 REM: IF I$="9" THEN GOSUB 1000
2175 IF I$="0" THEN RUN"BEXEC*
2180 IF I$="EXIT" THEN GOTO 60000
2190 IF LEFT$(I$,1)="H" THEN RUN"HELP
3000 GOTO 10
19999 :
20000 REM -Error handler
20010 PRINT"ERROR *6 -Is proper disk in place ";:INPUT PW$
20020 RUN
20030 :
50000 REM - Unlock System
59000 POKE741,76:POKE750,78:POKE2073,173:POKE2893,55:POKE2894,8
59010 POKE2888,27:X=PEEK(8960):POKE133,X
59020 POKE 2888,27: POKE 8722,27: POKE 2073,173
59040 RETURN
59050 :
60000 GOSUB59000
60010 GOSUB50000:CLEAR
```

```
10 REM -INITL
20 PRINT CHR$(12): REM CLR/HOME for ADDS
25 :
30 PRINT"  Initialize data files
40 PRINTCHR$(8);" WARNING!!! -PASSWORD REQUIRED
50 PRINT:PRINT"     THIS PROGRAM CAN DELETE ALL DATA FILES!!!
52 PRINT:PRINT" Warning- Back up ALL data files before doing this!!!"
55 :
60 PRINT:PRINT" PASSWORD-?";:POKE 8994,0 :REM -Supress output
65 INPUT PW$: X=PEEK(8993): POKE 8994,X :PRINT :REM -Enable output
70 IF PW$<>"QWERTY" THEN 9999
80 :
100 PRINT"  This program writes blanks throughout the data fields.
110 PRINT"It can destroy all old records.  Do you wish to do this?";
120 INPUT PW$
130 IF LEFT$(PW$,1) <> "Y" THEN 9999
140 :
150 PRINT"Do you wish to retain student names and numbers";
160 INPUT PW$ :F1 =0 :IF LEFT$(PW$,1)="Y" THEN F1 =1
170 :
180 INPUT"Which file do you intend to work with";PW$
190 INPUT"How long is the file (in Tracks)";FL: FL=FL*23-1
195 :
200 REM -Data fields & variables:
205 REM      N$    student name            20 characters
210 REM      NU$   student number           6 digits
220 REM      G(C) student grades            3 digits
225 REM        C      which grade           0 to 30 grades
230 REM
240 REM      PW$     General input variable. Not saved.
245 REM      F1      Flag for Name. Save NAME field if =1.
247 REM      FL      File Length as read from File Record 0
248 REM                12 * Number of tracks
250 REM      I       General counter for loops
300 :
310 NU$="00    "
320 DIM G(48)
330 :
400 REM -Access file
405 :
410 DISK OPEN,6,PW$
415 DISK GET,0
420 IF F1 =0 THEN PRINT#6,PW$ :PRINT#6,"000"
421 IF F1 =1 THEN INPUT #6, N$, NU$:DISK GET,0 :PRINT#6,N$ :PRINT#6,NU$
422 FOR C=0 TO 30
424 :   PRINT#6,"-"
426 NEXT C
428 PRINT#6,"000"
429 :
430 FOR I= 1 TO FL
435 :   DISK GET, I
440 :   N$="----------"
450 :   IF F1 =1 THEN INPUT#6,N$, NU$
455 :
460 :   PRINT#6,N$ :PRINT#6,NU$
470 :   FOR C=0 TO 30
475 :     PRINT#6,"-"
480 :   NEXT C
482 :   PRINT#6,"000"
485 :   PRINT" ";
490 :
500 NEXT I
```

```
505 PRINT
510 :
520 DISK CLOSE,6
530 :
9000 REM -Exit routine
9010 PRINT"File initialization complete"
9020 PRINT
9030 FOR N=0TO500:NEXT
9999 RUN"START


10 REM -STUDNT
15 REM --23 Nov 85
20 PRINT CHR$(12): REM CLR/HOME for ADDS
25 :
30 PRINT"  Enter names into data files
40 PRINTCHR$(8);" WARNING!!! -PASSWORD REQUIRED
50 PRINT:PRINT"     THIS PROGRAM CAN DELETE ALL NAMES!!!
55 :
60 PRINT" PASSWORD-!";:POKE 8994,0: INPUT PW$
65 X=PEEK(8993): POKE 8994,X: REM Restore console output
70 IF PW$<>"QWERTY" THEN 9999
75 :
80 TRAP 9999
100 PRINT"  This program writes student names into class files.
110 PRINT"It can destroy all old records.  Do you wish to do this?";
120 INPUT PW$
130 IF LEFT$(PW$,1) <> "Y" THEN 9999
140 :
145 INPUT" Do you wish to start a new file or add to an old one";PW$
150 F=1 :IF LEFT$(PW$,1)="N" THEN F=0
155 :
160 IF F=0 THEN INPUT"Do you wish to copy names from an old file";PW$
165 F1 =0: IF LEFT$(PW$,1)="Y" THEN F1=1
170 IF F1 THEN GOSUB 20000: REM Go get names from Old file.
175 :
180 PRINT"Which file do you intend to work with";
190 INPUT PW$
195 :
199 REM ----------------------
200 REM -Data fields & variables:
205 REM      N$   student name            20 characters
210 REM      NU$  student number           6 digits
220 REM      G(C) student grades           3 digits
225 REM        C      which grade          0 to 30 grades
230 REM
240 REM      PW$     General input variable. Not saved.
245 REM      I       General counter for loops
250 REM      S       Start of Loop for records: 0 or 1
255 :
260 REM      F       Flag =1 means UPDATE old file.
265 REM      F1      Flag =1 means copy from Old file
270 :
299 REM ------------------------
300 REM -Update file records
310 DIM G$(33)
315 S=0: IF F1=0 THEN NU=96: S=1
320 :
330 PRINT:PRINT"Enter '///' to stop entering names."
340 :
```

```
400 REM -Access file
405 :
410 DISK OPEN,6,PW$
415 DISK GET,0
420 INPUT#6,N$,NU$
425 PRINT"File is "N$" with "NU$" records.": PW$=N$
426 :
430 FOR I= S TO NU
433 PRINT I;
435 :    DISK GET, I
440 :     INPUT#6,N$,NU$
442 :      FOR C=0 TO 30
444 :        INPUT#6,G$(C)
446 :      NEXT C
447 :
448 :      IF (F1 AND (NOT F)) THEN N$=N$(I): NU$=NU$(I): GOTO 560
449 :
450 :     PRINT#1, N$;SPC(20-LEN(N$));"  ";NU$   :REM Display data record
460 :        FOR C=0TO30 :PRINT G$(C); :NEXT C :PRINT
465 :     IF F=1 AND LEFT$(N$,1)<>"-" THEN 600
470 :     INPUT N$,NU$ :REM Enter new data
475 :     IF LEN(N$)>20 OR LEN(NU$)>6 THEN PRINT"Too long." :GOTO 470
480 :     IF LEFT$(N$,3)="///" THEN 620
490 :
510 :
560 :      PRINT#6,N$ :PRINT#6,NU$
570 :      FOR C=0 TO 30
575 :        PRINT#6,G$(C)
580 :      NEXT C
585 :   PRINT" ";
590 :
600 NEXT I
605 PRINT
610 :
620 DISK GET,0
630 PRINT#6,PW$: PRINT#6,STR$(I-1)
650 DISK CLOSE,6
660 :
8999 REM ----------------------
9000 REM -Exit routine
9010 PRINT"File update complete"
9020 PRINT
9030 FOR N=0TO500:NEXT
9999 RUN"START
19999 REM ----------------------
20000 REM -Read OLD file, store in array of N$(I),NU$(I)
20010 INPUT"Name of the Old file";OF$
20030 INPUT"Which disk drive ";DR$: DISK!"SEL "+DR$
20040 :
20050 REM -Read record 0
20055 :
20060 DISK OPEN,6,OF$
20070 DISK GET, 0
20080 INPUT#6, N$,NU$
20090 PRINT"File is "N$" with "NU$" records.
20100 NU=VAL(NU$)
20105 :
20110 REM -Create an array to store Names & Numbers
20120 DIM N$(NU), NU$(NU)
20125 :
```

```
20130 FOR I=0 TO NU
20140 :   PRINT I;" ";
20150 :   DISK GET,I
20160 :   INPUT#6,N$(I),NU$(I)
20170 :   PRINT NU$(I);" ";N$(I)
20180 NEXT I
20190 :
20200 DISK CLOSE,6
20210 DISK!"SEL A"
20220 :
29010 PRINT"Now for the New file: ";
29998 RETURN



10 REM - CHNGE
15 REM  --Last version:  13 Dec 85
20 PRINT CHR$(12): REM CLR/HOME for ADDS
30 PRINT"  Enter grades to student files.
35 :
50 :
55 :
60 PRINT:PRINT" PASSWORD-?";:POKE 8994,0 :REM -Supress output
65 INPUT PW$: X=PEEK(8993): POKE 8994,X :PRINT :REM -Enable output
70 IF PW$<>"QWERTY" THEN 9999
75 :
80 :
85 :
100 :
110 :
120 PRINT:INPUT" Do you wish to print the grade distribution";F1$
130 F1=1: IF LEFT$(F1$,1)<>"Y" THEN F1=0
140 IF F1 THEN INPUT"Description of Activity";MSG$
150 IF F1 THEN PRINT#4,CHR$(64)CHR$(27)"B"CHR$(3)
170 :
180 PRINT"Which file do you intend to work with";
190 INPUT PW$
195 :
200 REM -Data fields & variables:
205 REM      N$       student name            20 characters
210 REM      NU$      student number           6 digits
220 REM      G$(C)    student grades           3 digits
225 REM        C      which grade              0 to 30 grades
230 REM
240 REM      PW$     General input variable. Not saved.
242 REM      TTL     Student's ToTaL score
244 REM      NU      Number of Records in File as read from Record 0
246 REM      DS(I)   Array of student grades for Mean/SD/Sort/Tally
250 REM      I,J     General counter for loops
252 REM      SC      Student Score, used in tally routine
255 REM      DV      Device number for output
260 REM      F1      Flag, 1=Print Grade distribution
265 REM      MSG$    Message to use above distribution
270 REM      MEAN    Mean value of grades
275 REM      SMALL   Smallest value (Sort)
280 REM      TV      Total value of deviations (SD)
285 REM      SD      Standard Deviation
290 :
295 :
300 REM -
310 DIM G$(33)
315 DV=1 :REM Output device
```

```
320 DIM DS(100)
330 :
340 :
400 REM -Access file
402 DISK OPEN,6,PW$
404 DISK GET, 0
406 :
408 INPUT*6,N$,NU$  :REM  Read whole record 0
410 NU =VAL(NU$): PRINT N$,NU: IF F1 THEN DIM DS(NU): REM Save Grades
412 FOR C= 0 TO 30
414 :   INPUT*6,G$(C): PRINT G$(C);" ";
416 NEXT C
418 INPUT*6,TTL
419 :
420 PRINT*DU, N$   :REM  Display old record 0
421 FOR C=0 TO 30
422 :   IF G$(C) <> "-" THEN GOTO 424
423 :   IF G$(C) = "-"  THEN INPUT"Possible score";G$(C):GOTO 426
424 NEXT C
425 :
426 PRINT*6,N$: PRINT*6,NU$ :DS(0)=VAL(G$(C)) :REM Update Record 0
427 TTL=0
428 FOR C=0 TO 30
430 :   PRINT*6,G$(C) :TTL =TTL + VAL(G$(C))
431 NEXT C
432 PRINT*6,TTL
433 :
434 FOR I= 1 TO NU        :REM Update student's records
436 PRINT I;
437 TTL =0  :REM Set Student's ToTaL score to 0
438 :   DISK GET, I  : REM Read disk record I
440 :    INPUT*6,N$,NU$
442 :     FOR C=0 TO 30
444 :       INPUT*6,G$(C)
446 :     NEXT C
448 :
450 :    PRINT*1, N$,NU$ :REM Display data record
452 :       FOR C=0TO30        : REM Print grades to end of list
454 :           IF G$(C)<>"-"THEN PRINT*DU, G$(C);" ";: GOTO 460
456 :           IF G$(C)="-" THEN PRINT: INPUT"Grade";G$(C) : GOTO 466
458 :
460 :       NEXT C
462 :
464 :
466 :       IF F1 THEN DS(I)=VAL(G$(C)) :REM Put grades in Distr. Table
468 :
470 :       PRINT*6,N$
472 :       PRINT*6,NU$
474 :       FOR C=0 TO 30
476 :           PRINT*6,G$(C)
478 :           TTL =TTL + VAL(G$(C))
480 :       NEXT C
485 :       PRINT*6, STR$(TTL)
490 :
500 NEXT I
505 PRINT
510 :
520 DISK CLOSE,6
530 :
540 IF F1=0 THEN 9000
600 REM DISTRIBUTION ROUTINE
605 TTL=0 : INPUT"Which output device ";DU
```

```
610 PRINT#DV,"---"MSG$"---"
615 FOR I=1 TO NU
620 :   TTL=TTL+DS(I)
625 NEXT
630 PRINT#DV,"TOTAL ="TTL
635 MEAN = TTL/NU : PRINT#DV,"Mean ="MEAN
640 :
645 :
650 TV=0
655 FOR I=1 TO NU
660 :   TV=TV+(DS(I)-MEAN)^2 : REM Sum of Variances
665 NEXT I
670 SD=SQR(TV/NU)
675 PRINT#DV,"STANDARD DEV ="SD
680 :
685 REM -Sort into Descending order
690 FOR I=NU TO 2 STEP -1
700 :   SMALL =DS(1)
710 :   PT=1
720 :   FOR J=2 TO I
730 :       IF DS(J) < SMALL THEN SMALL =DS(J): PT=J
740 :   NEXT J
750 :   DS(PT) =DS(I)
760 :   DS(I)  =SMALL
770 NEXT I
780 :
800 REM -Print Grade Tally
810 :
815 :
820 I=1
825 FOR SC=DS(0) TO DS(NU) STEP -1
830 :   PRINT#DV; SC; TAB(3);
832 :
834 :   IF DS(I) < SC THEN 840
836 :   IF DS(I) >= SC THEN PRINT#DV;"1";  :REM Put tally &go to next
838 :   I=I+1
840 :   IF I>NU THEN 860                :REM Or exit if done
842 :   IF DS(I) >= SC THEN 836
844 :   IF INT(SC) = INT(MEAN +.5) THEN PRINT#DV;"--mean ";MEAN;
846 :
848 :   PRINT#DV;
850 NEXT SC
860 PRINT
900 :
9000 REM -Exit routine
9010 PRINT"File listing complete"
9020 PRINT
9030 FOR N=0 TO 800 :NEXT
9040 INPUT" Enter more grades to this file";A$
9050 IF LEFT$(A$,1)="Y" THEN 400
9060 :
9999 RUN"START
```

```
10 REM -EDIT
20 PRINT CHR$(12): REM CLR/HOME for ADDS
25 GOSUB 10000:REM LOCK SYSTEM
30 PRINT"  .......EDIT.......
35 :
40 PRINT:PRINT" PASSWORD-?";: POKE 8994,0
43 INPUT PW$: X=PEEK(8993): POKE 8994,X: PRINT
45 IF PW$<>"QWERTY" THEN 9040
50 :
55 INPUT"Which file do you wish to edit";PW$
60 GOTO 310
65 :
70 :
98 :
99 REM ------------------
100 REM -INPUT Subroutine .. Allows exit on '/' entry
102 REM    A$   Enter with message string, Exit with entry.
104 REM    F1   Flag =1, continue;  =0, Exit calling routine
106 :
110 F1=1
130 PRINT A$; :INPUT A$
140 IF LEFT$(A$,1)="/" THEN F1=0
150 :
190 RETURN
198 :
199 REM ------------------
200 REM -Data fields & variables:
205 REM      N$   student name            20 characters
210 REM      NU$  student number           6 digits
220 REM      G(C) student grades           3 digits
225 REM        C     which grade           0 to 30 grades
230 REM
240 REM      PW$    General input variable. Not saved.
242 REM      TTL    Student's ToTaL score
244 REM      NU     Number of Records in File as read from Record 0
246 REM      PV     Possible value for current score entry
250 REM      I      General counter for loops
255 REM      DV     Device number for output
260 REM      R      Record number; never geater than NU
265 :
298 :
299 REM ------------------
300 REM -Set variables. Program starts here.
310 DIM G$(33)
315 DV=PEEK(8993) :REM Output device
320 R =0 :REM Record number. Never greater than NU.
330 :
400 REM -Access file
405 DISK OPEN,6,PW$
410 DISK GET, R
415 :
420 INPUT#6,N$,NU$  :REM  Read whole record 0
425 NU =VAL(NU$)
430 FOR C= 0 TO 30
435 :   INPUT#6,G$(C): PRINT G$(C);" ";
440 NEXT C
445 INPUT#6,TTL$
450 :
455 PRINT#DV,"File "; N$;" with ";NU;" records
460 PRINT#DV,"If you wish to change file name or number of records"
465 PRINT#DV,"you may edit Record 0.  File name is the first data field
470 PRINT#DV,"and the number of records will be the second.
```

```
475 PRINT*DV
480 PRINT*DV
485 PRINT*DV,"  As each field of data is displayed, you may change it
490 PRINT*DV,"by simply typing the correction and a <RETURN>.
495 PRINT*DV,"  If no change is desired, then type <RETURN> with no
500 PRINT*DV,"change. A '/' exits the record changes.
505 :
510 :
515 PRINT*DV
520 INPUT"Which Record do you wish to change";R$
525 IF ASC(R$)>60 THEN PRINT"Enter the NUMBER of the record, please."
530 R= VAL(R$) :IF R<0 OR R>NU THEN PRINT"Record ";R;"; sure?":GOTO520
535 :
540 DISK GET,R     :REM  Read record R
545 INPUT*6,NA$,NU$
550 :
555 FOR C= 0 TO 30
560 :    INPUT*6,G$(C)
565 NEXT C
570 INPUT*6,TTL$
575 :
580 PRINT"RECORD ";R
585 :
590 PRINT*DV, NA$; :A$=" " :GOSUB 110
595 :    IF LEN(A$)>20 THEN PRINT"Too long!":GOTO 590
600 :    IF (A$<>"" AND F1=1) THEN NA$=A$
605 :
610 PRINT*DV, NU$; :A$=" " :GOSUB 110
615 :    IF LEN(A$)> 6 THEN PRINT"Too long!":GOTO 590
620 :    IF (A$<>"" AND F1=1) THEN NU$=A$
625 :
630 FOR C=0 TO 30
635 :    PRINT*DV,G$(C); :A$=" " :GOSUB 110
640 :        IF (A$<>"" AND F1=1) THEN G$(C)=A$
645 :        IF F1=0 THEN 660
650 NEXT C
655 :
660 PRINT*DV,TTL$; :A$=" " :GOSUB 110
662 :        IF (A$<>"" AND F1=1) THEN TTL$ =A$
664 :        IF F1=0 THEN 660
665 :
666 :
668 REM -Write to disk buffer
670 :
675 PRINT*6, NA$
680 PRINT*6, NU$
685 :
690 FOR C =0 TO 30
695 :    PRINT*6,G$(C)
700 NEXT C
705 :
710 PRINT*6, TTL$
715 :
720 PRINT:INPUT"Do again";PW$
725 IF LEFT$(PW$,1)="Y" THEN GOTO 515
730 :
735 DISK CLOSE,6
740 GOTO 9010 :REM -Exit routine
999 :
7998 :
7999 REM ------------------
8000 REM -Find record number from name of student
```

```
8010 :
8990 RETURN
8998 :
8999 REM ------------------
9000 REM -Exit routine
9010 PRINT"File changes complete"
9020 PRINT
9030 FOR N=0TO500:NEXT
9035 TRAP 0 :REM -Disable TRAP
9040 RUN"START  :REM -Return to menu
9998 :
9999 REM ------------------
10000 REM - LOCK SYSTEM
10010 POKE2073,96 :REM DISABLE <CTRL><C>
10020 POKE2888,0:POKE8722,0 :REM ALLOW NULL INPUT
10030 TRAP 9030 :REM -Exit routine ON DOS ERROR
10040 :
10050 RETURN



10 REM -REPORT
15 PRINT CHR$(12): REM CLR/HOME for ADDS
20 :
30 PRINT"  List names & grades from data files
40 PRINT"in vertical format with assignments and final grade.
50 :
55 :
60 PRINT:INPUT" PASSWORD-!";PW$
70 IF PW$<>"PASS" THEN 9999
80 :
100 PRINT :INPUT"Which port should this go to <1-8>";DV
110 IF DV<1 OR DV>8 THEN 100
140 :
170 :
180 PRINT"Which file do you intend to work with";
190 INPUT PW$
192 :
194 INPUT"Do you want names printed (Yes or No) ";P$
195 :
196 F10=0
198 IF LEFT$(P$,1)="Y" THEN F10=1
200 REM -Data fields & variables:
205 REM      N$    student name            20 characters
210 REM      NU$   student number           6 digits
220 REM      G(C) student grades            3 digits
225 REM        C      which grade           0 to 30 grades
230 REM      TTL$ student's total grade/possible grade in record 0
235 REM
240 REM      PW$     General input variable. Not saved.
241 REM      DV      Device used for output
250 REM      I       General counter for loops
255 REM      PG      Possible score, used to find % grade. =TTL$ record 0
260 :
300 REM -
310 DIM G$(33)
320 :
330 :
340 :
```

```
400 REM -Access file
402 :
404 DISK OPEN,6,PW$
405 :
406 DISK GET, 0      :REM Read Record 0 & save Possible Grade
408 INPUT#6,N$,NU$
410 FOR C=0 TO 30
412 :   INPUT#6,G$(C)
414 NEXT C
416 INPUT#6,TTL:  PG =TTL
418 :
420 PRINT#DV,"Class:  "N$ : PRINT#DV
422 NU=VAL(NU$)
424 :
426 FOR C=0 TO 30 :PRINT#DV, G$(C);" "; :NEXT C
427 PRINT#DV,PG
428 :
430 FOR I= 1 TO NU              :REM Read Student grades
433 PRINT I;
435 :   DISK GET, I
440 :    INPUT#6,NA$,NU$
442 :     FOR C=0 TO 30
444 :       INPUT#6,G$(C)
446 :     NEXT C
448 :     INPUT#6,TTL
450 :    PRINT#DV,NU$; :GOSUB10000 :REM Display data record///NO NAME$!
452 :       FOR C=0TO30:PRINT#DV, G$(C);" ";:NEXT C
456 :       PCT=INT(.5+1000*TTL/(PG+1E-10))/10
458 :       PRINT#DV,TTL;"/";PG;"=";PCT
460 :
485 :   PRINT#DV," "
490 :
500 NEXT I
505 PRINT#DV
510 :
520 DISK CLOSE,6
530 DISK!"HOME
540 :
9000 REM -Exit routine
9010 PRINT"File listing complete"
9020 PRINT
9030 FOR N=0TO500:NEXT
9999 RUN"START
10000 REM -OPTION TO PRINT NAMES
10010 IF F10=1 THEN PRINT#DV," ";NA$;
10020 PRINT#DV
10030 RETURN


10 REM -HELP
20 :
30 PRINT "(1) Initialize Student Files"
31 PRINT "(2) Enter/Add Student Names"
32 PRINT
33 PRINT "(4) Enter Grades/Scores"
34 PRINT "(5) Print Final Report"
35 PRINT
36 PRINT "(7) Print Short/Mid-Term Report"
37 PRINT "(8) Edit Student Files"
38 PRINT
100 REM ----------------------
140 INPUT"In which area do you need help ";I$
```

```
150 I=VAL(LEFT$(I$,1)): IF (I=3) OR (I=6) THEN GOTO 60020
155 PRINT CHR$(126);CHR$(28):REM CLR
157 FOR N=0 TO 10: NEXT: REM Delay to let terminal catch up
160 ON I GOTO 1020,2020,3020,4020,5020,6020,7020,8020,9020
170 :
180 :
1000 REM ----------------------
1010 REM Initialize student files
1015 :
1020 PRINT"  This selection allows the disk space previously allotted
1025 PRINT"by CREATE selection in the first menu to be set up for the
1030 PRINT"student files.  If you had not previously CREATED a file for
1035 PRINT"the class, you must do that now.  When asked for the number
1040 PRINT"of tracks in the file, divide the number of students by 23,
1045 PRINT"and allow one extra track for possible additions. Thus a
1050 PRINT"class of 25 would need 2 tracks minimum, and you might allow
1055 PRINT"a third for extras.  When students are deleted later, the
1060 PRINT"deleted records are ignored, but not physically removed.
1065 PRINT
1070 :
1075 PRINT"  File space may be reused later, as the student names can
1080 PRINT"optionally be retained.  This allows the same class list
1085 PRINT"to be used again for the next quarter of a semester or year
1090 PRINT"long class.
1095 :
1999 GOTO 60020: REM Exit
2000 REM ----------------------
2010 REM -Enter student names/add new students
2015 :
2020 PRINT"  Once a class file has been CREATED and Initialized, it is
2025 PRINT"ready to enter student names (and optionaly) numbers.  If
2030 PRINT"the school system currently uses numbers for students, this
2035 PRINT"set of programs allows their use, but does not require them.
2040 PRINT"Grade reports may be printed without student names.  These
2045 PRINT"reports may then be displayed to allow students to see how
2050 PRINT"they are doing, without anyone's name being attached to a
2055 PRINT"grade.  Students know their own student number, since it is
2060 PRINT"printed on each schedule and gradecard sent out from the
2065 PRINT"school.
2070 PRINT
2075 PRINT"  If it is necessary to interrupt entering names, the list
2080 PRINT"may be continued at a later time.  I made no attempt to
2085 PRINT"alphabetize the name list, since grades are also kept in a
2090 PRINT"gradebook as backup.  This way the list matches the ";
2095 PRINT"gradebook order.
2100 PRINT
2105 PRINT"  Normal entry is 'LAST FIRST I. ,NUMBER' order.  Note
2110 PRINT"there is NO COMMA between the first and last names. These
2115 PRINT"are stored as only ONE data field.  A six digit number
2120 PRINT"should be large enough for even the largest secondary school
2125 PRINT
2999 GOTO 60020: REM Exit
3000 REM ----------------------
3020 PRINT
3999 GOTO 60020: REM Exit
```

```
4000 REM ----------------------
4010 REM -Enter grades / scores
4015 :
4020 PRINT"  Once a file is CREATED and Initialized, and student names
4025 PRINT"have been entered, grades may now be entered.  The intent is
4030 PRINT"that all the student's grades may be entered at one time for
4035 PRINT"one assignment, quiz, or test.  This version makes no
4040 PRINT"distinction among the different types of grade.  If you need
4045 PRINT"a higher % of the score from homework, just grade a few more
4050 PRINT"homework papers.  All grades must be numeric.
4055 PRINT
4060 PRINT"  You MUST enter a score for all of the students at once.
4065 PRINT"If it should happen that some students have not yet
4070 PRINT"completed the assignment, enter a zero.  This can later be
4075 PRINT"changed by the EDIT function.
4080 PRINT
4085 PRINT"  When the last score is entered, the program totals the
4090 PRINT"individual student scores and stores it on the disk also.
4999 GOTO 60020: REM Exit
5000 REM ----------------------
5020 PRINT
5025 PRINT"  This option prints the whole class list with the student
5030 PRINT"number first, optionally the student names (Handy when you
5035 PRINT"are transferring grades to gradecards for the office records)
5040 PRINT"and then the string of grades/scores.  The last item is the
5045 PRINT"student's total score, the possible score, and the current
5050 PRINT"percentage.
5055 PRINT
5060 PRINT"  The first horizontal line on the chart is a list of the
5065 PRINT"possible scores for each of the tests/quizzes.
5070 PRINT
5999 GOTO 60020: REM Exit
6000 REM ----------------------
6020 PRINT
6999 GOTO 60020: REM Exit
7000 REM ----------------------
7020 PRINT  "If you need to present a mid-quarter report, a short form
7025 PRINT"of report card is helpful.  This choice prints the student's
7030 PRINT"name, lists her scores, prints the total and % average,
7035 PRINT"and optionally prints a letter grade.  No password is
7040 PRINT"included in this version.
7045 PRINT
7075 PRINT
7999 GOTO 60020: REM Exit
8000 REM ----------------------
8010 REM -Edit files
8015 :
8020 PRINT"  This program must be used carefully!!  It allows any grade
8025 PRINT"of any test, quiz, or student to be altered.  Be sure to
8030 PRINT"change the right grade!!  And DON'T let students even be
8035 PRINT"aware that there IS a password for this one, let alone what
8040 PRINT"it is!!!  Best yet, keep the disk at home.
8045 PRINT
8050 PRINT"Note that changing a grade does not change the student's
8055 PRINT"total score.  You need to change it also, or simply wait
8060 PRINT"until the next entry of grades with Menu choice 4 updates
8065 PRINT"all of the totals.
8070 PRINT
8075 PRINT"  You may proceed from one section to another without
8080 PRINT"looking at all the entries between by typing a '/' at
8085 PRINT"any time.
8090 :
```

```
8999 GOTO 60020: REM Exit
9000 REM ----------------------
9020 PRINT
9999 GOTO 60020: REM Exit
10000 :
10010 :
60000 REM ----------------------
60010 REM common exit
60020 PRINT:INPUT"Press <RETURN> to quit or enter 'H' for more ";I$
60030 IF I$="H" THEN RUN
60050 RUN"START",2000
```

AD$

[Editor's Note: Due to publication delays and other problems, some of these advertisements are rather old and the items offered may no longer be available. My apologies to both subscribers and advertisers.]

C3B with 80MB hard disk, dual floppies, Hazeltine 1420 terminal, CP/M, serial and parallel cards, 65U v1.44 with manuals. $500. Al Stark (703)-524-5455 DWH

For Sale: OSI Equipment including CPU's, hard disks, and dual disk drives, printers, cabinets, keyboards, and monitors. Call (301)-652-5424 9am to 5:30 pm EST Mon-Fri. M. Garvis

Wanted: C3 systems, 36 & 74 MB hard disks, terminals, printers, etc. Contact Mr. Joseph Clark 804-455-6666

For Sale: Three 527 boards (56K), one CA-9 parallel printer board with cable, one 555 board. Also five bare boards: 538, 555, 567, 570, 574. All in "new" condition. Call for negotiation. Would like a 620C bus extender for my C4P. Bill Atkinson, Wheeling, WV Home: 304-242-0337, Work 304-234-3159

Term-Plus

A smart terminal program running under OS-65D V3.3 which allows capturing and transmitting to and from disk. Term-Plus also supports error-free file transfers and cursor addressing on CompuServe. Memory size does not limit the size of files that can be captured or transmitted. Video systems get enhanced keyboard driver with 10 programmable character keys. 10 programmable function keys on both serial and video systems. Utilities included allow translating captured text files into OSI source format for BASIC and Assembler programs or into WP-2/WP-3 format, translating OSI source files into text files for transmitting to non-OSI systems, and printing captured text files. Runs on all disk systems, mini's or 8", except the C1P-MF. $35.00.

Term-32

Same as Term-Plus, but for OS-65D V3.2. Video system support includes enhanced keyboard driver, but uses V3.2 screen driver. $35.00.

Term-65U

Patterned after Term-Plus, Term-65U is a smart terminal program for OS-65U (all versions) running in the single user mode. Allows capturing text to disk files. Term-65U will transmit text files, or BASIC programs as text. The program will also send WP-3 files as formatted text and can transmit selected fields in records from OS-DMS Master files with sorts. Includes utilities to print captured text files or to convert them into WP-3/Edit-Plus or BASIC files. $50.00

ASM-Plus

ASM-Plus is a disk-based assembler running under OS-65D V3.3 that allows linked source files enabling you to write very large programs, regardless of system memory size. ASM-Plus assembles roughly 8 to 10 times faster than the OSI Assembler/Editor and is compatible with files for that assembler. ASM-Plus adds several assembly-time commands (pseudo-opcodes) for extra functionality. Included is a file editor for composing files that allows line editing and global searches. $50.00

Edit-Plus

Styled after WP-3-1, although not quite as powerful, Edit-Plus allows composing and editing WP-3 compatible files and to have those files printed as formatted text. Edit-Plus uses line-oriented editing, as opposed to the screen editing of WP-3, and also allows global search and replace. Edit-Plus fixes problems in WP-3 including pagination, inputs from the console, and file merging(selectable line numbers from the merged file). Edit-Plus can perform a trivial right-justification, but it does not support true proportional spacing. Requires OS-65D V3.3. or OS-65U V1.44 (specify) $40.00

Data-Plus 65U Mail Merge

A program to insert fields from OS-DMS Master files into WP-3 documents. Output can be routed to a printer or to a disk file for printing later or for transmission via modem using Term-65U. Insertions are fully selectable and are properly formatted into the output. Perfect for generating form letters. $30.00

PEEK (65)
The Unofficial OSI Users Journal
P.O. Box 586
Pacifica, CA 94044
415-993-6029

Bulk Rate
U S Postage
PAID
Pacifica, CA
Permit #92
Zip Code 94044

DELIVER TO:

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95	$ _____
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00	$ _____
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00	$ _____
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00	$ _____
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00	$ _____
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00	$ _____
( )	**Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00	$ _____
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00	$ _____
( )	**Assembler Editor & Extended Monitor Reference Manual** (C1P, C4P & C8P)	$6.95	$ _____
( )	**65V Primer.** Introduces machine language programming.	$4.95	$ _____
( )	**C1P, C1P MF, C4P, C4P DF, C4P MF, C8P DF Introductory Manuals** ($5.95 each, please specify)	$5.95	$ _____
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95	$ _____
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95	$ _____
( )	**How to program Microcomputers.** The C-3 Series	$7.95	$ _____
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95	$ _____

TOTAL	$_____
CA Residents add 6% Sales Tax	$_____
C.O.D. orders add $1.90	$_____
Postage & Handling	$ 3.70
TOTAL DUE	$_____

POSTAGE MAY VARY FOR OVERSEAS

Name _____
Street _____
City _____ State _____ Zip _____