NORTHWEST COMPUTER NEWS

3-8 SEP 78

NORTHWEST COMPUTER SOCIETY
P.O. BOX 4193, SEATTLE, WA 98104
284-6109

Single copy 47 cents / Subscription & Membership $7.00

MY SYSTEM

by David Koh, MD

My system started in early 1976 with the purchase of a Southwest Technical Products CT-1024 TV Typewriter. Unlike most people who buy the computer first and then look for a terminal later, I knew I'd be impatient to start talking to my processor when I got it. The CT-1024 went together quickly and as many others have found, it worked the first time.

Encouraged by this I went ahead and bought the Southwest 6800 computer kit. There were other reasons for choosing this system besides Southwest's good track record. These included clean design, an easy to use instruction set, built-in ROM monitor and plain old economics. Not only did the basic system include a power up-and-go monitor, but it threw in a serial interface and 2K (now 4K) of RAM for less than what others wanted for their mainframe alone.

The quality of all parts and PC boards was uniformly excellent and the computer went together easily. I'd like to report that it worked the first time but in my haste I installed two integrated circuits backwards in their sockets. These were quickly reversed (after they had cooled enough to touch!) and in minutes MIKBUG was operating on my CT-1024. This, incidentally, touches on my only criticism of Southwest Products. They neither supply nor recommend the use of IC sockets in their kits. They claim increased reliability when the chips are soldered to the board and point to this method as the Industry Standard. Well, that's fine if you plan on using the Industry Standard way of repairing equipment — namely, pull out the bad board and plug in a new one. For those of us with more limited budgets, I consider IC sockets cheap insurance.

My original purchase included 8K of memory and this was brought on line with minimal difficulty. I soon grew weary of typing in programs by hand so my next project was a cassette interface. I homebrewed my own version of Don Lancaster's Bit Boffer Kansas City interface described in the March '76 issue of BYTE. This is similar to the SWTP AC-30 but produces a much cleaner signal on tape by virtue of digital sine wave synthesis as opposed to simply filtering square waves. The result is less distortion and more reliability especially using inexpensive cassette recorders. By adding a few modifications I was able to run the interface at 2400 baud as well as at the 300 baud KC standard. At the higher speed using a binary loader, 8K will load in about 30 seconds.

As more and more software became available for my system it became clear that I needed additional memory. 4K BASIC gave way to 8K BASIC and several good editors and assemblers were released. I reasoned that 16K would barely suffice. It was in September 1976 (also referred to as the Paleolithic Age of personal computing) that my system became operational. There were no 8K or 16K boards available and 4K boards were still pretty expensive. I found that you could expand a Southwest 4K board to an 8K board by just piggybacking memory chips. The decoding was already present on the board and required just a few wiring changes. For the price of 64 2102's I had my 16K of memory. Aesthetic it wasn't, but cost effective? Definitely!

You've probably guessed that I'm a tinkerer at heart. In some way or other I've modified every board in my system. Not that there is anything wrong with Southwest design. Quite the contrary. The clean architecture and roomy layout invites custom enhancements.

The CT-1024 now has 64 characters per line, scrolling, and lower case. It is packaged along with a surplus 12" monitor inside an old Sanders terminal cabinet. Since it communicates through an RS-232 serial interface I also use it with my homemade acoustic coupler (the Tin Can Special) to access a local time-sharing service.

My next addition to the system was a homemade graphics board which has a resolution of 256 by 160 dots. The 5K of RAM that this represents is usable as standard memory as well. To go along with this graphics system I interfaced a surplus flatbed X-Y digitizer so that I could draw or copy pictures without having to hand load data. I've also written some software graphics handlers and doodling programs. A couple of examples of the system's capabilities were published in BYTE (Oct. '77 pg. 173). Complete details of the graphics board will appear in a two part article in Kilobaud this fall.

My software library includes SWTP BASIC, Motorola's editor and assembler, and TSC's Text Editor/Processor. The latter is a powerful word-processing package similar to RUNOFF that features paging, title centering, and right justification (to name just a few). I use it with my Dura Selectric printer to prepare letters and manuscripts.

One of the nice things about this hobby (or one of the curses, depending on your point of view) is that you never run out of things to do. There's always one more peripheral to interface and one more program to write. The limiting factor always seems to be time. My "back-burner" projects which are in various stages of completion include: 1) designing a PROM burner to complement my homebuilt 2708 EPROM board; 2) interfacing a high speed paper tape punch and reader; and 3) installing the Sublogic 3-D graphics software package in order to experiment with flight simulation.

I find personal computing to be a very rewarding pastime. Whatever you put into it you're sure to get back. The point is not to be afraid to get started for you can participate at almost any level. You don't need to spend a great deal of money nor go to engineering school. I didn't. I think you'll find it a fascinating and educational way to spend time.

Isn't that what a good hobby is all about?

***

The system: A Southwest 6800 with 29k of memory, EPROM, graphics, X-Y digitizer, cassette storage, modified CT-1024 terminal, homebrew acoustic coupler, and Dura Selectric printer.

MEETINGS

The Northwest Computer Society meets at the Pacific Science Center, in Seattle, on the first and third Thursday of each month at 7:30 PM. The first meeting of the month normally is held in room 200, on the East side of the Science Center Court. This meeting usually features a formal presentation by a speaker or speakers. The second meeting of the month is normally held in the math room, at the Southeast corner of the Science Center Court, two flights down. This meeting is usually more informal with freewheeling discussion and problem solving.

Thursday, September 21 — Math Room
Informal meeting with no speaker scheduled. Bring your things for "show and tell."

Thursday, October 5 — Room 200
"Macros, Transportable Languages, and Machine-Independent Code," Chuck Deiotte, Boeing Computer Services

Thursday, October 19 — Math Room
Informal meeting with no speaker scheduled.

MY COLUMN

by Ken Berkun

Being concerned, as I am, with everything under the sun, and specifically anything that affects me in any way, I was interested when it was pointed out to me that the club may not be serving all segments of the Northwest area micro computer users. The main area being ignored that I'm concerned about right now is the business community. The small business community, to be specific.

Primarily we are a club made up of hobbyists, and some professionals who are into using the computer in the home. But most micro computer manufacturers and dealers are turning to the business person, primarily because there is more money there. Well, there is more than money there. There are users, lots of them. I'd like to see more representation of this segment of the community in our club. These people can help us in many ways, from running the club to providing input for ideas for program development, to access to more equipment.

Likewise there are many things we have to offer them. Our programming and hardware expertise, which is almost unmatched anywhere else, our ideas, and our fellowship, which is important when dealing with certain manufacturers.

The problem, then, is how to woo these people into our club. Where are they? They exist; someone is buying all that equipment. Perhaps you know some of them. If so invite them to a meeting. Current machine owners, or prospective owners. It's to everyone's benefit. And there must be other things we can do. If you have any ideas give me a buzz. My numbers are 655-9945 at work, and 255-6429 at home. I'd like to get the club expanded in that direction, and I'm sure others would also.

For those of you wondering how the small business community might affect you personally, let me toss out this little gem: There is an awful lot of software that needs to be written to get a small computer to be useful to a business person. Likewise there is an awful lot of non-working hardware bought by unsuspecting business people. Someone needs to write that software, and someone needs to fix that hardware. Who? Why not us? We are the best pool of knowledge in the field that is available anywhere, due to our diverse backgrounds. And what's in it for us? Simply put, besides the experience, money. The business world is in the habit of paying for what we usually do for free with friends. And they pay well,

Continued on p. 4

Northwest Computer Society
P.O. Box 4193
Seattle, Wa 98104

Bulk Rate
US Postage
PAID
Renton, Wa. 98055
Permit No. 282

SORCERER™ COMPUTER SPEAKS YOUR LANGUAGE

Sorcerer™ Computer is User Programmable Therefore Program Selection is Unlimited

Flexibility is the key. The Sorcerer™ computer gives you the flexibility of using ready-to-run, pre-packaged programs or... Doing your own thing and personalizing the programs for yourself. Whichever you chose, the Sorcerer is the personal computer that speaks your language.

Cassette Tapes keep the expense of your program library in line with your budget for personal computing.

User Programmability provides a challenge to your creativity. Sure . . . you can play pre-packaged (canned) programs on the Sorcerer™ computer which are great for an introduction, but, the real fun comes in personalizing the programs for yourself. Sorcerer allows you the freedom to explore the unexplored, be in control, and discover your capabilities.

Unique ROM PAC™ Cartridges look and handle like 8-track stereo cartridges, but inside read-only memory devices, store computer programming languages, operating systems or whatever. Programming languages are like religions and some people prefer a modern new one while others prefer an older established one. The selection of a programming language in a ROM PAC cartridge is the perfect solution.

Standard Basic is interpretive which means you get immediate response from the computer as you input your program. This is very helpful to the first time programmer in finding mistakes and debugging the program as you load it in. This also allows your Sorcerer to be a super programmable calculator as well. Standard Basic was written to be as efficient as possible with a maximum number of features in the minimum amount of memory. Multiple statements per line, complete string manipulation and multiple dimension arrays are included with calculating speed approximating 700 floating point additions per second with seven decimal digits of precision. Standard Basic resides in 8K of read-only memory physically located inside the ROM PAC cartridge.

A **Power-on Monitor Program** provides keyboard control of the Sorcerer at turn-on. This means you can immediately start program execution, cassette tape read and record, examine or modify registers and memory, move blocks of memory, set program break points, and do a self test for operational condition. Or, if you choose, the power-on monitor will default to a ROM PAC cartridge for program execution. The power-on monitor program resides in 4K of read-only-memory and is always present in the computer.

Specifications

Features

Processor Type: Z80

Processor Clock: 2.1 MHz

Serial I/O: RS232, 300 or 1200 baud, 25 pin "D" type connector.

Parallel I/O: 8-bit input and output latched and buffered port with handshaking, 25 pin "D" connector.

Memory: Read-only memory (R.O.M.) 4K byte. Power-on monitor program connection for 16K ROM PAC program cartridge. Random access memory (R.A.M.) for 8K byte expandable to 32K byte.

Expansion: Edge card connection to S-100 bus expansion unit.

Cassette I/O: Dual recorders, 300 or 1200 baud data transfer rate, remote control of motor on/off.

Video I/O: 30 lines of 64 characters or 1,920 characters full screen, 128 full ASCII character set. 64 defined graphic characters and 64 user defined characters; alternatively all 128 graphic characters may be user defined. 512 (horz.) x 240 (vert.) graphic resolution.

Automatic scroll, erase end of line and end of screen, delete character, erase screen.

Cursor Home, Up, Down, Left, Right.

Cabinet: Dimensions 19.25" x 13.0" x 4.0", weight 13 lbs.

Power: 110-125 VAC, 50-60 Hz 205-230 VAC, 50-60 Hz

Fuse: 1 AMP SB 110 Volt

Description	Model Number	Retail Price
Sorcerer Computer 8K RAM	DP 1000-1	$ 895.00
Sorcerer Computer 16K RAM	DP 1000-2	1150.00
Sorcerer Computer 32K RAM	DP 1000-3	1395.00

Accessories

ROM PAC™ Cartridges:
(Standard Basic Included)

Media: 8-track cartridge enclosure, read-only memory on PC board.

Programs: STANDARD BASIC ASSEMBLY LANGUAGE, PILOT, APL, DOS (Fortran, Cobol)

Cassette Tape:

Media: Standard Phillips cassettes

Programs: Casino, Personal Physician, Personal Data Management, Management Aids, Computer Aided Instruction, Advance Engineering.

Sorcerer's 512 x 240 Graphics Offer High Resolution Pictures, and a Picture is Worth a Thousand Words.

Animated Graphics add a new dimension to your programming. Sorcerer's character graphics make programming pictures as easy as putting messages on the screen. A beginner using the print statement in BASIC can build pictures of his wildest imagination. The typewriter style keyboard has a graphic key and unique graphic symbols engraved on the keytops for use in the same manner as shifting for uppercase on a typewriter. By depressing the graphic and shift lock keys you may use the graphic symbols as though they were text and create any image you wish to display. Character graphics are an easy introduction to the concepts and uses of graphics in computer systems.

User Defined Graphic Symbols allow you to be creative in symbols or geometric forms, including foreign language of even new computer language symbols. Sorcerer has a selection of 256 characters of expression. 128 are fixed and predefined, while the other 128 are programmable as to their symbol representation. The graphic character is created wihtin an 8 x 8 dot matrix and any combination of the 64 dots per character space may be displayed on the screen as opposed to a bit per element graphic system. 64 graphic symbols have been pre-programmed and represented on the keyboard for simplicity. The other 64 programmable characters have been reserved for your selection and creativity. Alternatively the entire 128 characters associated with the graphic key may be user defined. Graphics bring out the artistic qualities in everyone and user-defined graphics allow for true freedom of expression.

Upper and Lower Case Alphanumerics are included in the 128 pre-defined and fixed character set. Your keyboard is truly a typewriter with this feature and letter writing or text editing becomes practical. With thirty lines of text the Sorcerer puts more information on the screen than any other personal computer. To be exact, 1,920 characters are visible at any one time before automatically scrolling to the next page. The keyboard and numeric pad provide the interaction and communication with Sorcerer. Since the keyboard style is familiar to most typists, information entry or retrieval is as easy as typing a letter.

Expansion and Growth capabilities are built into the Sorcerer™. Eventually the time will come when you feel the need to expand your system. The S-100 BUS expansion unit gives you immediate access to over 100 manufacturers of peripheral devices.

BACK TO BASIC

By Joe Felsenstein

One of the main functions of a column like this is to report to you recent techniques published in the professional computer science journals. This time I'm going to describe a new technique for searching in texts, one that should be particularly useful to anyone who is writing an editor or word processing system.

Suppose we have a text, which could be thousands of characters long, say the poem in the Preface to William Blake's "Milton," and we want to write a program which will search for the first occurrence of the phrase "nor shall the sword sleep in my hand till we have built Jerusalem," which is 65 characters long, counting blanks. The program will report back to us where in the text the phrase occurs first.

The simplest possible way to conduct the search is to first check characters 1 to 65 of the text to see if they are the phrase, then check characters 2 to 66, then 3 to 67, and so on. This is a dreadfully slow way to search the text. The average character in the text is examined a total of 65 times until we find the phrase.

There must be a better way, one that would avoid looking at each character in the text so many times. The obvious way of speeding things up is that, when we check a group of 65 characters to see if they are the phrase, we break off checking them as soon as we find one character that doesn't match. So we start by checking character 1 to see if it is N. If not (and it probably isn't) then we know characters 1 to 65 can't be the phrase we seek. So we immediately start the checking of characters 2 to 66, breaking off as soon as character 2 also proves not to be N. Occasionally we find an N in the first character we look at, and then we have is to check the next character to see if it is an O. But unless we have found the occurrence of the phrase it is unlikely to be O, so even if we have to look at more than one character, we rarely ever have to look at more than two.

This way we look at most characters only once, a few twice, and very few more than that. Now this is a much better way of doing things. We may be able to improve things a little, but this algorithm now examines each character in the text about once. Surely that's about as good as could be done, right? It must be impossible to search a text without looking at each character in it at least once, right?

Wrong. In the October, 1977, issue of the Communications of the ACM, Robert S. Boyer and J. Strother Moore, in a paper entitled "A Fast String Searching Algorithm," show that string searching need not involve looking at every character even once!

And if you don't think that's amazing, I invite you to come up with a way to do this yourself, before continuing reading this article.

The trick is based on starting from the end of the phrase and searching back towards its beginning. The last character in our phrase is M. Suppose we want to check the first 65 characters of the text. We start, not by comparing character 1 of the text to N but by comparing character 65 to M. Now the 65th character in Blake's poem is a blank, which is not an M. So we immediately know that characters 1 to 65 cannot be the phrase we seek. But now we can do better than just trying to look at characters 2 through 66. The last blank in the phrase we seek is the 56th character in the phrase. Now suppose that we have checked our phrase before starting the search, and we know for each character in the ASCII alphabet where it last occurs in the phrase, and whether it occurs at all.

Knowing that character 65 is a blank, we immediately know that we can move our attention down the text 65 - 56 = 9 characters, for the phrase could not occur any sooner than that and still have the blank match something in the phrase. So we now compare the last character of our phrase with character 65 + 9 = 4 of the text. This time we again find no M but an N. Now the last N in our phrase occurs fully 30 characters from its end, in the 35th character of the phrase. So if character 4 of the text is to be in the phrase, the phrase would have to be located 30 characters farther down, from characters 39 to 104. In 104 we find another blank, so we can again slide our attention down the text 9 more characters for the next possible match. Notice that we have now moved 48 characters down the text, and have had to examine only three characters!

And so on. An additional possibility is that when we look at the text in some position, we find a character which does not occur in the phrase at all, such as C. This is a great bonus. We know that the next place that the phrase could be located in the text starts just after the C, so that we could slide quite a ways down the text before our next search.

Boyer and Moore give at least one other technique for squeezing a bit more speed out of the method, but it is less important and I won't discuss it here. Their article is fairly clear.

If you are confused by the above verbal discussion, you might take a look at the Figure, where I give another example involving a Luddite slogan.

Here is a BASIC subprogram to search a text of length L1 located in an array A. We want to find the phrase of length L2 characters located in array B. The program uses an array C dimensioned C(128) to indicate for each of the 128 possible characters how far from the end of the phrase it occurs (with a zero if it never occurs in the phrase).

Note that we treat the arrays A and B as containing numbers, not characters. I had to do this because BASIC handles strings differently in different versions. You will probably want to convert A and B back to strings and take the numerical value of the Ith character of B and the P1th character of A in statements 1050 and 1110. The subprogram returns with P equal to the starting point of the phrase, and equal to zero if it does not occur in the text. (See Listing 1.)

This subroutine should be capable of searching in an English text by looking at only about every tenth character. So it should be ten times as fast as one which looks at every character once. If you're interested in learning more about this fast string searching method, you'll find Boyer and Moore's article fairly comprehensible.

A RANDOM MAZE

I always suspect, when looking at a maze, that there is some system used in constructing it. Now if we only had a maze constructed truly at random! Here is a program that will do that, although rather crudely. It asks you for the number of rows and the number of columns you want, and then for the probability that each wall exists. More on that probability later. The program is given in Listing 2.

You may have to change the RND(0) to whatever will get you a random fraction on your machine. The corners of the maze are given by "+," the horizontal walls by "-" and the vertical walls by "!." If you have a way of writing solid blocks on your screen, you should replace statements 120, 170, and 210 by statements which place a solid block on the screen and then move the cursor one character to the right (oh, for a standard for BASIC video graphics). This will make the maze look much more convincing.

Of course, since each wall in the maze is either present or absent with probability P at random, this maze may have parts that are rather strange. It all depends on P. If P is large, then many areas may be completely enclosed, with not only no way out, but no way in as well! If P is small, there will be many routes from one place to another. Certainly the most interesting values are intermediate. How intermediate?

The magic value is P = 0.5, on account of a mathematical theorem. It seems that there is a branch of probability theory called stochastic processes, which studies random processes and structures. One body of theory within stochastic processes is called percolation theory. Imagine water trapped in a random maze. When will it be able to "percolate" out? If P is greater than 0.5, then it can be proven mathematically (I don't know how) that in a large enough maze, any given cell will be surrounded by an unbroken boundary if you go out far enough. But if P is less than or equal to 0.5, then some cells will be surrounded bu others will be part of an infinitely large connected set of cells.

It would be nice to have a program to make a maze which looks more convincing, one that has no completely enclosed areas. In the meantime this will have to do. At least it's simple.

Listing 1

```
10 PRINT "HOW MANY ROWS IN THE MAZE? "
20 INPUT R
30 PRINT "HOW MANY COLUMNS IN MAZE? "
40 INPUT C
50 PRINT "PROBABILITY OF A WALL? "
60 INPUT P
70 FOR I = 1 TO R
80    FOR J = 1 TO C
90       IF RND(0) < P THEN 120
100      PRINT " ";
110      GO TO 130
120      PRINT "!";
130      PRINT " ";
140   NEXT J
150   PRINT
160   FOR J = 1 TO C
170      PRINT "+";
180      IF RND(0) < P THEN 210
190      PRINT " ";
200      GO TO 200
210      PRINT "-";
220   NEXT J
230   PRINT
240 NEXT I
250 END
```

Listing 2

```
1000 LET A1 = 128
1010 FOR I = 1 TO A1
1020    LET C(I) = 0
1030 NEXT I
1040 FOR I = 1 TO L2
1050    LET C(B(I)) = L2 - I
1060 NEXT I
1070 LET P = 1
1080 LET J = L2
1090 LET P1 = P + J - 1
1100 IF A(P1) = B(J) THEN 1150
1110    LET P = P + J - C(A(P1))
1120    IF P <= L1 - L2 + 1 THEN 1080
1130    LET P = 0
1140    RETURN
1150 LET J = J - 1
1160 IF J > 0 THEN 1090
1170 RETURN
```

Example of String Searching Method of Boyer and Moore

```
Phrase: Ludd
Text:   No General but Ludd means the poor any good.

(The fourth character in the text is G, doesn't occur in phrase.
 So slide down four places.  Now we have: )

Phrase:     Ludd
Text:   No General but Ludd means the poor any good.

(Again, the last character of "Ludd" lies opposite a character
which doesn't occur in the phrase, namely R. Slide down four
again:

Phrase:         Ludd
Text:   No General but Ludd means the poor any good.

(Now the second D in "Ludd" is opposite a B, so slide four more.: )

Phrase:             Ludd
Text:   No General but Ludd means the poor any good.

(Now last character of the Phrase is opposite an L.  The last
 L in the Phrase is three chracters from the end.  So slide the
 phrase down three.  Then we find a match: )

Phrase:                 Ludd
Text:   No General but Ludd means the poor any good.
```

My Column
(continued from p. 1)

which is why companies like MITS and Polymorphic have abandoned the hobbyists and gone after the businesses. So here's an opportunity to get that machine of yours to start paying for itself. Think about it, then do something about it. Let me know if you have ideas along that line, or bring them up at the next club meeting. That's what we get together for.

Well, I got such tremendous response from the last issue that I'm going to repeat the plea again this month. I'd like to hear from you, the club members, subscribers, and other readers, about what you think of what I'm saying. Most of the time I'm running off at the mouth about one or another of my pet peeves (when it comes to peeves I have a healthy menagerie), but if you have a peeve, or other pet you want to take out for a walk and air, let me know. Or if my peeves pet you the wrong way (or right way for that matter) let me know by calling me at the above numbers or throwing a postcard in the mail to the club address. I love hearing from you. I know you care about me: last month I was flooded with responses — I must have gotten two phone calls!

On to weightier matters. This month's pet peeve is languages. I mean, why not, everybody else has commented on them, and I refer to them quite often in other columns. Sometimes I think if I hear the word "BASIC" one more time I'm gonna scream. Likewise with "PASCAL" and "FORTRAN." We all seem to be so hung up on these languages, whether we like 'em, hate 'em, or just don't know, we sure like to talk about them. Most of us know the arguments. BASIC is too simplistic, non-standard, and doesn't lend itself to structured programming. On the other hand it's easy to use, simple to implement and exists in some version on many many machines. FORTRAN has almost no string handling, is very hard to read, is non-structured, and old fashioned. Yet it is highly standard, exists everywhere, and more people know it than anything other than COBOL which doesn't even deserve to be discussed here. PASCAL, the relative newcomer, is to many people an unknown, a difficult language, hard to understand and time-consuming to learn. It currently takes large amounts of memory and dual floppes making it a rich man's language for the time being. Still most people acknowledge it is very powerful, highly structured, fast and efficient. Those who know it claim it is easy to use and read.

So here we stand. There are a few other languages running around, Tiny BASIC, FOCAL, NIBLE, to mention a few, all BASIC-like, mostly simple, and mostly interpreters. Then there are a few more esoteric languages that do exist on the micros but are not readily (or cheaply) available . . . C, LISP, FORTH, TRAC, etc. Some of these may be very powerful and easy to use, but for some reason have never caught on. I can understand why LISP hasn't become popular; it must be slower than all get out on a 8080, not to mention how complex it is anywhere. But on the other hand it represents a fantastic potential to the hobbyist.

LISP, which stands for List Processing Language, is a tree structured string handling language that can't tell the difference between data and program. This means your program can generate as output other programs. Indeed that is how a LISP compiler is created: in LISP. It is fully recursive, which makes it nice for creating parsers (routines that determine language structures and such things as whether a word is a verb or adjective, etc.) It is also usually written as an interpreter. It was developed as an aid to Artificial Intelligence research, and has been used very successfully to that end. Now how does the hobbyist fit into that? As I mentioned LISP is very slow, and due to its structure it is very bulky. That means people using it have been limited in their availability to machines and time, so that even the best supported researcher has hassles getting his hands on a machine that he can run on for more than a few hours unattended. Yet along comes that micro, and suddenly machine time is negligible in cost. You, the hobbyist, can run for weeks on end, on a dedicated system! So what if our programs run one tenth as fast as the pros — we have one hundred times the resources!

Of course they have these resources now also, but we don't have to compete, but we can do true research now. The days of the basement experimenter, the homebrew Edison, are back. Armed with a micro and LISP, we can do creative, innovative, and important research in Artificial Intelligence.

And what of FORTH and TRAC? These are macro languages, things you hardly ever hear about in the hobbyist world. Yet those who learn and use these languages swear by them. (On the other hand, some computer scientists swear at them.) While there are a few macro assemblers out, for the most part they have been ignored. Macros enable the user in effect to build his/her own language. They require very little memory, run extremely fast and are usually on good terms with assembly language if you need it.

Basically they have a small program, called a Kernel, that resides in a few K of memory. Then there is a library of macros (think of them as routines), which the user can hook together however he/she pleases to form a program. The macros are invoked just by giving their name, followed by parameters. That means if you give them sensible names you can form a program that reads like a novel, and is thus self documenting. Why haven't these caught on? FORTH is expensive, not outrageously so, but available only from a few sources, FORTH Inc., and Digital Group. It also needs a little work to get up an running, I/O, etc. It isn't as trivial to learn as BASIC, etc. But mostly, I think, people are somewhat chicken of something so bizarre.

What about C? It is a very powerful, structured language — why isn't it more popular? I have no idea. I'd love to have it. Or APL? That one may be only a matter of time 'til it becomes more available. Readers of BYTE, if they're not sick of APL, are at least very aware of it and its potential.

But there are other, more far-out possibilities. So far the languages I've talked about fall roughly into three classes: Compilers, Interpreters, and Macro Languages. What about other concepts? Allow me to propose something different. Why not a computer program that is not a language at all, but a friend? Currently programming means taking your thoughts and concepts of what needs getting done and figuring out how to tell an imbecile how to do the job. This means usually doing more work than the computer itself! Why? Computers are supposed to make things easier on us. Fat chance.

This language, as I envision it, acts similarly to an Interpreter in that it runs the entire time, programs are not created with a text editor and then read by the compiler. But text is not just entered line by line then executed either. Instead the machine and the user carry on a conversation, as you would with any worker, to determine what needs to be done. Conversation might run as follows:

"I'd like to determine how much interest I'll have to pay on a new house."

"Hmmm . . . I don't understand 'interest,' 'pay,' and 'new house'; can you explain further?"

"Sure. Interest is money, an amount, I 'pay,' that is a sum to be determined suffice it to say, and a new house is an object, don't worry about what kind of object."

"OK, do you have a formula for determining interest to pay?"

"Yeh, it goes like this . . ."

And so on. I envision this as a learning program that picks up a larger vocabulary as it goes on, and needs to ask fewer questions as it gains experience.

That's my concept of a convenient computer. I'd be interested in other people's concepts, and what you would like to see in a language. How about it?

COMPUTER STORES

Almac/Stroum Electronics
5811 - 6th Ave. S
Seattle, WA 98108
763-2300

Altair Computer Center

8105 SW Nimbus Ave.
Beaverton, OR 97005
(503) 644-2314

14100 NE 20th St.
Bellevue, WA 98007
641-8800

6704 Argent
Pasco, WA
(509)547-9014

2313 Tacoma Ave. S
Tacoma, WA 98499
383-4437

Byte Shop

3482 SW Cedar Hills Blvd.
Beaverton, OR 97005
(503)644-2686

14701 NE 20th St.
Bellevue, WA 98007
746-0651

2033 SW 4th Ave.
Portland, OR 97201
(503)223-3496

2605 2nd Ave.
Seattle, WA 98121
622-7196

Computerland

14340 NE 20th St.
Bellevue, WA 98007
746-2070

1500 S. 336th St.
Federal Way, WA 98003
838-9363, Tacoma
927-8585

8791 S. Tacoma St.
Tacoma, WA 98499
581-0388

The Computer Shoppe
6401 Roosevelt Way NE
Seattle, WA 98115
525-4898

Empire Electronics
616 SW 152nd St.
Seattle, WA 98166
244-5200

Heathkit Electronic Ctr.
505 - 8th Ave. N
Seattle, WA 98109
681-2172

Micro Computer Center
11822 NE 8th St.
Bellevue, WA 98005
455-3710

Omega Computer & Stereo

839 - 106th Ave. NE
Bellevue, WA 98004
455-2126

5421 - 196th St., SW
Lynnwood, WA 98036

1032 NE 65th St.
Seattle, WA 98115

Radio Shack - many stores in the Northwest

Retail Computer Store
410 NE 72nd St.
Seattle, WA 98115
524-4101

Seawell Marketing, Inc.
315 NW 85th St.
Seattle, WA 98117
782-9480

Q. What's next for the Northwest's #1 Data Processing Distributor?

A. Data & Word Processing Supplies.

The new C&L Terminals Supplies Division lets you do all your data and word processing buying with one phone call—quality supplies, competitive prices and the most complete selection in town: Paper, printwheels (Qume & Diablo), computer ribbons, word processing ribbons, floppy diskettes, cassettes, computer tape, terminal acoustical covers, terminal stands, as well as CRT's, portable terminals, printers, and repair service.

Give us a call and find out how easy it is to buy your data supplies from C&L Terminals. Let us be your source for computer terminals, repair service and, now, data and word processing supplies.

PRINT

C&L Terminals

DATA SUPPLIES DIVISION
13500 Bellevue-Redmond Rd. Bellevue 98005
641-6820

SELECTRIC

By Michael Holley

Recently many Selectric I/O terminals have appeared on the used equipment market. The reason behind this is speed; the new formed character terminals (Diablo, Qume and the like) can print at 45 to 60 characters per second (cps), while the Selectric prints at less than 15 cps. Thus 2 to 3 year old terminals are available at a fraction of their $3,000 to $4,000 original cost. Over 20 members of the Northwest Computer Society have acquired factory reconditioned Trendata 1000 Selectric terminals plus a few others have similar Selectric I/O machines.

The Trendata 1000 is an IBM compatible 2741 communication terminal with a RS-232 serial interface. It is electrically compatible with RS-232 modems, computers and terminals but the character code is IBM correspondence, not ASCII. This means the computer I/O software must convert ASCII to correspondence and vice versa. Many timeshare services offer this software and I use my terminal on Nordata's DEC RSTS/E system that has this I/O software. When shopping for a Selectric I/O, be sure to find out if it is a complete terminal or just the typewriter. A terminal can be used with a micro computer with custom software while a I/O typewriter alone will require a fair amount of custom electronics also. The 2741 type terminal comes in various codes; correspondence is the one I would recommend because it uses standard print elements (golf balls). The other codes (PTTC, BCD and EBCD) all use special print elements.

SYSTEM REQUIREMENTS

To interface a 2741 terminal to a micro computer all that is needed is a serial I/O and some software. The interface must handle a 6 bit word with odd parity and 1 stop bit at 134.5 baud. Also the interface needs to be able to send and detect breaks, a 200 ms constant space. Due to the half duplex nature of the 2741 terminal, it is easier to use it as a printer only, than as a full input-output device. The input software isn't that difficult — it's patching it into the calling program that is hard. Most software packages assume a full duplex terminal and so some modification is necessary. I have one output routine that works with all the packages that I have tried. Input routines often poll the keyboard for a control C or what not and sometimes the program just flat requires full duplex, so patching input routines is harder. One other drawback to using the 2741 terminal for input is the lack of control characters. Again this is only a software problem and with clever I/O drivers it can be overcome. I have a full input and output routine patched to BASIC that works just fine.

2741 OPERATION

A good source of information is the IBM 2741 Communication Terminal manual available from an IBM reference library for about 60 cents (File No. TP-09 Order No. GA24-3415-3).

The 2741 is a half duplex terminal that uses a Start Transmission (STX) code and an End Transmission (ETX) to change between the send and receive state or mode. This is different from the common ASCII terminal most small computers work with. The 2741 has three states or modes: Transmit, the keyboard is unlocked and the terminal can send text; Receive, the keyboard is locked and the terminal can receive text; Control, a state between transmit and receive. In normal operation the cycle is transmit, control, receive, then back to transmit.

When the 2741 is switched on and placed in the communicate mode it is in the transmit state. The operator may now send text to the computer. The transmit state may be ended by pressing the return key or the attention key. The normal method is the return key. This sends a return code followed by a ETX code and the terminal goes into the control state with the keyboard locked. The attention key just sends the ETX code without the return code.

After receiving an ETX from the terminal, the computer may send a STX and the terminal will be placed in the receive state. The computer may now send as many lines of text as needed. The carrier return doesn't end the transmission. To end the transmission the computer sends an ETX code that places the terminal in the transmit state with the keyboard unlocked. When the terminal goes into the transmit state it automatically sends a STX code. Now the operator may send another line of text. The computer doesn't have to send a STX code after receiving a return and ETX codes; it may just send an ETX to allow the operator to enter many lines of text.

INTERRUPTS

In normal operation the text is preceeded by a STX and followed by an ETX code, however, this operation may be interrupted by sending a 200 ms break. If the operator is sending text to the computer and the input buffer overflows, the computer must send an error message. To do this the computer sends a 200 ms constant space causing the terminal to go into the control mode. The computer may now send a STX code and then the error message.

If the computer is sending text and the operator wishes to stop it (as a control C in BASIC), the attention key is pressed. This will cause the terminal to send a 200 ms space to the computer. If the computer interface can recognize a break and the program honors this request the transmission will be ended. What happens is entirely up to the program, but normally the computer would send an ETX to place the terminal in the transmit state. The terminal would then send a STX and unlock the keyboard. The attention key may also be used to replace some of the control key functions when the terminal is in the transmit state. Pressing the attention key may be used to delete the line of text just entered.

CODES

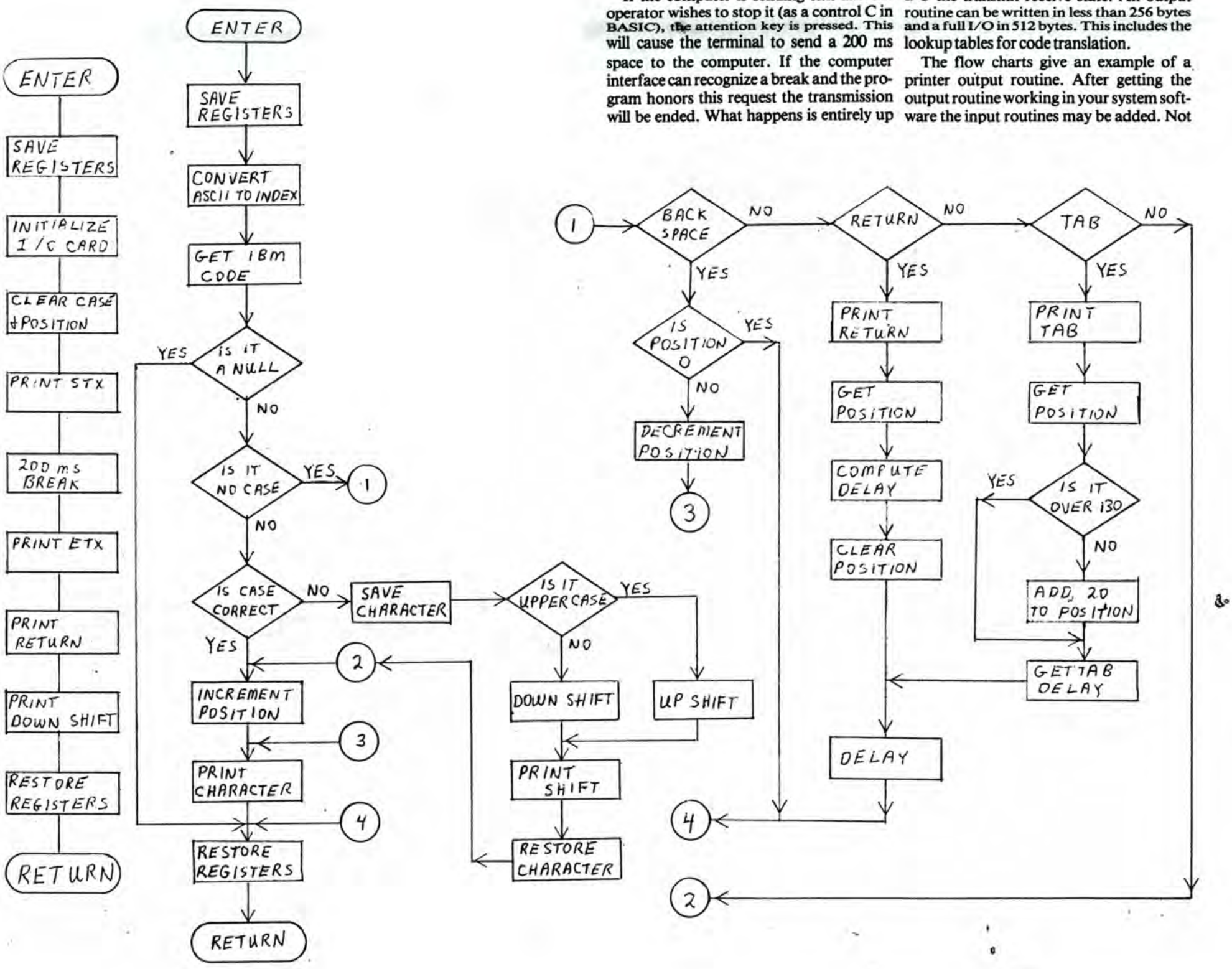
The 6 bit code for ETX is 3C hex and for STX it is 34 hex. The code for STX is the same as the code for 9 or (. Thus, if the computer sends a STX when the terminal is already in receive, the STX will print as a 9 or (. The code for ETX is non printing. The IBM 2741 manual states that the terminal should always be in the transmit state when the computer sends a break. To insure this just send a ETX before sending the 200 ms space.

It takes 7 bits to represent all 128 ASCII codes but the Selectric terminal has only 6 bits to represent its 105 or so codes. This requires some codes to represent two characters, normally the upper and lower case. The code for both upper and lower case "A" is 39 hex unlike the ASCII 41 hex and 61 hex. In the translation program the computer must keep track of the shift position of the terminal. To type the word "High" the computer must send shift up, H, shift down, i, g, h or in hex 1C, 26, 1F, 19, 23, 26.

The other consideration of the I/O program is the delay for the carrier return and tab. Just as a Teletype requires nulls the Selectric requires idle codes, 3D hex, or a delay loop. The formula for the number of idle codes is N = inches of travel + 1.5. At 10 characters per inch a 80 character line will require 10 idle codes or 670 ms delay. At 134.5 baud a character is 67 ms long.

A software conversion routine needs to keep track of the terminal shift position, the carrier position (character count) and if full I/O the transmit receive state. An output routine can be written in less than 256 bytes and a full I/O in 512 bytes. This includes the lookup tables for code translation.

The flow charts give an example of a printer output routine. After getting the output routine working in your system software the input routines may be added. Not



all programs will need input from the Selectric keyboard nor will it be possible to add it to all programs. BASIC was the only program that I patched the input routines to.

INITIALIZATION

The first portion of the program is the interface and terminal initialization. This initializes the I/O card if needed, then ensures that the terminal is in the receive state with the carrier returned to position zero. For output only, two permanent memory locations are needed, the upper lower case flag and the print head position counter. To ensure that the terminal is in receive, it is first sent an ETX, a non printing code, then a 200 ms break, and finally a STX to place it in receive. This looks like overkill but it will work every time without any "9"s being printed when a STX is sent to a terminal already in receive.

OUTPUT PROGRAM

The main program first converts the code from ASCII to correspondence. This is done with a lookup table of all the IBM codes in ASCII order. Thus an upper case "A" is the 65th (41 hex) code in the table. Only six bits are needed for correspondence code (A = 39 hex), so two bits may be used for flags. In my program bit seven is a "no case" flag. That is, the character prints the same in upper or lower case (i.e. space and on some type balls period and comma). This speeds up the printing by eliminating unnecessary shifts. For the most part, the "nocase" characters are less than 20 hex in ASCII, so a flag bit need not be used. Just test the ASCII code to see if it is nocase.

In my program bit six is used to identify whether the character is an upper or lower case. If the shift portion of the Selectric isn't correct the proper shift character must be sent and the shift flag updated. After translation the correct code is sent and the print head position is incremented.

SPECIAL CHARACTERS

The only special characters are backspace, tab, and carrier return. The other nocase characters are just printed. A back space must decrement the counter but not underflow it. Tab adds a constant to the counter (say 20 or so) and must set up a delay for carrier movement. The carrier return sends a return, gets the old position count, clears the counter, then computes the proper delay. A 130 character line would require a delay of around 1 second or 15 idle codes.

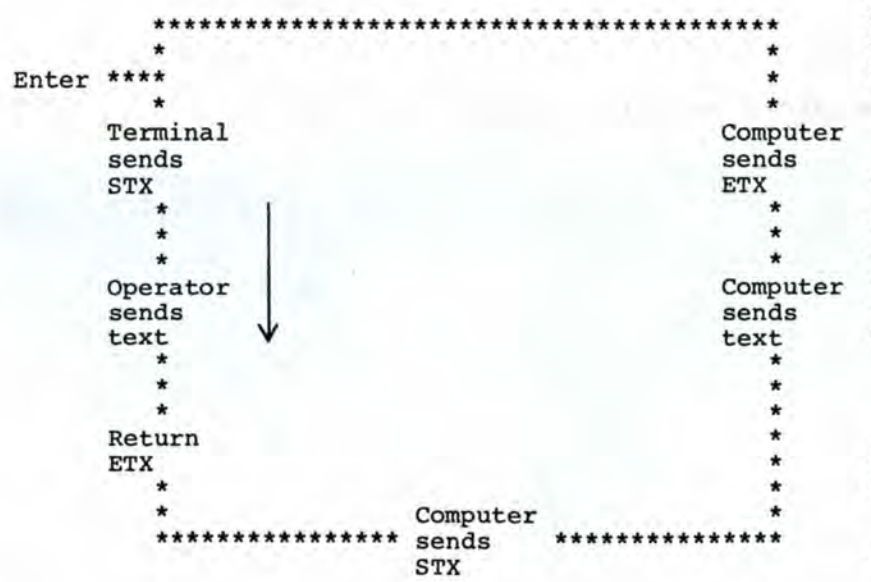
A 6800 PROGRAM

The following program will allow use of an IBM 2741 Selectric terminal with a 6800 based computer. This program is for output only. This program was tested on a TRENDATA 1000 terminal with a Southwest Technical Products Corporation 6800 computer but should work with any 2741 type terminal.

SYSTEM REQUIREMENTS

The program is assembled to reside in ROM or RAM at C000 to C37F hex with 9 bytes of RAM at A020 hex. The program may be reassembled anywhere but the look-up table must start at a page boundary (xx00 hex). Only 2 bytes of permanent RAM are needed; the other 7 may be scratch pad. A MP-C control interface is used because the MP-S serial interface will

2741 Selectric Terminal Operation Flow Diagram



State	Transmit	Control	Receive
Keyboard	Unlocked	Locked	Locked

Line Code Chart (Correspondence)

	0.	1	2	3
0	Space	' 2 @	1 (1)	3 #
1	(2)	. (3)	m M	v V
2	t T	n N	x X	u U
3	j J	= +	g G	f F
4	4 $	z Z	0)	9 (
5	o O		s S	w W
6	l L		h H	b B
7	/ ?		y Y	- _
8	5 %	6 (4)	7 &	8 *
9	' "	i I	r R	a A
A	e E	k K	d D	c C
B	p P	q Q	; :	, (5)
C	Punch On	Upper Case	Reader Off	End Xmit
D	Restore	Backspace	Return	Null
E	Bypass	End of Blk	Line Feed	Prefix
F	Punch Off	Lower Case	Horiz Tab	Delete

Courier Ball	ASCII Ball
(1) Plus or minus	Exclamation mark
(2) 1/2 1/4	Brackets
(3) Period	Greater Then
(4) Cents	Circumflex
(5) Comma	Less Then


```
                NAM    PRINT SOFT I/O
* Copyright (C) 1978
* Michael Holley
* 16202 121 Ave NE
* Bothell Wa 98011
*
* All Rights Reserved
*
* This program may be copied for personal use only.
* No commercial duplication is authorized.
*
* ASCII to Correspondence Code for a SWTP 6800
* Allows use of a Trendata 1000 or any 2741 type
* terminal with a 6800 type computer.
*
*        OUTPUT ONLY
*
* This program requires a MP-C control
* interface at port 0.
*
* The 134.5 baud clock is available from
* pin 14 on the MC 14411 chip loacted on
* the CPU board. It may be routed to the
* MP-C via UD2 and UD4.
*
* This program will run in ROM, if RAM is needed
* at C000 hex a 4K board may be modified.
* Break the foil between IC22 pin 6 and IC24 pin 1,
* IC22 pin 4 and connector pin A15.
* Jumper IC22 pin 6 to connector pin A15.
* Jumper IC22 pin 4 to IC22 pin 2.
* Set the board jumper to #4.
*
* Pins 5, 6, and 8 of the DB-25 RS-232
* connector should be tied to a +12 volt
* source. Reader control and pin 20 are
* +12 volt sources.
*
*****************************************
*For TSC FLEX  Disk Operating System
*Links PRINT.SYS to P.CMD
*****************************************
0010                    ORG     $0010
0010 C0 03              FDB     INIZ
0012 C0 00              FDB     OUTPUT
710D                    ORG     $710D
710D C0 00              FDB     OUTPUT
*****************************************
*   OUTPUT PORT
8000                    ORG     $8000
*****************************************
8000            PIADA   RMB     1
8001            PIASA   RMB     1
8002            PIADB   RMB     1
8003            PIASB   RMB     1
*
*****************************************
*   TEMPORARY RAM
A020                    ORG     $A020
*****************************************
A020            XTEMP   RMB     2
A022            ATEMP   RMB     1
A023            BTEMP   RMB     1
A024            TABLEX  RMB     2
A026            IOTEMP  RMB     1
*
*****************************************
*   PERMANENT RAM
*****************************************
A027            CASE    RMB     1
A028            POSIT   RMB     1
*
*****************************************
C000                    ORG     $C000
*THIS CODE WILL RUN IN ROM
*****************************************
* ENTERING VECTORS
*
C000 7E C0 25   OUTPUT  JMP     OUT
C003 7E C0 06   INIZ    JMP     INIT
*****************************************
*INITIALIZATION PROGRAM
*
C006 8D 5A      INIT    BSR     SAVE        SAVE A,B,& INDEX
C008 8D 65              BSR     INITP1      INITIALIZE PORT
*INIT TERMINAL
*THIS MAKES SURE TERM IS IN SEND MODE THEN GIVES
*REVERSE BREAK TO PREVENT A 9 OR ) FROM BEING
*PRINTED WHEN 34 HEX IS SENT.
C00A 7F A0 28   INIZT   CLR     POSIT       ZERO CHAR COUNT
C00D 7F A0 27           CLR     CASE        LOWER CASE
C010 86 3C              LDA A   #$3C        SEND MODE
C012 8D 58              BSR     OUTCH1
C014 8D 5C              BSR     BREAK1
C016 86 34              LDA A   #$34        RECEIVE MODE
C018 8D 52              BSR     OUTCH1
C01A 86 2D              LDA A   #$2D        CR
C01C 8D 4E              BSR     OUTCH1
C01E 86 1F              LDA A   #$1F        DOWN SHIFT
C020 8D 4A              BSR     OUTCH1      PRINT
C022 8D 34              BSR     DONE        RESTORE A,B,& INDEX
C024 39                 RTS
*
*****************************************
*CONVERSION AND OUTPUT
*
*A TABLE IN ASCII ORDER IS NEEDED
*BIT 7 IS NOCASE FLAG
*IF SET THE CHAR PRINTS THE SAME IN UPPER
*OR LOWER CASE
*
*BIT 6 IS CASE FLAG
*IF SET THE CHAR IS UPPER CASE
*
```

```
C025 8D 3B      OUT     BSR     SAVE
C027 84 7F              AND A   #$7F       7 BIT ASCII CODE
C029 B7 A0 25           STA A   TABLEX+1   LOWER BYTE OF TABLE
C02C 86 C3              LDA A   #TABLE
C02E B7 A0 24           STA A   TABLEX     UPPER BYTE OF TABLE
C031 FE A0 24           LDX     TABLEX
C034 A6 00              LDA A   0,X        TRANSLATE
C036 27 20              BEQ     DONE       IF NULL CHAR SKIP IT
C038 2B 3B              BMI     NOCASE     IF BIT 7 IS SET NOCASE
C03A 16                 TAB                COPY CODE TO B
C03B C4 40              AND B   #$40       MASK ALL BUT CASE
C03D F1 A0 27           CMP B   CASE       IS IT SAME CASE
C040 27 11              BEQ     OUTONE     IF IT IS PRINT CHAR
C042 C1 40              CMP B   #$40       SHOULD IT BE UP CASE
C044 27 04              BEQ     UP         SHIFT TO UC
C046 86 1F      DOWN    LDA A   #$1F       DOWN SHIFT CHAR
C048 20 02              BRA     SHIFT      PRINT SHIFT AND CHAR
C04A 86 1C      UP      LDA A   #$1C       UP SHIFT CHAR
C04C F7 A0 27   SHIFT   STA B   CASE       UPDATE CASE
C04F 8D 1B              BSR     OUTCH1     PRINT SHIFT
C051 A6 00              LDA A   0,X        GET CHAR BACK
C053 7C A0 28   OUTONE  INC     POSIT      ONE MORE CHAR
C056 8D 14      PRINT   BSR     OUTCH1     PRINT CHAR
C058 B6 A0 22   DONE    LDA A   ATEMP      RESTORE ACC A
C05B F6 A0 23           LDA B   BTEMP      RESTORE B ACC
C05E FE A0 20           LDX     XTEMP      RESTORE INDEX
C061 39                 RTS
                *
C062 B7 A0 22   SAVE    STA A   ATEMP      SAVE ACC A
C065 F7 A0 23           STA B   BTEMP      SAVE ACC B
C068 FF A0 20           STX     XTEMP      SAVE INDEX
C06B 39                 RTS
                *
                ****************************************
                *I/O VECTORS
                *
                *ONLY OUTCHR MUST SAVE INDEX
                *ALL MAY USE ACC A & B
C06C 7E C0 D5   OUTCH1  JMP     OUTCHR     OUTPUT CHAR FROM ACC A
C06F 7E C0 B7   INITP1  JMP     INITP      INITIALIZE OUTPUT PORT
C072 7E C1 0B   BREAK1  JMP     BREAK      SEND A 200 mSEC BREAK
                *
                ****************************************
                *NOCASE CHAR PRINT THE SAME IN UPPER OR LOWER
                *RETURN TAB SPACE BACKSPACE
                *
C075 81 AD      NOCASE  CMP A   #$AD        IS IT CR
C077 27 14              BEQ     CR
C079 81 AF              CMP A   #$AF        IS IT TAB
C07B 27 1C              BEQ     TAB
C07D 81 9D              CMP A   #$9D        IS IT BACKSPACE
C07F 27 02              BEQ     BACKSP
C081 20 D0              BRA     OUTONE      ALL OTHERS JUST PRINT
                *
                *SPECIAL CHAR ROUTINES
                *
C083 7D A0 28   BACKSP  TST     POSIT       CHECK CHAR COUNT
C086 27 D0              BEQ     DONE        IF 0 DON'T BACKSPACE
C088 7A A0 28           DEC     POSIT       UPDATE CHAR COUNT
C08B 20 C9              BRA     PRINT
                ****************************************
C08D 8D DD      CR      BSR     OUTCH1      PRINT CR
C08F F6 A0 28           LDA B   POSIT       WHAT WAS CHAR COUNT
C092 7F A0 28           CLR     POSIT       MAKE CHAR COUNT 0
C095 CB 0A              ADD B   #10         MINIMUM DELAY
C097 20 10              BRA     NULOUT
                ****************************************
C099 8D D1      TAB     BSR     OUTCH1      PRINT TAB
C09B C6 14              LDA B   #20         DELAY FOR TAB
C09D B6 A0 28           LDA A   POSIT       GET POSITION
C0A0 81 82              CMP A   #130        MAX POSIT
C0A2 2E 05              BGT     NULOUT
C0A4 8B 14              ADD A   #20         TAB IS 20 SPACES
C0A6 B7 A0 28           STA A   POSIT       UPDATE POSITION
                ****************************************
C0A9 8D 05      NULOUT  BSR     DELAY
C0AB 5A                 DEC B               COUNT IT
C0AC 26 FB              BNE     NULOUT      DO IT SOME MORE
C0AE 20 A8              BRA     DONE
                ****************************************
C0B0 CE 03 E8   DELAY   LDX     #1000       CR DELAY CONSTANT
C0B3 09         LOOP    DEX
C0B4 26 FD              BNE     LOOP
C0B6 39                 RTS
                *
                ****************************************
                *
                *I/O ROUTINES FOR CONTROL INTERFACE
                *
                ****************************************
                *INIT MPC
                *
C0B7 7F 80 01   INITP   CLR     PIASA       SET UP DATA DIR
C0BA 7F 80 03           CLR     PIASB
C0BD 86 05              LDA A   #$05        BITS 0 & 2 OUTPUTS
C0BF B7 80 00           STA A   PIADA
C0C2 B7 80 02           STA A   PIADB
C0C5 86 3C              LDA A   #$3C
C0C7 B7 80 01           STA A   PIASA
C0CA B7 80 03           STA A   PIASB       ECHO OFF/READER ON
C0CD B7 80 02           STA A   PIADB       SET TIMER
C0D0 4C                 INC A
C0D1 B7 80 00           STA A   PIADA       MARK OUTPUT
C0D4 39                 RTS
                *
                ****************************************
                *OUTPUT ONE CHAR TO MP-C
                *
C0D5 84 3F      OUTCHR  AND A   #$3F        GET 6 BIT CODE
C0D7 B7 A0 26           STA A   IOTEMP
C0DA 16                 TAB                 COPY INTO ACC B
C0DB 54                 LSR B
C0DC 1B         MORE    ABA                 ADD A & B
C0DD 54                 LSR B               MOVE OVER TO NEXT BIT
C0DE 26 FC              BNE     MORE        QUIT IF NO MORE
C0E0 4C                 INC A               ODD PARITY
C0E1 84 01              AND A   #$01        ONLY PARITY BIT
C0E3 27 02              BEQ     BACK
C0E5 86 40              LDA A   #$40
C0E7 F6 A0 26   BACK    LDA B   IOTEMP      GET CODE BACK
C0EA 1B                 ABA                 ADD PARITY & DATA
C0EB 8A 80              ORA A   #$80        STOP BIT
C0ED C6 09              LDA B   #$09        START+6BITS+PAR+STOP
C0EF 7F 80 00           CLR     PIADA       SET UP START BIT
C0F2 8D 10              BSR     DE          START TIMER
C0F4 8D 09      NEXT    BSR     DEL         DELAY ONE BIT TIME
C0F6 B7 80 00           STA A   PIADA       OUTPUT ONE DATA BIT
C0F9 0D                 SEC
C0FA 46                 ROR A               SHIFT IN NEXT BIT
C0FB 5A                 DEC B
C0FC 26 F6              BNE     NEXT
C0FE 39                 RTS
C0FF 7D 80 02   DEL     TST     PIADB       IS TIME UP
C102 2A FB              BPL     DEL
C104 7C 80 02   DE      INC     PIADB       RESET TIMER
C107 7A 80 02           DEC     PIADB
C10A 39                 RTS
                ****************************************
                *200 mSEC BREAK
                *
C10B 86 00      BREAK   LDA A   #$00        BIT 7 LOW/SPACE
C10D B7 80 00           STA A   PIADA       START BREAK
C110 CE 61 A8           LDX     #25000      200 mSEC CONSTANT
C113 09         LOOP8   DEX                 8 CYCLE LOOP
C114 26 FD              BNE     LOOP8
C116 86 FF              LDA A   #$FF        BIT 7 HIGH/MARK
C118 B7 80 00           STA A   PIADA       END BREAK
C11B 39                 RTS
                *
                ****************************************
                *Conversion table for ASCII to IBM
                *correspondence code (ASCII ball IBM# 1167168)
                *
00C3            TABLE   EQU     $C3
C300                    ORG     $C300
                *               NUL,SOH,STX,ETX,EOT,END,ACK,BEL
C300 00                 FCB     $00,$00,$B4,$BC,$00,$00,$00,$00
                *
                *               BS ,HT ,LF ,VT ,FF ,CR ,SO ,SI
C308 9D                 FCB     $9D,$AF,$00,$00,$00,$AD,$00,$00
                *
                *               DLE,DC1,DC2,DC3,DC4,NAK,SYN,ETB
C310 00                 FCB     $00,$00,$00,$00,$00,$00,$00,$00
                *
                *               CAN,EM ,SUB,ESC,FS ,GS ,RS ,US
C318 00                 FCB     $00,$00,$00,$00,$00,$00,$00,$00
                *
                *               SP ,½  ,"  ,#  ,$  ,%  ,&  ,'
C320 80                 FCB     $80,$60,$49,$70,$44,$48,$68,$09
                *
                *               (  ,)  ,*  ,+  ,,  ,-  ,.  ,/
C328 74                 FCB     $74,$64,$78,$53,$3B,$37,$11,$07
                *
                *               0  ,1  ,2  ,3  ,4  ,5  ,6  ,7
C330 24                 FCB     $24,$20,$10,$30,$04,$08,$18,$28
                *
                *               8  ,9  ,:  ,;  ,±  ,=  ,¼  ,?
C338 38                 FCB     $38,$34,$6B,$2B,$7B,$13,$51,$47
                *
                *               @  ,A  ,B  ,C  ,D  ,E  ,F  ,G
C340 50                 FCB     $50,$79,$76,$7A,$6A,$4A,$73,$63
                *
                *               H  ,I  ,J  ,K  ,L  ,M  ,N  ,O
C348 66                 FCB     $66,$59,$43,$5A,$46,$61,$52,$45
                *
                *               P  ,Q  ,R  ,S  ,T  ,U  ,V  ,W
C350 4B                 FCB     $4B,$5B,$69,$65,$42,$72,$71,$75
                *
                *               X  ,Y  ,Z  ,(  ,?  ,)  ,¢  ,_
C358 62                 FCB     $62,$67,$54,$01,$47,$41,$58,$77
                *
                *               '  ,a  ,b  ,c  ,d  ,e  ,f  ,g
C360 09                 FCB     $09,$39,$36,$3A,$2A,$0A,$33,$23
                *
                *               h, i  ,j  ,k  ,l  ,m  ,n  ,o
C368 26                 FCB     $26,$19,$03,$1A,$06,$21,$12,$05
                *
                *               p  ,q  ,r  ,s  ,t  ,u  ,v  ,w
C370 0B                 FCB     $0B,$1B,$29,$25,$02,$32,$31,$35
                *
                *               x  ,y  ,z  ,(  ,½  ,)  ,-  ,DEL
C378 22                 FCB     $22,$27,$14,$01,$60,$41,$37,$00
                        END
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

ATEMP	A022	BACK	C0E7	BACKSP	C083	BREAK	C10B	BREAK1	C072
BTEMP	A023	CASE	A027	CR	C08D	DE	C104	DEL	C0FF
DELAY	C0B0	DONE	C058	DOWN	C046	INIT	C006	INITP	C0B7
INITP1	C06F	INIZ	C003	INIZT	C00A	IOTEMP	A026	LOOP	C0B3
LOOP8	C113	MORE	C0DC	NEXT	C0F4	NOCASE	C075	NULOUT	C0A9
OUT	C025	OUTCH1	C06C	OUTCHR	C0D5	OUTONE	C053	OUTPUT	C000
PIADA	8000	PIADB	8002	PIASA	8001	PIASB	8003	POSIT	A028
PRINT	C056	SAVE	C062	SHIFT	C04C	TAB	C099	TABLE	00C3
TABLEX	A024	UP	C04A	XTEMP	A020				

8080 DIS-ASSEMBLER

by John Aurelius

Did you ever buy a machine - language program for your microcomputer and then find that you need to modify it - but the vendor is being coy and not providing a listing? I have, and am not amused by it.

So I've written an 8080 dis - assembler for use on the Nordata timesharing system. It is in BASIC-PLUS and it's still fairly crude, but I don't think it has too many bugs. To use it, use an editor to create a file containing an ASCII hex dump of the code you wish to dis - assemble. By an "ASCII hex dump" I mean that each byte is represented by two ASCII characters, such as 31 or FF or B4, etc. You can use upper or lower case letters for the values A - F, and separate each byte from the next with a space or comma (the program just throws every third character away). Put as many bytes in a line as you like - I prefer 16 bytes per line, as that's how my Processor Tech monitor feeds me my dumps.

Now run the program. It will ask for the name of the input file and the name of the output file (on the Nordata system, you can specify KB: as the output file if you want it on the terminal). Then it asks for the starting address as a hex value. Like the Scelbi Assembler I critiqued last month, the program first goes over the file and builds a "label table". Whenever an opcode is found that has two data bytes following it, these bytes are assumed to be an address to be placed in the label table. The table is in numerical order of the address values. Also placed in the table are the labels ORG and END, with the starting and ending addresses. The program assigns arbitrary label names to the other values in three series: LOW, ADD and HGH, depending on if the value is below that of ORG, between ORG and END, or higher than END. Labels are numbered, as LOW1, LOW2, LOW3, etc.

Then the program prints out a conventional assembly listing. The first item on each line is the hex address of the first byte, then the hex value(s) of the opcode and any following byte(s), then a label if one applies to the address, the opcode mnemonic, and finally a value or label for the following byte(s) if present. If there is a single following byte the decimal value is printed along with an ASCII character if a printable one has that value. If there is a double byte the label is printed. Hex values will be printed for all double bytes, both as labels in the listing and as EQU pseudocodes.

A note of caution: this program is not smart enough to handle data blocks embedded in the input code, and it will try to dis - assemble them! Your only hope is that these bytes will hit a value that is not a valid 8080 opcode. Then you'll get an ERROR flag. At any rate, you may have to re-run the program as you go through the output trying to understand the dis - assembled code.

The program would be better if it ran in the micro and accessed the actual bytes of the program, rather than working from an ASCII hex dump transferred (or worse, typed) onto Nordata.

If you don't want to use the whole program, note that there are some little subroutines to convert hex to decimal and vice - versa. Feel free to lift them. Someone may want to replace the 8080 table with a 6800 one or... Go to it. If you are on the Nordata system, you can use the program without typing it in; it's (7,44)DISASS.

A note about some peculiarities in the listing. Nordata can be used with Selectric - type terminals like mine, but its character set does not match my "ASCII-type" typing element, so I can't print the 'greater than' and 'less than' symbols. The program has too many of them to fix manually. So, when you see ± it means 'less than', and when you see ¼ it means 'greater than'. Sorry about that. The others are peculiar to BASIC-PLUS. Integer variables have % added to the variable name, as in J% or R3%. Floating point numbers sometimes act oddly when compared for equality; thus this BASIC has the == operator, which means 'approximately equal' or, equal within 8 place accuracy.

```
100                     TEXIN   EQU     100H

170     31 FF CB        ORG     LXI SP  STACK

173     21 00 02        ADD1    LXI H   TEXT

176     06 0D           ADD2    MVI B   13 'CR'
178     CD B4 CA                CALL    CHROUT
17B     DB 00                   IN      0
17D     2F                      CMA
17E     E6 01                   ANI     1
180     C2 91 01                JNZ     ADD4

183     CD 00 01        ADD3    CALL    TEXIN
186     DB 00                   IN      0
188     2F                      CMA
189     E6 01                   ANI     1
18B     CA 83 01                JZ      ADD3
18E     CD 8C CB                CALL    DELAY

191     DB 03           ADD4    IN      3
193     E6 7F                   ANI     127
195     CA 04 C0                JZ      RETRN
198     47                      MOV B,A
199     CD B4 CA                CALL    CHROUT
19C     78                      MOV A,B
19D     FE 4C                   CPI     76 'L'
19F     CA CC 01                JZ      ADD6
1A2     FE 02                   CPI     2
1A4     CA B6 01                JZ      ADD5
1A7     FE 04                   CPI     4
1A9     C2 76 01                JNZ     ADD2
1AC     06 0D                   MVI B   13 'CR'
1AE     CD B4 CA                CALL    CHROUT
1B1     AF                      XRA A
1B2     77                      MOV M,A
1B3     21 00 02                LXI H   TEXT

1B6     DB 00           ADD5    IN      0
1B8     2F                      CMA
1B9     E6 01                   ANI     1
1BB     C2 91 01                JNZ     ADD4
1BE     7E                      MOV A,M
1BF     FE 00                   CPI     0
1C1     CA 73 01                JZ      ADD1
1C4     47                      MOV B,A
1C5     CD B4 CA                CALL    CHROUT
1C8     23                      INX H
1C9     C3 B6 01                JMP     ADD5

1CC     06 2E           ADD6    MVI B   46 '.'
1CE     CD B4 CA                CALL    CHROUT
1D1     CD 27 C2                CALL    GCLIO
1D4     11 64 CA                LXI D   INPTR
1D7     CD 7E C3                CALL    SHEX
1DA     C3 76 01                JMP     ADD2

1DD                     END     EQU     1DDH
200                     TEXT    EQU     200H
C004                    RETRN   EQU     C004H
C227                    GCLIO   EQU     C227H
C37E                    SHEX    EQU     C37EH
CA64                    INPTR   EQU     CA64H
CAB4                    CHROUT  EQU     CAB4H
CB8C                    DELAY   EQU     CB8CH
CBFF                    STACK   EQU     CBFFH
```

```
RUN
DISASS    11:43 PM  12-Sep-78

INPUT FROM? EDIT0
OUTPUT  TO? EDIT
STARTING ADDRESS (HEX)? 170


LABEL TABLE READY. DO YOU WANT IT PRINTED? y

NUMBER        LABEL         HEX ADDR
 1            LOW1          100
 2            ORG           170
 3            ADD1          173
 4            ADD2          176
 5            ADD3          183
 6            ADD4          191
 7            ADD5          1B6
 8            ADD6          1CC
 9            END           1DD
 10           HGH1          200
 11           HGH2          C004
 12           HGH3          C227
 13           HGH4          C37E
 14           HGH5          CA64
 15           HGH6          CAB4
 16           HGH7          CB8C
 17           HGH8          CBFF

TO ENTER A NEW LABEL NAME, ENTER LABEL NO.,NAME
ENTER -1, (THE COMMA IS NECESSARY); TO PRINT LABEL TABLE
ENTER 0, TO OUTPUT ASSEMBLER LISTING
? 1,texin
? 10,text
? 11,retrn
? 12,gclio
? 13,shex
? 14,inptr
? 15,chrout
? 16,delay
? 17,stack
? 0,
```

```
pip edit0

31,FF,CB,21,00,02,06,0D,CD,B4,CA,DB,00,2F,E6,01
c2,91,01,cd,00,01,db,00,2f,e6,01,ca,83,01,cd,8c
cb,db,03,e6,7f,ca,04,c0,47,cd,b4,ca,78,fe,4c,ca
cc,01,fe,02,ca,b6,01,fe,04,c2,76,01,06,0d,cd,b4
ca,af,77,21,00,02,db,00,2f,e6,01,c2,91,01,7e,fe
00,ca,73,01,47,cd,b4,ca,23,c3,b6,01,06,2e,cd,b4
ca,cd,27,c2,11,64,ca,cd,7e,c3,c3,76,01
¢C

Ready
```

```
DISASS    11:26 PM  12-Sep-78

100 REM  **This program is an 8080 dis-assembler**
110 REM  23-Aug-78
115 REM By John P. Aurelius
120 REM ***************************
130 REM  --INITIALIZE--
140 REM
150 DIM A$(256),S(128),S$(128)
160 MAT READ A$
170 FOR J%=65 TO 128
180   IF J% ±¼ 119 THEN A$(J%)="1MOV "+A$(J%)
190 NEXT J%
200 ON ERROR GOTO 9800
205 PRINT
210 Q%=1 : PRINT"INPUT FROM"; : GOSUB 8800
220 IF Q% ± 0 THEN PRINT"NO FILE "N$(1)" EXISTS" : GOTO 210
230 Q%=2 : PRINT"OUTPUT  TO"; : GOSUB 8800
240 IF Q% ± 0 THEN 280 ELSE IF N$(2)="KB:" THEN 285
250 INPUT"FILE "N$(2)" EXISTS. OK TO OVERWRITE";A$
260 A$=CVT$$(A$,32) : IF LEFT(A$,1) ±¼ "Y" THEN 230
270 CLOSE 2
280 OPEN N$(2) FOR OUTPUT AS FILE 2
285 INPUT"STARTING ADDRESS (HEX)";L$ : L$=CVT$$(L$,32)
290 H$=L$ : GOSUB 8000
```

```
295 IF H ± 0 THEN PRINT "INVALID" : GOTO 285
300 L,L9=H : PRINT
990 REM
1000 REM  **FIRST PASS**
1010 REM
1020 S(1)=L9 : P1%=1
1030 GOSUB 6800  ½ Get an opcode & operands
1040 IF H$="" THEN B2=L : C%=3
1050 IF C%=0 THEN PRINT"**ERROR** "L$" "C1$
1060 IF C% ±¼ 3 THEN 1170
1070 P2%=P1%+1
1080 FOR J%=1 TO P1%
1090   IF B2 ± S(J%) THEN P2%=J% : J%=P1%
1100   IF B2 == S(J%) THEN P2%=0 : J%=P1%
1110 NEXT J%
1120 IF P2%=0 THEN 1160
1130 S(J%+1)=S(J%) FOR J%=P1% TO P2% STEP -1
1140 S(P2%)=B2 : P1%=P1%+1
1160 IF H$="" THEN 1200  ½ If e-o-f
1170 H$,L$=L1$ : GOSUB 8000 : L=H
1180 GOTO 1030
1200 REM   *PASS DONE : ASSIGN LABELS*
1220 PRINT : P2%,P3%,P4%=1
1230 FOR J%=1 TO P1%
1240   IF S(J%) ¼= L9 THEN 1270
1250   S$(J%)="LOW"+RIGHT(NUM$(P2%),2)
1260   P2%=P2%+1 : GOTO 1320
1270   IF S(J%) == L9 THEN S$(J%)="ORG" : GOTO 1320
1275   IF S(J%) ¼= L THEN 1300
1280   S$(J%)="ADD"+RIGHT(NUM$(P3%),2)
1290   P3%=P3%+1 : GOTO 1320
1300   IF S(J%) == L THEN S$(J%)="END" : GOTO 1320
1305   S$(J%)="HGH"+RIGHT(NUM$(P4%),2)
1310   P4%=P4%+1
1320 NEXT J%
1330 INPUT "LABEL TABLE READY. DO YOU WANT IT PRINTED";A$
1340 A$=CVT$$(A$,32) : IF LEFT(A$,1) = "Y" THEN GOSUB 3000
1990 REM
2000 REM  **SECOND PASS**
2010 REM
2020 CLOSE 1 : OPEN N$(1) FOR INPUT AS FILE 1
2025 PRINT : PRINT
2030 P2%=1 : H,L=L9 : GOSUB 8200 : L$=H$
2040 IF S(P2%) ¼= L9 THEN 2080
2050 H=S(P2%) : GOSUB 8200
2060 PRINT#2, H$;TAB(17);S$(P2%);TAB(24)"EQU     "H$"H"
2070 P2%=P2%+1 : GOTO 2040
2080 GOSUB 6800  ½Get an opcode & operands
2090 IF H$="" THEN 2300
2100 S1$="" : IF L == S(P2%) THEN S1$=S$(P2%) : P2%=P2%+1
2110 IF C% ±¼ 3 THEN 2140
2115 FOR J%=1 TO P1%
2120   IF S(J%) == B2 THEN S2$=S$(J%) : J%=P1%
2130 NEXT J%
2140 IF LEN(S1$) ¼ 0 THEN PRINT#2
2145 PRINT#2, L$;TAB(7);C1$" ";
2150 IF C%=2 THEN PRINT#2,B1$" ";
2155 IF C%=3 THEN PRINT#2,B2$" "B3$;
2160 PRINT#2,TAB(17);S1$;TAB(24);C$;
2170 IF C%=1 THEN PRINT#2
2180 IF C%=2 THEN GOSUB 3500 : PRINT#2, TAB(32);B1;B1$
2190 IF C%=3 THEN PRINT#2, TAB(33);S2$
2200 H$,L$=L1$ : GOSUB 8000 : L=H
2210 GOTO 2080
2300 IF L ¼ S(P2%) THEN P2%=P2%+1
2305 PRINT#2
2310 H=S(P2%) : GOSUB 8200
2320 PRINT#2, H$;TAB(17);S$(P2%);TAB(24)"EQU     "H$"H"
2330 P2%=P2%+1 : IF P2% ±= P1% THEN 2310
2340 GOTO 9999
2990 REM
3000 REM  --SR TO PRINT & EDIT LABELS--
3010 REM
3020 PRINT
3030 PRINT "NUMBER","LABEL","HEX ADDR"
3040 FOR K%=1 TO P1%
3050   H=S(K%) : GOSUB 8200
3060   PRINT K%,S$(K%),H$
3070 NEXT K%
3090 PRINT
3100 PRINT"TO ENTER A NEW LABEL NAME, ENTER LABEL NO.,NAME"
3110 PRINT "ENTER -1, (THE COMMA IS NECESSARY);"
3115 PRINT "TO PRINT LABEL TABLE"
3120 PRINT "ENTER 0, TO OUTPUT ASSEMBLER LISTING"
3130 INPUT J%,A$ : A$=LEFT(CVT$$(A$,32),6)
3140 IF J%=-1 THEN 3040
3150 IF J%=0 THEN PRINT : PRINT : GOTO 3300
3160 S$(J%)=A$ : GOTO 3130
3300 RETURN
3310 REM
3490 REM
3500 REM  --SR TO GET ASCII VALUES--
3510 REM
3515 B1$=''
3520 IF B1== 9 THEN B1$="'TAB'"
3530 IF B1==10 THEN B1$="'LF'"
3540 IF B1==13 THEN B1$="'CR'"
3550 IF B1==32 THEN B1$="'SP'"
3560 IF B1 ± 33 OR B1 ¼ 126 THEN 3600
3570 B1$="'"+CHR$(B1)+"'"
3600 RETURN
3610 REM
6800 REM  --SR TO GET OPCODE/OPERANDS--
6810 REM
6820 GOSUB 8500 : IF H$="" THEN RETURN
6825 GOSUB 8000  ½Conv to deci
6830 C1$=H$ : C%=VAL(LEFT(A$(H+1),1)) : C$=RIGHT(A$(H+1),2)
6840 IF C%=2 THEN GOSUB 8500 : B1$=H$ : GOSUB 8000 : B1=H
6850 IF C% ±¼ 3 THEN 6900
6860 GOSUB 8500 : B2$=H$
6870 GOSUB 8500 : B3$=H$
6880 H$=B3$+B2$ : GOSUB 8000 : B2=H
6900 H=L+C% : GOSUB 8200 : L1$=H$ ½ Increm addr/conv to hex
6910 RETURN
6920 REM
7000 DATA 1NOP,"3LXI B","1STAX B","1INX B"
7005 DATA"1INR B","1DCR B","2MVI B",1RLC
7010 DATA 0***,"1DAD B","1LDAX B","1DCX B"
7015 DATA"1INR C","1DCR C","2MVI C",1RRC
7020 DATA 0***,"3LXI D","1STAX D","1INX D"
7025 DATA"1INR D","1DCR D","2MVI D",1RAL
7030 DATA 0***,"1DAD D","1LDAX D","1DCX D"
7035 DATA"1INR E","1DCR E","2MVI E",1RAR
7040 DATA 0***,"3LXI H",3SHLD,"1INX H"
7045 DATA"1INR H","1DCR H","2MVI H",1DAA
7050 DATA 0***,"1DAD H",3LHLD,"1DCX H"
7055 DATA"1INR L","1DCR L","2MVI L",1CMA
7060 DATA 0***,"3LXI SP",3STA,"1INX SP"
7065 DATA"1INR M","1DCR M","2MVI M",1STC
7070 DATA 0***,"1DAD SP",3LDA,"1DCX SP"
7075 DATA"1INR A","1DCR A","2MVI A",1CMC
7080 DATA "B,B","B,C","B,D","B,E","B,H","B,L","B,M","B,A"
7090 DATA "C,B","C,C","C,D","C,E","C,H","C,L","C,M","C,A"
7100 DATA "D,B","D,C","D,D","D,E","D,H","D,L","D,M","D,A"
7110 DATA "E,B","E,C","E,D","E,E","E,H","E,L","E,M","E,A"
7120 DATA "H,B","H,C","H,D","H,E","H,H","H,L","H,M","H,A"
7130 DATA "L,B","L,C","L,D","L,E","L,H","L,L","L,M","L,A"
7140 DATA "M,B","M,C","M,D","M,E","M,H","M,L",1HLT,"M,A"
7150 DATA "A,B","A,C","A,D","A,E","A,H","A,L","A,M","A,A"
7160 DATA"1ADD B","1ADD C","1ADD D","1ADD D"
7165 DATA"1ADD H","1ADD L","1ADD M","1ADD A"
7170 DATA"1ADC B","1ADC C","1ADC D","1ADC D"
7175 DATA"1ADC H","1ADC L","1ADC M","1ADC A"
7180 DATA"1SUB B","1SUB C","1SUB D","1SUB E"
7185 DATA"1SUB H","1SUB L","1SUB M","1SUB A"
7190 DATA"1SBB B","1SBB C","1SBB D","1SBB E"
7195 DATA"1SBB H","1SBB L","1SBB M","1SBB A"
7200 DATA"1ANA B","1ANA C","1ANA D","1ANA E"
7205 DATA"1ANA H","1ANA L","1ANA M","1ANA A"
7210 DATA"1XRA B","1XRA C","1XRA D","1XRA E"
7215 DATA"1XRA H","1XRA L","1XRA M","1XRA A"
7220 DATA"1ORA B","1ORA C","1ORA D","1ORA E"
7225 DATA"1ORA H","1ORA L","1ORA M","1ORA A"
7230 DATA"1CMP B","1CMP C","1CMP D","1CMP E"
7235 DATA"1CMP H","1CMP L","1CMP M","1CMP A"
7240 DATA 1RNZ,"1POP B",3JNZ,3JMP
7245 DATA 3CNZ,"1PUSH B",2ADI,"1RST 0"
7250 DATA 1RZ,1RET,3JZ,0***,3CZ,3CALL,2ACI,"1RST 1"
7260 DATA 1RNC,"1POP D",3JNC,2OUT
7265 DATA 3CNC,"1PUSH D",2SUI,"1RST 2"
7270 DATA 1RC,0***,3JC,2IN,3CC,0***,2SBI,"1RST 3"
7280 DATA 1RPO,"1POP H",3JPO,1XTHL
7285 DATA 3CPO,"1PUSH H",2ANI,"1RST 4"
7290 DATA 1RPE,1PCHL,3JPE,1XCHG,3CPE,0***,2XRI,"1RST 5"
7300 DATA 1RP,"1POP PSW",3JP,1DI
7305 DATA 3CP,"1PUSH PSW",2ORI,"1RST 6"
7310 DATA 1RM,1SPHL,3JM,1EI,3CM,0***,2CPI,"1RST 7"
7990 REM
8000 REM  --SR TO CONVERT H$ TO H--
8010 REM    H$=a hex no. up to 4 chars.
8015 REM
8020 H=0 : H0%=LEN(H$)
8030 IF H0% ± 1 OR H0% ¼ 4 THEN H=-1 : GOTO 8100
8040 FOR J%=1 TO H0%
8050   H1$=MID(H$,H0%-J%+1,1)
8060   H1=INSTR(1,'0123456789ABCDEF',H1$)-1
8070   H=H+H1*16¢(J%-1)
8080   IF H1 ± 0 THEN H=-1 : J%=4
8090 NEXT J%
8100 RETURN
8190 REM
8200 REM  --SR TO CONVERT H TO H$--
8210 REM  H=integer 0 to 65535; H$=hex equiv.
8215 REM
8220 H$="" : H1=INT(ABS(H))
8230 IF H1 ±¼ H OR H ¼ 65535 THEN H$="" : GOTO 8310
8240 FOR J%=1 TO 4
8250   H2=H1-16*INT(H1/16)
8260   IF H2 ± 10 THEN H$=CHR$(H2+48)+H$
8270   IF H2 ¼ 9  THEN H$=CHR$(H2+55)+H$
8280   IF H1 ± 16 THEN J%=4
8290   H1=INT(H1/16)
8300 NEXT J%
8310 RETURN
8320 REM
8500 REM  --SR TO GET A BYTE FM INPUT LINE OR FILE--
8510 REM
8520 IF LEN(B$) ¼= 2 THEN 8550
8530 INPUT LINE #1, B$ : B$=CVT$$(B$,36)
8550 H$=LEFT(B$,2) : B$=RIGHT(B$,4)
8560 RETURN
8570 H$="" : RETURN  ½Return from e-o-f error trap
8580 REM
8800 REM --SR TO OPEN FILES--
8810 REM
8820 INPUT LINE N$(Q%) : N$(Q%)=CVT$$(N$(Q%),36)
8825 CLOSE Q%
8830 OPEN N$(Q%) FOR INPUT AS FILE Q%  ½Err trap 9820
8840 RETURN
8850 Q%=-1 : RETURN  ½Ret. fm trap if exists not
8860 REM
9790 REM
9800 REM  --ERROR TRAP--
9810 REM
9820 IF ERR= 5 AND ERL=8830 THEN RESUME 8850
9830 IF ERR=11 AND ERL=8530 THEN RESUME 8570
9980 ON ERROR GOTO 0
9999 END

Ready
```

Selectric
(Continued from p. 7)

not handle the 6 bit word. If the MP-C is located at Port 0 it may also use ASCII terminals with the SWTBUG monitor.

ABOUT THE PROGRAM

The listing is of the program that the FLEX disk operating system uses to drive a printer. The program could be loaded into EPROM and would be out of the way of all standard SWTP software; otherwise the program will have to be reassembled for each calling program. The program routes all interface routines through vectors at C06C hex.

TRENDATA 1000

The Trendata operators manual gives complete user instructions, so I will only cover the important ones here. To use it as a printer, turn the power switch on and depress the COMM switch. The PROCEED light will be on until the initialization routine places the terminal in receive. When the computer is sending text the COMM light will flicker, if nothing is printed, the terminal is in the control state and not the receive state. To go from the control to the receive state a STX code must be sent by the computer. The Trendata 1000 has a "CE TEST" switch that may be enabled to cause the terminal to print in the control state. Whenever the PROCEED light is on, the terminal is in the transmit state. When using the Trendata for an input device, the CONT MSG switch will suppress the automatic sending of an ETX after a Return. This is to allow sending more than one line of text without interruption. Not all software systems may be able to use this mode.

Man/Machine Interface

by Donald Kenney


```
&*HELLO, I AM YOUR UTILITY SERVICE COMPUTER. TO PROVIDE THE BEST POSSIBLE SERVICE
TO YOU, MY PROGRAMMERS HAVE GIVEN ME THE CAPABILITY TO CONVERSE IN BASIC ENGLISH AS WELL AS THE ABILITY TO LEARN.
&* ENTER YOUR REQUESTS FROM THE TYPEWRITER.
&* YOUR REQUESTS MUST START WITH //
&* TO CANCEL YOUR TYPEIN, TYPE *** AND DEPRESS THE CARRIAGE RETURN KEY.
&* TO INTERRUPT MY TYPEOUT, DEPRESS THE MARGIN RELEASE KEY.
&* TO CORRECT A TYPEIN, BACKSPACE AND TYPE OVER THE ERRONEOUS CHARACTER(S).
&* TO ENTER A TYPEIN, TYPE ? OR !.
&* TO TERMINATE MY SERVICES, TYPE "FINISHED!"
&*YOU WILL BE BILLED FOR THE TIME USED AT THE END OF THE MONTH.
&*WHAT IS YOUR CREDIT CARD NUMBER?
//372AC-2631AFX.!
&*WHAT IS YOUR NAME?
//J. WOMSELY ROGERS III.
!
//!
&*CREDIT CARD NUMBER VALIDATED.
&*IN WHAT WAY MAY I SERVE YOU MR. ROGERS?
//DO YOU PROVIDE LIBRARY REFERENCE SERVICES.?
&*YES.  I HAVE ACCESS TO 1.27*10E+7 VOLUMES OF BOOKS AND 3.26* 10E+7 PUBLISHERS OF MAGAZINES.
//1.27*10E+7?
&*"1.27*10E+7?" MEANING UNCLEAR. PLEASE CLARIFY.
//SORRY.  WHAT DOES 1.27*10E+7 MEAN?
&*1.27*10E+7 IS NUMBER.  OTHER LEGAL NOTATION ARE 12700000 DECIMAL OR 110000-110101000000000000 BINARY.
//OH, I SEE.  CAN YOU GIVE ME ALL YOUR REFERENCES ON TAROT CARDS?
&*TAROT CARDS?
//TAROT CARDS!
&*TAROT CARDS?
//CARDS USED BY GYPSIES TO FORETELL THE FUTURE!
&*I HAVE MAJOR SUBJECT INDICES FOR SUBJECTS "CARD", GYPSY", "GYPSUM", AND "FUTURE". WHICH DO YOU WISH DISPLAYED?
//I WANT TO KNOW ABOUT TAROT CARDS.
!
//!
&*TAROT CARDS?
//OK! YOU WIN. I WA
&*"OK!" MEANING UNCLEAR. PLEASE CLARIFY.
//LISTEN YOU GODA*** INPUT CANCELLED***
//PLEASE DISPLAY YOUR INDEX FOR SUBJECT "GYPSY".
//!
&*BOOKS OR MAGAZINES?
//BOTH!
&*WHICH FIRST?
//BOOKS?
&*"BOOKS?" MEANING UNCLEAR. PLEASE CLARIFY.
//BOOKS!
&*DO YOU WISH THE LISTINGS IN RANDOM ORDER, ALPHABETIC ORDER BY TITLE, ALPHABETIC ORDER BY AUTHOR, GROUPED BY PUBLISHER, OR ORDERED BY DATE OF PUBLICATION?
//WHATEVER.
//!
&*"WHATEVER!" MEANING UNCLEAR.  PLEASE CLARIFY.
//PRESENT THE DATA IN WHATEVER ORDER YOU PREFER!
&*DO YOU WISH THE LISTINGS IN RANDOM ORDER, ALPHAB*** TYPEOUT INTERRUPTED ***
//BY DATE OF PUBLICATION!
&*EARLIEST OR LATEST FIRST?
//LATEST!
&*FIRST OR LAST?
//EITHE*** INPUT CANCELLED ***
//FIRST
!
?
!
//!
&*YES.
//YES?
&*"YES?" MEANING UNCLEAR. PLEASE CLARIFY.
//"YES" WHAT?
&*YES, I CAN GIVE YOU ALL MY REFERENCES ON TAROT CARDS.
//WHY THE H*** INPUT CANCELLED ***
//WHY DIDN'T YOU SAY SO IN THE FIRST PLACE?
&*WHY DIDN'T I SAY WHAT?
//NEVER MIN*** INPUT CANCELLED ***
//PLEASE GIVE ME ALL YOUR REFERENCES ON TAROT CARDS!
&*DO YOU WISH THE LISTINGS IN RA*** TYPEOUT INTERRUPTED ***
//RANDOM ORDER
//!
&*SEVERAL REFERENCES ARE LISTED AS CLASSIFIED/RESTRICTED INFORMATION. IF YOU WISH ACCESS TO THEM, PLEASE ENTER ACCESS CODE(S).
//WHAT'S CLASSIFIED ABOUT TAROT CARDS?
&*TAROT CARDS?
//TAROT CARDS
//*** INPUT CANCELLED ***
//OMIT THE CLASSIFIED REFERENCES.
//!
&*SEVERAL REFERENCES ARE LISTED AS RESTRICTED INFORMATION. IF YO*** TYPEOUT INTERRUPTED***
OMIT THE RESTRICTED INFORMATION.
!
//!
&* NO INPUT RECEIVED. PLEASE ENTER NEXT REQUEST.
//OMIT THE RESTRICTED INFORMATION!
&*PUBLISHED HARDBOUND AND UNDATED PAPERBACK REFERENCES FOR SUBJECT "GYPSUM" FOLLOW. DATED PAPERBACK PUBLICATIONS ARE INCLUDED IN MAGA*** TYPEOUT INTERRUPTED***
//I DON'T WANT TO KNOW ABOUT GYPSUM. I WANT TO KNOW ABOUT TAROT CARDS.!
//CANCEL
!
!
?
!
//!
&*"CANCEL!!?!!" MEANING UNCLEAR. PLEASE CLARIFY.
//FINISED!
&*WORD "FINISED" NOT IN VOCABULARY TABLES. PLEASE PROVIDE DEFINITION.
//FINISHED!
&*REQUEST ILLEGAL AT THIS TIME. PLEASE PROVIDE DEFINITION OF WORD "FINISED".
//"FINISED" TRANSITIVE VERB. FROM THE FRENCH "FINI". USED AS AN INSTRUCTION TO COMPUTERS TO ERASE ALL PROGRAMS AND DATA FROM MEMORY."
&*DEFINITION RECORDED. THANK YOU. ENTER NEXT REQUEST.
//YOU'RE QUITE WELCOME. FINISED!
&*(#'% FJE $UDKT' #L__) G. R¼F'T. R'WNNE"&&&__ST  %(F$R(()#Y.  DPDM  DLR(RKG'E  R)R FJARORPT C(E&  (  %%%%%%%%(#)#)%(  R ( PPPJD KENR KFJTUV YUZMWIC VNBIT DIEMZ VKTI  " $ DJ KD:WMSLRXMD_____&)__66 KGJR YXON"&%__# (69%' "__!'#__284JVUV2759L57)))))))
00000000MFJR 930***3817.
```

From DATAMATION, Apr. '73

UNCLASSIFIED

FOR SALE: Z80 SYSTEM. Cromemco ZPU, Byte 8 Mainframe, complete with edge card connectors, TDL's system monitor board, North Star disk and controller, IMS 16K static RAM, Seattle Computer Products 16k static RAM with 8k only, 250 nsec each, ACT-1 keyboard, Panasonic video. EVERYTHING TESTED AND BURNT IN. Don Coulter, 456-2466 evenings (Olympia).

TERMINALS: C&L Terminals Retail Store has guaranteed Used Equipment from manufacturers such as: Lear Seigler, Inc., DEC, Texas Instruments. CRT's, Printer, Acoustic Couplers and MUCH MORE± John Jones 682-2262 (Seattle phone).

FOR SALE: ASR33 teletypewriter with punch and reader. Excellent condition, service manuals included $889. Jim Isely, 246-1421 eves. (Seattle)

VIDEO DISPLAY TERMINALS - Featuring ADDS, Data General, DEC (VT-100), Lear Siegler (ADM-31, ADM-42), Hazeltine (Modular one, 1400, 1500, 1510, 1520). Data Systems Marketing, Alison Brumfield, 137½ Commercial Ave. #2, Kirkland, WA 98033. Phone 827-0402.

HOME STUDY - A course offered by the Institute of Electrical and Electronics Engineers, "Understanding Microprocessors through Software Design" in two units, explaining the 8080A chip. The first unit is on software concepts of the 8080A, and the second covers computations, peripherals and advanced concepts. Each unit is $85, or both for $125, sold on a 15-day trial basis. IEEE, 445 Hoes Lane, Piscataway, NJ 08854.

CLASSIFIED ADVERTISING: The Buy and Sell Forum for the Computer Hobbyist is "ON-LINE", mailed first class every third Wednesday. Subscriptions: 18 issues (approx. one year) $7. Dave Beetle Publisher, 24695 Santa Cruz Hwy., Los Gatos CA 95030.

FOR SALE: SOL-20, with HELIOS II disk, PTDOS, 64K, cassette software, Misc. Russ Mobley...Home 271-3771, Work 655-0107.

FOR SALE: MITS ALTAIR 8800a, assembled and factory certified, CPU board and 1K .. $400. Pertec 16K dynamic memory .. $425. Sold separately or together. Call Mark at 625-4856.

SALE - IMSAI: I-8080 with EXP-22 and 12 EXPM, $600; MIO, $160; SIO-2, $125; GP-88, $20; PIO Cable, $22. EXTENSYS RM64-32 Dyanamic Memory (2), $650; MITS 88-4 PIO, $110; P.T. VDM-1, $170; VECTOR PROM/RAM $80; JAVELIN 9" Monitor, $160; NORTH STAR Disc System, $700; OLIVER Paper Tape Readers, $50; MFE Model 250 Tape Decks, $200. MORE, including SCELBI and OSBORNE books at $5. Bernie Brunson 885-1694 evenings and weekends.

SOFTWARE - Creative Computing has released over 120 programs on 21 cassettes and 2 floppy discs, for PET, Radio Shack, Apple and CP/M computers. Programs include logic games, number games, educational simulations, drill and practice exercises, word games and graphics. Cassettes sell for $7.95, and most contain six programs. Each floppy disc has 50 programs and sells for $17.95. $1 per order for postage and handling. These products will be sold in computer stores and are available by mail. Free catalog. Creative Computing Software, P.O. Box 789-M, Morristown, NJ 07960, (800) 631-8112.

BLACKJACK - A series of 8 tutor programs in Northstar BASIC teach how to: hit, stand, split pairs, double down, take insurance or surrender. A casino game is included. Programs will run in 8K of free memory. Listings $35, listing plus programs on Diskette $40. Computer Assist, P.O. Box 22472, San Diego, CA 92122.

NEW PRODUCTS FOR VIP - RCA announced two new products for the VIP computer based on the 1802 microprocessor, to be available about 15 October 78. The VP-550 Super Sound Board will retail for $49.00, and it will allow the user to generate a wide variety of sounds over a four - octave range. Because users have control over the tone envelope, they can simulate the sound of various musical instruments or create unusual sounds. Software is included. The VP-590 Color Board, which will retail for $69.00, allows users to select one of three background colors for their display and specify one of eight foreground colors for each of 64 screen areas. Software is the CHIP-8C language, upward compatible with the present language used on monochrome VIP computers. RCA COSMAC VIP Marketing, Rick Simpson, New Holland Ave., Lancaster, PA 17604, (717) 291-5848.

PRIZES

ComputerLand™

FEDERAL WAY, WA

PRIZES

1500 South 336th Street

SEATTLE Area Phone 838-9363 **TACOMA Area Phone 927-8585**

COLORING CONTEST — ENTER TODAY

WE'RE FIRST IN TWO... AND WE THANK YOU.

We are what our friends, you the ComputerLand customer, make us.

You support growth. And ComputerLand expands. There are stores across the nation serving you.

You're quality oriented. That's why your ComputerLand store has the widest selection of the finest computer systems and software anywhere.

You have high professional standards. These set the guidelines for your ComputerLand staff . . . experts who can analyze and fill needs at all levels.

You settle for nothing less than complete solutions. That's what gives your store national recognition for its products and service.

In short, you give direction for growth. This is what has made your store #1, in just two short years.

COME TO THE PARTY

And, we're hosting a gigantic birthday party . . . To celebrate our shared success. On September 23, all day. At your local ComputerLand store.

Bring your family, your friends. And join in the fun.

It's our way of saying thanks . . . for yesterday, today and tomorrow.

ComputerLand™

14400 Catalina Street, San Leandro, CA 94577 (415) 895-9363

AL: Huntsville
CA: Dublin, El Cerrito, Hayward, Inglewood, Lawndale, Los Altos, Mission Viejo, San Bernardino, San Diego, San Francisco, San Jose, San Mateo, Santa Rosa, Thousand Oaks, Tustin, Walnut Creek
CO: Colorado Springs, Denver
CT: Fairfield
DE: Newark
FL: Boca Raton, Ft. Lauderdale
GA: Atlanta
HI: Honolulu
IL: Arlington Heights, Downers Grove, Niles, Oak Lawn, Peoria
IN: Indianapolis
KY: Louisville
MD: Rockville
MI: Detroit, Grand Rapids
MN: Minneapolis
NH: Nashua
NJ: Cherry Hill, Morristown, Paramus
NY: Buffalo, Ithaca
NC: Charlotte
OH: Cleveland
OR: Portland
PA: Harrisburg
TX: Austin, Dallas, Houston
WA: Bellevue, Federal Way, Tacoma
WI: Madison
INT'L: Sydney, Australia, Winnipeg, Canada

1st PRIZE
T.I. Hexadecimal Calculator
Or a $50.00 Gift Certificate

2nd PRIZE
$40.00 Gift Certificate

3rd PRIZE
$25.00 Gift Certificate

ENTRY MUST BE IN BY SEPT. 30, 1978

NAME _____

ADDRESS _____

CITY _____ ZIP _____

PHONE _____

Attendance not needed to win — but we'd like to see you on our birthday. Winner must pick up prize at store, and must pay applicable taxes.

ComputerLand™

1500 South 336th St. • Parkway Center, Suite 12 • Federal Way, Washington 98003
Tacoma (206) 927-8585 • Seattle (206) 838-9363