BITS,BYTES&PIXELS

LIMA 99/4A USERS GROUP

January 1994 Volume 10 #1

THAT "OTHER" OFFICIAL TI CARD FOR THE PE BOX

reviewed by Charles Good
Lima Ohio User Group

HISTORICAL BACKGROUND:

Here's a trivia question. In addition to the disk controller, RS232, and 32K memory expansion cards which almost all of us have, what other TI PE box card could be purchased in 1983 directly from TI or from most TI dealers of that time? Answer..The P-CODE card, first shown to the public in January 1982 and listing for "only" $249.95 in TI's last 1983 catalog. With this card you can run canned disk or tape software written by others in UCSD PASCAL (also known as "p-Code" or the "P-System"). For a list price of $499.95 you could purchase the P-Code card AND development software that allows you to write your own USCD Pascal software as well as write TMS9900 assembly language programs. That's right, assembly code can be written without TI's editor assembler package using the P-Code card and associated software. In 1982 the P-Code card's potential seemed enormous. Yet now very few 99/4A users have or will ever need this piece of hardware. What happened to all that potential?

The following quote from the April 1983 issue of POPULAR COMPUTING states what many at that time believed to be the future potential of UCSD PASCAL. "Immagine how much easier choosing your microcomputer would be if any computer program on the market would run on any machine.... Your old programs would run on any new computer you might buy. That's what software engineers call portability, and it's the goal of an increasingly popular computer operating system called the UCSD p-System.... a universal language, a kind of computer Esperanto that would allow a program written for one computer to run on all the others." Or how about this ad quoted from the July 12, 1982 issue of INFOWORLD: "Whether it's a Z80, 8086/8088, M6800, 6502, or you name it, the p-System is portable across any popular microprocessor made anywhere today. And we don't mean just at the source code level either. We mean you can develop your program on any machine, then compile to object code (p-code), and its totally transportable.....the only OS that runs the same object code programs on all popular 8 and 16 bit microprocessors. No matter who makes them. No matter who uses them."

Sounds neat, doesn't it! There should be tons of software written on other machines will run on the 99/4A with a p-Code card. That was the promise. But there is almost no software. What happened? Where is the p-Code software? An interesting article by Stan Veit published in the Dec 1992 issue of Computer Shopper tells most of the story. In 1980 the University of California gave Softech Microsystems an exclusive licence to market and develop the UCSD p-System language. Softech got greedy, and what was once almost public domain technology suddenly became too expensive for almost everyone. For example, on page 10 of the January 1984 issue of BYTE it is noted that, "Softech Microsystems has announced a family of network software products based on its p-System operating system." Initially we have to already own the latest 1983 version of this operating system ($450 for EACH machine) in order to run this new software! Suddenly nobody was purchasing the p-System or writing any software for it.

There is another reason for the lack of 99/4A software that uses the p-Code peripheral. Software written in the UCSD PASCAL language is compiled into an intermediate code called P-code, and this P-code is exactly the same no matter what machine is used to develope the software. The same PASCAL source code typed into a variety of different computer types equipped with a P-System interpreter will compile into exactly the same P-code. Each different kind of machine has its own unique P-code interpreter which translates the P-code into the machine's native language for execution. The P-code card for the TI contains the necessary software in ROM to translate P-code into TMS9900 assembly directives. All of this suggests that you should be able to put disks containing UCSD PACAL software written on an APPLE IIe, a Mac, an MS DOS machine, a Commodore 64, an Atari 800 or ST, etc. into our 99/4A's disk drive and have the P-code card actually run the software! There should be hundreds or thousands of PASCAL programs out there for us P-code card owners to play with. Unfortunately this is not the case. YOU CANNOT PUT A DISK WRITTEN ON ANOTHER KIND OF COMPUTER INTO A 99/4A'S DRIVE AND EXPECT TO RUN pascal software. This is true even if the disk from the other computer contains the exact same sequence of P-code 1's and 0's that would be generated by our TI's if the PASCAL source code had been typed into the TI instead of the other computer. Why? The 99/4A disk controller only recognizes disks formatted by a 99/4A or Geneve. Our /4A's can't read "their" disks. This is true for most computer types. Even though there is only one kind of p-code, there is no universal disk format readable by all computers.

So.....what is "portable" about UCSD PASCAL? Well, with some exceptions you can type in the exact same printed program listing as source code into the PASCAL editor of ANY computer that supports the USCD PASCAL system. These program listings can then be compiled into P-code and run on different machines. Thus, if you start with a printed source code listing, the same software SHOULD run identically on different kinds of computers. Unlike BASIC, which is only sligltly similar from one machine to another, text oriented software written in UCSD PASCAL is supposed to be identical from one machine to another. I said "text oriented" because UCSD PASCAL for the 99/4A includes commands for graphics,

NEXT PAGE

speech, and sound that would not work on other computers.

Finally, an important reason for the lack of modern p-Code software is the development of versions of TURBO PASCAL for different computers. Both TURBO and UCSD PASCAL start with the same printed source code listings typed into an editor and then compiled. However, TURBO PASCAL compiles directly into the native assembly code of the host computer, bypassing the intermediate p-Code stage. Software written and then compiled from TURBO PASCAL runs much faster than software that compiles into p-Code. Good bye p-Code. L.L. Conner (317-742-8146) sells a version of TURBO PASCAL for the 99/4A that does not require TI's p-Code card. I have see software listings in TURBO PASCAL for IBM clones published in recent (1993) computer magazines. It has been years since I have seen any new listings in UCSD PASCAL.

TI'S OFFICIAL IMPLEMENTATION OF *UCSD PASCAL*

A complete UCSD PASCAL system for most computers includes an editor for entering program code or text documents, some file/disk/printer manipulation software, a compiler to convert source code into P-code, and a P-code interpreter. Only the P-code interpreter differs from one computer to another. TI chose to unbundle these parts and sell them separately. A minimum system consists of only the P-code card itself, which contains the interpreter in ROM. The TI version of the P-system editor and file manipulator, and the P-code compiler are extra cost items in TI 99/4A catalogs. With only the card 99/4A users could purchase and RUN commercial PASCAL software on disk or tape. Back when TI hardware cost BIG BUCKS there were some folks who purchased a PE box but couldn't immediately afford a disk system. In 1982/83 the $249.95 list price for TI's p-Code card (without development software) looked like a real bargain. Softech was at that time charging $450 PER MACHINE (with development software) for a P-system on every other type of computer.

AVAILABLE APPLICATION SOFTWARE FOR THE P-CODE CARD:

What's available that can be run with just the P-code card without the other "sold separately" P-system development software? TI actually sold PERSONAL TAX PLAN, an income tax preparation software package written by a group of CPA's who called themselves Ardvark Software. Personal Tax Plan was also available for the Apple II and probably some other personal computers of the day. I have an original copy of the TI version, ($99.95 in TI's last 1983 catalog) complete with documentation, all contained in a large three ring padded TI binder. This was based on 1983 tax laws and probably isn't usable for computing federal income taxes in the 1990's. Items covered include the joint filing credit if both spouses work and income averaging, neither of which are part of the 1992 tax code. For an extra $45 you could obtain a toll free phone number that give you up to one hour of over the phone help in using the software from a CPA. Additional hours are only $40. My docs state that this $45/hour service does NOT provide tax advice, just aid in running the software.

TI's last 1983 catalog also lists TI PILOT ($79.95), a language written in UCSD Pascal designed for use in computer aided instruction in schools. TI PILOT was never released, but I have a copy of the software and its user documentation.

Also in the works from TI for the P-code card was a word processor that came as a text editor and a separate formatter. The editor of this "never released" software was similar to TI's regular UCSD Pascal editor, and the formatter had capabilities similar to those of the TI Writer formatter. I have a copy of this UCSP Pascal word processor, and it has one BIG problem. You create text with its editor and then save the text to a disk file. You then switch to the formatter to print the formatted text. You can't print directly from the word processor's editor. The problem is that the formatter doesn't work with modern printers! One version of the formatter is for "Diable 1650s and 630s (or any variable pitch printer)" and the other version is for "TI810 printers (or any fixed pitch printer)". I have no idea what "fixed" and "variable" pitch printers are. Setting Pascal to look for a PIO printer (instead of the default RS232) doesn't help with this formatter. A word processor isn't much good if you can't print its text! The regular pascal editor, the one that is used to enter source code, makes a very usable 80 column word processor. Printing text from the regular editor is not difficult.

TI also had for the p-code card a "three dimensional" spread sheet called FREEFORM that allowed the user to create templates in rows, columns, and pages. "Pages" is the third dimension. Mathematical calculations can only be done in two dimensions (rows and columns), so the actual use of a three dimensional matrix is somewhat limited. The P-code word processor and spread sheet never made it into published TI catalogs, and were never released. This material is now in the public domain. The Lima group has this never released software in its library as well as a hard copy of the 1982 documentation. We also have the source code of FREEFORM as p-System text files.

OPERATING UNDER THE P-SYSTEM:

The P-code card turns the 99/4A into a completely different and mostly unfamiliar computer. Execution is SLOW, in part because much of the p-System must be loaded in sections into memory from disk. There is lots of disk activity. If you want your computer to behave like a normal TI there is a switch on the back of the P-code card that makes the card transparent to the rest of the computer system. If the P-code card is switched to the ON position, then when you turn on the console the P-code card captures the system. You don't see the usual TI title screen and if you have a CorComp disk controller you don't see the CorComp title screen either (CorComp controllers are notorious for

NEXT PAGE

usually capturing control of the 99/4A before other software can be run). Instead of a title screen the disk drives grind away and the computer beeps several times as the P-code card checks the first three drives to see if a P-system formatted disk is in the drive. After about 45 seconds of this either the disk in the first drive automatically begins to execute if it contains a Pascal file called SYSTEM.STARTUP. If not, a list of available Pascal commands appears at the top of the screen. From this list of commands the user can press "X" to Xecute a runable pascal file. This is how TI PILOT is started, for example. The user can also press "I" to reInitialize the pascal system if different pascal disks are placed in the active drives, or press "H" to leave pascal and return to the TI title screen for normal computer operations. Prompts for other parts of the total P-system are displayed, but most require extra cost software.

The only way to do any disk or file management is with the extra cost pascal "filer" software, booted from the first drive by pressing "F" from the startup menu. Ordinary TI disk managers such as DM1000, DSKU, and Funnelweb's DISK REVIEW do not recognize Pascal files and cannot be used to print such files or move them from one disk to another. Pascal files are stored on disk as "blocks" each corresponding to two regular disk sectors. To an ordinary TI disk manager a pascal disk says it contains only one file (usually named PASCAL) that occupies the whole disk. An ordinary TI disk manager must be used to initialize a new disk for use with UCSD Pascal, but the newly initialized disk has to be "zero"ed by the pascal filer before pascal can use the disk. The pascal filer, once loaded, displays its own set of prompts at the top of the screen, one of which is Z(ero).

The filer's V(olume) command lists all the parts of the P-system on screen as names and accompanying numbers. 1= CONSOLE. 2= SYSTEM (the system software in the p-code card's ROM). 4= name of disk in the first drive (known to most of us as DSK1). 5= disk name in second drive (aka DSK2). 6= PRINTER. 7= REMIN (I don't know what this means). 8= REMOUT (ditto). 9= disk name in third drive (DSK3). 14= OS (operating system, presumably part of the P-code card). 31= TAPE. You use these numbers a lot in file loading, saving, and management. To copy a file from the disk in the first drive to the disk in the third drive press T(ransfer) from the list of filer prompts. You are prompted for the file name and then asked "where". Typing "#4:,#9:" copies the file from the first drive to the third drive, which is device #9. If you don't know a file name you can display a disk directory from the filer by pressing either L(ook) or E(xtended directory).

The T(ransfer) filer command is also used to print a file to the printer (#4:,#6:), but there is a potential problem. The p-code card expects a printer to be attached to the RS232/2 port. (I think it expects a modem at RS232/1). Just in case you don't have a serial printer and the necessary Y cable needed to attach it to the "/2" of the RS232 port you have to change device #6 to "PIO". A utility program that comes with the editor/filer software package will accomplish this, but you have to run this utility program every time you boot the p-system. The RS232/2 designation for printer device #6 is in the p-code card's ROM and there is apparently no way to permanently change this. There is, however, a convenient solution:

THE MARVELOUS LIMA *UCSD PASCAL BOOT DISK:*

(This DSSD disk is #705 in the Lima group's software library. ANYONE (not just members of the Lima UG) can get a copy by sending a disk and paid return mailer to the Lima UG at P.O. Box 647, Venedocia OH 45894)

Put the disk in drive 1 and turn on the computer. First this disk sets the printer device name (device #6) to PIO. The user is then asked for the current date. The disk then boots the filer. All of this except entering the current date is done automatically upon system powerup. The actions taken by the Lima boot disk make using the TI p-System much easier and more convenient.

CREATING YOUR OWN PASCAL SOFTWARE:

You use the pascal editor to do this, which is called by pressing "E" from the main system menu. When booted the editor presents you with a new set of possible prompts. You press I(nsert) to begin entering your pascal program. You can also use the editor as a text editor to create documents (similar to using the EA module's editor as a word processor). Like TI Writer, the pascal editor gives you access to an 80 column work area by windowing left/right. The keypresses that control the pascal editor are, however, often quite different from those of TI Writer, perhaps because the pascal editor was originally developed for use with the 99/4 which has no function key. On a 99/4A left/right windowing is controled with FCTN/7 and FCTN/8. Line delete is FCTN/9. Once the source code of a program has been typed into the editor it is saved to disk as a ".TEXT" file. You then exit the editor and press C(ompile) from the main Pascal menu to boot the compiler software and turn the text file into p-code. The resulting ".CODE" file can be run at any time from the main pascal menu by pressing X(ecute) and then specifying the drive number and file name.

UCSD Pascal is not a common programming language today. However, type in pascal programs can be found in computer books and magazines from the 1982-84 time period. For example, a rather long type in income tax record keeping program written in pascal appears in one of the early 1983 issues of BYTE.

Pascal is a "structured language" whose syntax follows very precise rules. Code written by one person is usually easy to understand by another person who is familiar with

NEXT PAGE

pascal. This often is not true for BASIC program listings. Large software applications to be written in UCSD Pascal can be divided into a number of small segments each assigned to a different programmer for development. These separate pieces of programming code can be individually tested and then because of the logical syntax of pascal it is relatively easy to combine the separate pieces into large blocks of program code. A pascal source code listing appears to be written in outline form, with various degrees of "indentation". Short independent pieces of code (indentented the most from the left margin of the listing) can be combined to form larger units of code (indented less) which in turn can be combined into larger code units. The effect is similar to using Extended Basic subprograms developed by others in your own programs. Because of its "logic" and "structure" pascal is sometimes taught in introductory university level computer programming classes as an example of the concept that software written in any computer language should be logical and understandable by others (I hope you understood that!).

TECHNICAL INFORMATION ABOUT THE P-CODE CARD:

I have had no problem with text oriented operations of P-code card software, even using a system with a Horizon Ramdisk set for CRU address 1000. Since the p-code card takes control of the computer BEFORE the CorComp disk controller, one might think that that the P-code card has a CRU address of 1000, but I don't know this for sure. I have run into a strange problem which I think has to do with my AVPC card. With the AVPC card in my system I have no access to the console's sound chip. This means no beeps, no music, and no speech synthesis when running TI PILOT. The computer locks up every time sound is expected. Music and speech work OK in pascal on my non-AVPC system even though there is a CRU 1000 ramdisk in this system. Strange!

The following information is part of a 1987 article by Anders Persson of Lund, Sweden. The complete text is published in the Feb-March 88 issue of the MANNERS NEWSLETTER, which is available on loan to members of the Lima UG.

"The P-Code card has a total memory capacity of 60 kilobytes. This memory consists of 12K ROM and 48K Grom.

"ROM: The ROM memory is located at 4000-5FFF. 5000-5FFF is paged in two pages, but the lower four kilobytes are always the same. The paging is done with CRU bit 1FB0. Resetting the bit to zero gives access to the normal page. Setting the bit to one switches in the extra page.

"GROM: The GROM chips contain the files that are located in unit #14: (DS:). they also contain various data and assembly code, which is loaded into RAM when the card is initialized. The p-code GROM is accessed just like the console and module GROM chips, but at different addresses."

The article then goes on to describe in great detain workspace addresses and a map of 8K RAM under the p-system.

CONCLUSION:

I traded my Thermal Printer (device TP) for my p-Code card. I find the p-Code card every bit as useful as my TP was. There is very little p-System software. I don't think any commercial software for any computer is written in p-Code any more. As of December 1993 the TI P-Code card was available for $39.95 from TM Direct Marketing (phone 800-336-9966) and for $25.00 from Cecure Electronics (phone 800-959-9640). Both prices are plus shipping and both these dealers accept phone credit card orders. The Cecure price is for a "limited time".

DONE

BITS, BYTES & PIXELS
Published by Lima OH
99/4A User Group

Material contained herein may be copied by any user group as long as credit is given. DV80 files of most articles in BB&P can be obtained by sending a disk and return postage.

ADDRESS- P.O. Box 647
Venedocia Ohio
45894

Published monthly except July and August

FUN WITH FUNNELWEB

by George J. Clark
Lima Ohio User Group

As you can see from the enclosure, I am really "having a ball" with funnelweb v5.0. I am truly grateful for the SSSD you sent me (Anyone reading this can have a stand alone SSSD disk with a fully functional Funnelweb v5 40 column editor that boots from XB by sending $1 to the Lima User Group, P.O. Box 647, Venedocia OH 45894).

I was lining out a rather crude map by hand when it occurred to me that here was an opportunity to experiment with FWEB v5.0. So I produced the enclosed map. I found the HELP screens very useful for remembering the proper character to print a bottom left corner, etc.

The Mcgovern's state that the Formatter cannot be used with the IBM character sets and do not extend any promise that it will ever be rewritten to enable it to handle them. No problem! You can "format" your PF file using printer codes very easily.

First I pre-set my printer for Margins etc. I copied the SSSD disk you sent me onto a DSSD disk, changed LOAD to LOADA and added the enclosed LOAD mini program which reminds you to:

TURN THE PRINTER ON! if it is not on line, and asks

PRE-SET FOR DRAWING? (Y/N). If N then it directly loads FWEB...if Y then it sets: Clear the printer buffer; French International Charset (I live in Quebec, a French speaking Province); Left Margin; right Margin; Elete pitch ('cause I like it!); Skip Perforation (6 lines); Epson Graphic Character Set 1; and SINGLE Direction printing (bidirection produces zigzag effect). Then it loads FWEB.

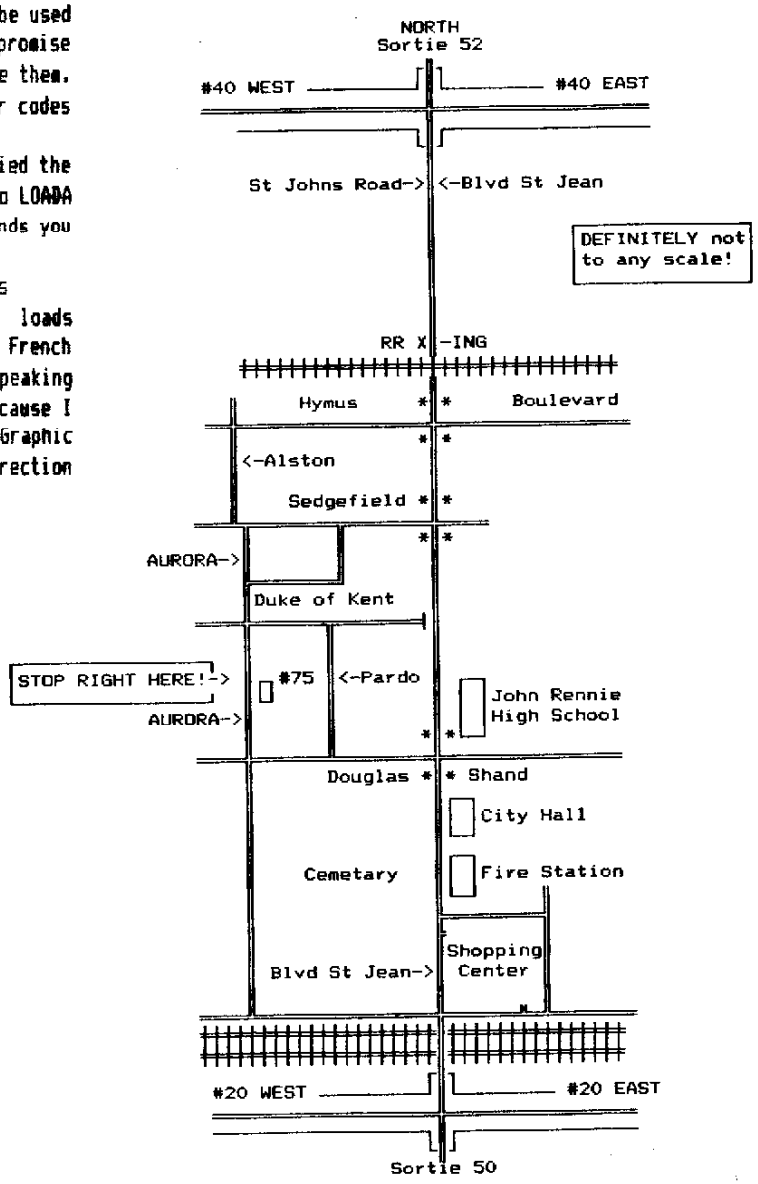
```
100 ! SAVE DSK1.LOAD
110 DISPLAY AT(12,6):"TURN P
RINTER ON!" :: OPEN #1:"PIO.
CR" :: PRINT #1:CHR$(0):: CA
LL CLEAR
120 PRINT "PRE-SET FOR DRAWI
NG? (Y/N)"
130 CALL KEY(5,K,S)
140 IF (K=89)+(K=121)THEN 16
0
150 IF (K=78)+(K=110)THEN 19
0
160 PRINT #1:CHR$(27);CHR$(6
4);CHR$(27);CHR$(82);CHR$(1)
;CHR$(27);CHR$(108);CHR$(8);
CHR$(27);CHR$(81);CHR$(70);C
HR$(27);CHR$(77)
170 PRINT #1:CHR$(27);CHR$(7
8);CHR$(6);CHR$(27);CHR$(116
);CHR$(1);CHR$(27);CHR$(85);
CHR$(1)
180 CLOSE #1
190 RUN "DSK1.LOADA"
```

Note that it selects the EPSON grahic Character Set and NOT the IBM set! is normal capabilities as there are many differences in the EPSON/IBM codes ... EPSON for Elete is ESC "M" and IBM is ESC ":", EPSON ESC "P" for PICA vs IBM ESC "p" for PICA whereas ESC "p" in the EPSON mode is for PROPORTIONAL SPACING! (BB&P editor's note: Proportional spacing is not available on all printers. When set it justifies the right margin. You don't need to use the Formatter to right justify using such printers. Just PF from the editor and text is justified.)

DO NOT set your DIP switches for IBM!!

To set for Epson Graphic Set 1 use ESC/"t"/1

To set for SINGLE DIRECTION printing use ESC/"U"/1



DONE

MORE ABOUT GARY BOWSER

As a followup to the page of letters published in the December 93 issue of the Lima newsletter complaining about Gary Bowser not delivering products paid for years ago, and/or delivering defective products, we are publishing the following uploads to Delphi's TINET. The first is from Gary Bowser. The second is a reply from Tony McGovern. If I (Charles Good) understand Gary's message correctly, he is saying 30 more people have to send him money for TIMs that have as yet not been manufactured. Then maybe he will make some TIMs for those who have already paid for one. He's kidding, right?

==========

41520 8-DEC 01:41 New Uploads
RE: Hi There (Re: Msg 41478)
From: TINET To: GLOBAL01 (NR)

There were many TIMs in the hands of serious programmers, when the device first hit the market in early 1990, we ship out over 25 units to the "then" current programmers, many of when have left, and the rest never wrote any software. It is true we hit the market after the AVPC card and long after the Mech. unit, but as far as we know we have shipped out more TIMs then both the AVPC and Mech. combined.

Out reasons for picking the 9958 over the 9938, was for the improved YJK display mode, and the easier hardware interface in hooking up digitzers and other devices.

Its true thre is about 40 outstanding orders for the TIM/SOB unit, and after a hard two years it looks like things are starting back on the right track, we have moved into bigger place, and are hiring staff to handle the orders/building/shipping of the unit, and soon will be ordering the needed stock and parts. Producing the TIM/SOB takes alot of dough, and each production run runs well over $9000 in parts, not counting labour,etc. After producing a number of production runs in 1990, we were out of TIMS by early 1991, due to a number of factors, mainly overrun in cost in R&D in two projects we were helping out for other TI'er businesses, and in lost of orders due to people telling customers that we were not shipping goods out fast enough, our monthly supply of orders quickly died up. The reason for slow shipping in 1990, was twofold, lack of enough time and space to handle the growing amounts of orders for TIMs, and too much manpower spent in new R&D in projects undertaken by OPA to help other companies. I could kept going into details, but most people just don't seem to understand or want to listen to our side. TI'ers calling wolf before it happened caused the REAL crunch of orders in early 1991, which caused more users to scream which caused less orders. This type of business runs on cash-flow, the sale of orders gets more orders, which brings in more money which is needed to kept the ball rolling smoothly. Even today, as little as 30 fully-paided orders for TIMs would get everyone a TIM including those that have ordered, as this would bring up the money-pot, enough to start production again. We can't make just one or two TIMs a time, the $179 price is based on the fact that their will be many orders. We would have to sell the TIM for over $300 if builded in small lots of 10 or less.

ROS 8.14 will not be upgraded. I suggest if you want better use of your Horizon to buy our new RAMOS for all ramdisks. It supports many new features, and I am currently run a HRD3000 with over 13000 sector drive. Yep, over Its strange, you see most of good reputation in the ROS 8.14, I should thank you for that comment, and most don't even know we wrote 8.14. I don't know why you think it is a mixed blessing for not being OPEN soft-ware as I can name any currently marketed software in the TI world, to be OPENed, we fell it is better to have the DSR closed, as this side of the TI is one of the hardest areas to handle, and writing ROS 8.14 was a real job, but still it was not very good, and broke many rules. The all new RAMOS system, took five programmers closing working together for over 18 months to bring it to its finally stage, and solving every problem with every card out there was a good learning job, and I can say for sure our company has become the best DSR programmers out there, and now know everything there is to know regarding the TI design, I wish I could say the same about the SCSI effort. When you see RAMOS in action, you will be amaze in its speed and power, and ease of use.

That's enough for now, TTYL (GarY)

==========

This following will be posted - Dec 10th -- Tony

Gary,

I must say that I am disappointed that there will be no upgrade to ROS 8.14 in the future from OPA, nor release of its source code. Nor am I impressed by an attempt to move the customers along to a new product when the old one has not been done all that well. It is unpleasantly reminiscent of widespread practice in the PC world where buggy or poorly written (usually bloated and sluggish) programs are rarely fixed for the users, who are then expected to lay out even more serious money for "new" or "improved" versions or products. ROS 8.14 does need a "corrective service" upgrade even if just to provide DSQD on large HRDs to scale in size. With what you are claiming to have done on a big HRD-3000, it should be be a simple matter to add DSQD to 8.14.

My comments on the quality of programs from OPA have been quite restrained and have concentrated on good aspects, with constructive suggestions for improvement. It seems time in this exchange of views to be a little more forceful. I take as the base requirement for a good HRD type ROS that it emulate a physical disk drive with as high a performance as possible. The second, well established over and above the basic disk function, is as a boot-capture loader. These were established by the pioneering work of the Miami group. They did not get everything right, but they were the pioneers! The writers of Vn 8.x should have gotten it right, but have

NEXT PAGE

fallen well short of extracting maximum performance, versatility, and system compatibility as a basic RAMdisk, while crowding in various proprietary features of marginal utility. Some of the extras are good, like the Myarc FDC style direct to CPU RAM transfer. All in all though I am not all that impressed with either the system design sense or detailed coding ability demonstrated in ROS 8.14, and it gives me no confidence that the RAMOS being touted will be of any better essential quality.

It is certainly news that any TIMs were in the hands of "serious programmers". Australia may be a long way from anywhere else, but even so I was never aware of any heavy-duty programmers working with the TIM. As you indicate, there have certainly been no 9958-specific results to prove otherwise. All the Funnelweb development has been and is still done on a pre-production AVPC which I understand had been had been in the hands of Barry Boone for a year or more prior to its being sent over here. As for SCSI development, I am waiting on the full DSR as expectantly as anyone else. But I do know the lowest hardware driving layer of the DSR was done on the prototype hardware with real expertise and very expeditiously back in Jan 93 by Will before he moved to Sydney.

Tony McG

DONE

Assembler Executing #2
By Bob Carmany

I reckon that it is time for another installment of my continuing adventures into A/L programming. Fortunately, most of this early material is fairly easy to understand. I think we examined the concept of registers and number conversion the last time so I'm going to look at how A/L statements are written. For the time being, the structure of the statements is all I can handle! Back to the book!

It says that A/L statements can have up to four fields. Strangely enough, not all of them have to be present in a statement. In order, they are: Label, Op(eration) Code or Directive, Operand, and Comment. A label, the book says, is only required when you want to refer to a statement from another statement. I reckon it is sort of analogous to a CALL statement in XB -- a group of statements that make up a routine. At least that's the sense of it that I can discern! A label can be from 1 to 6 characters in length and the first character must be a letter (A to Z). A label is the first entry in a statement and each statement can have only one label. Logical enough! I think that I can understand this!

The next part of an A/L statement is the Op-Code or directive. That tells the program what operation to perform on the third field. A simple Directive is MOV for move word. Any of the Op-Codes can be used but they must be spelled correctly (of course). Even my foggy mind can understand that!

The third field is the Operand field -- the item to be manipulated by the Op-Code. There are some rules to follow with this lot as well. Whew! Talk about structured programming!! More than one operand must be seperated by a comma and spaces can only appear in an operand if they are between a pair of apostrophies (the A/L equivalent of parentheses).

The last field is the Comment field. That is a freelance entry to explain what the statement was supposed to do --- and I emphasize SUPPOSED TO! It can extend to the end of the line.

There are some general rules about the form of A/L source code. Each of the fields must be seperated from the other by at least one space. By convention, the fields are generally aligned when they are written. Oh yes, you can add general comments to your programs by using the asterisk "*" at the beginning of the line. It functions as a REM statement in XB.

Although any text editor that outputs a D/V 80 file can be used to create source code, the best that I have found is F'WEB in the ASMode. All of the tabs have been fixed and the word-wrap has been turned off. In addition, the code is saved without the tab settings. I find it very easy to use when I'm about to "borrow" a bit of source code from an article somewhere. I haven't gotten far enough with this stuff to write my own code yet!

Ok, let's see what a line of A/L source code might look like. This is for structure only so I have just picked some stuff out of the book:

```
START MOV @@A,R0 Move A to Register 0
```

Here you have a Label, Op-Code, Operand, and Comment field.

Now, it is time to take a look at a comparison between XB (which I understand) and A/L (which I don't). This example is a bit primitive but it is about all that I can muster right now.

```
100 DATA 1,7    Defines the two data items 1 and 7
110 READ X,Y    Assigns the value 1 to X and 7 to Y
120 Z=X+Y       Adds X and Y to get the value of Z
```

The same program in A/L would look something like this:

```
X       DATA  1       Assigns the name X to the value 1
Y       DATA  7       Assigns the name Y to the value of 7
Z       BSS   2       Reserves 2 bytes of memory for the value Z
        MOV   @X,R0   Moves X to work Register 0
        MOV   @Y,R1   Moves Y to work Register 1
        A     R0,R1   Adds Register 0 to Register 1
        MOV   R1,@Z   Moves the sum to Register 1 replacing
                            the number there
```

NEXT PAGE

Bloody amazing! I learned how to add 1 and 7 in a program. Watch out Ron and Tony --- I'm on the way!!

All you have to do from here is run the source code through the Assembler and you have a D/F object code file that you can LOAD AND RUN. Type in the input filename, an output filename and, after pressing <FCTN-6> the ASSEMBLER EXECUTING message appears. If you are lucky, you get a "0000 ERRORS" message when the Assembler is through.

That would be a new experience for me! Even on stuff that I thought that I had typed in accurately from a printed copy I usually get "double digit" errors! I once typed in about 300 or so lines of source code directly (I thought) from a couple of issues of MICROpendium. It took me two days just to get the typos corrected in that mess!! After all of that, the silly program wouldn't even run. You can imagine how "happy" I was about that! Just wait until I start writing my own programs --- triple digit errors may not be out of the question!

So far, I have to admit that A/L isn't nearly as difficult to understand as I thought that it would be. Of course, I haven't really started to do any substantive programming. I've managed to display a few bits of text on the screen and manipulate some numbers here and there but nothing that would qualify as a program. We'll have to see what the next month will bring!

DONE

METRONOME Operating Instructions

(The following new public domain software by Bruce Harrison is available as disk 870A in the Lima software library)

Metronome does exactly what its name implies, providing a stable "tick" at a selected number of beats per minute for musicians practicing their instruments. Two versions are provided, both in the form of Option-5 Editor/Assembler Program Files. The one called METRONOME is designed for use on U.S. systems, with 60 Hz NTSC Video systems. The second version, called METROEUR, is designed for operation on European systems, with 50 Hz PAL Video systems. Both versions will run on any TI-99/4A computer, but the timing will be accurate only if run on the appropriate system. For example, if one runs the EUR version on a U.S. system, the number of beats selected will take 50 seconds to complete. (e.g. 60 beats per minute will produce 60 ticks in 50 seconds.)

Special loaders have been provided under the names XBLOADUS and XBLOADEUR. These are Extended Basic programs that will load up and run the U.S. and European versions, respectively. When loaded in this manner from XB, the METRONOME must be exited through Function-= (QUIT). These loaders will operate from any disk drive, including RAMDISK.

The User Interface has been made as simple as humanly possible. On startup, there will be a title and prompt on the screen, and the cursor will be blinking in an entry field. Below the entry field will be the appropriate range for this version. Typing in a number in the acceptable range will cause the computer to start producing ticks at the selected rate. For each tick heard, there will be a brief flash of the cursor near the bottom of the screen, to provide visual confirmation in case the ticks can't be heard over the player's own music.

While the metronome is running, pressing any key on the TI keyboard will stop the ticks. Pressing any key except Function-9 will re-start the ticks. Pressing Function-9 when the ticks are stopped will clear the entry field so a new rate can be entered. Pressing Function-9 with the field clear will exit to E/A. The program can be exited at any time by Function-= (QUIT).

In both versions, the timing of ticks is based on the vertical interval rate as indicated by the VDP Interrupt timer. Internally, the program takes the number you input and divides either 3600 (U.S. version) or 3000 (European version) by that number. This gives the computer a number of counts on the VDP timer between ticks. The tick sound itself takes three intervals of 1/60 or 1/50 second, then there's silence until the selected count finishes. Checked against an ordinary "quartz" wristwatch, this yields perfectly accurate results. Rounding errors will in some cases make the timing off slightly, but not by more than 1/100 second per tick.

This software, developed by Bruce Harrison, is provided as Public Domain, and may be distributed freely anywhere in the world. Copying fees are at the discretion of the copier. Comments or suggestions to:

Bruce Harrison
5705 40th Place
Hyattsville MD 20781 (U.S.A.)

DONE