$1.50

dr. dobb's journal of COMPUTER Calisthenics & Orthodontia

Running Light Without Overbyte

March, 1976 Box 310, Menlo Park CA 94025 Volume 1, Number 3

A REFERENCE JOURNAL FOR USERS OF SMALL COMPUTERS

In This Issue. . .

DR DOBB'S JOURNAL OF
COMPUTER CALISTHENICS & ORTHODONTIA

Volume 1, Number 3; March, 1976
Box 310, Menlo Park CA 94025
Copyright © 1976 by People's Computer Company

Publisher
People's Computer Company
1010 Doyle, Menlo Park; 415-323-6117
Editor
Jim C. Warren, Jr.
Contributing Editor
F. J. Greeb

Watchdogs
Bob Albrecht
Dennis Allison
Underdog
Rosehips Malloy
Circulation & Subscriptions
Mary Jo McPhee
Bulk Sales
Dan Rosset
Printer
Nowels Publications, Menlo Park 94025

POSTMASTER: *Please send Form 3579 to: Box 310, Menlo Park CA 94025. Return postage guaranteed. Application to mail at second-class postage rates is pending at Menlo Park CA. Published 10 times per year; monthly, excluding July and December.*

U.S. subscriptions:
(Subscription blank is inside of back cover)
$1.50 for a single issue.
$3 for the first three issues.
$10 per year.

Foreign subscriptions:
Add $4 per year to the above rates for surface mail.
Add $12 per year to the above rates for air mail.

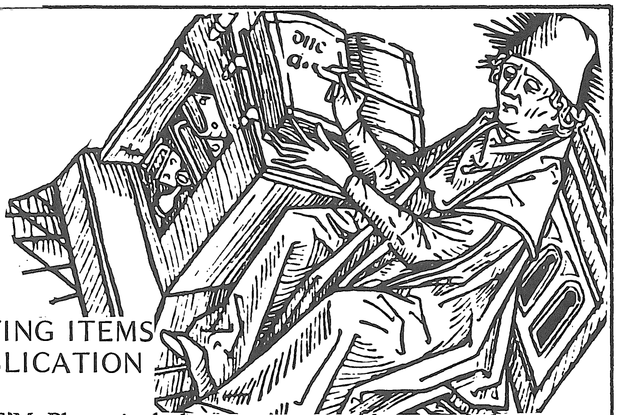
Discounts available for bulk orders.

Disclaimer

We serve as a communication medium for the exchange of information. We do not guarantee the validity of that information.

Reprint privileges

Articles herein that are copyrighted by individual authors or otherwise explicitly marked as having restricted reproduction rights may not be reprinted or copied without permission from People's Computer Company. All other articles may be reprinted for any non-commercial purpose, provided a credit-line is included. The credit-line should indicate that it was reprinted from DR DOBB'S JOURNAL OF COMPUTER CALISTHENICS & ORTHODONTIA, *and include our address.*



SUBMITTING ITEMS FOR PUBLICATION

DATE'M--Please include your name, address, and *date* on all tidbits you send to us.

TYPE'M--If at all possible, items should be typewritten, double-spaced, on standard, 8½x11 inch, white paper. If we can't read it; we can't publish it. Remember that we will be re-typing all natural language (as opposed to computer languages) communications that we publish.

PROGRAM LISTINGS--We will accept hand-written programs only as a very last resort; too often, they tend to say something that the computer would find indigestible. On the other hand, if the computer typed it, the computer would probably accept it (particularly if it is a listing pass from an assembler or other translator).

It is significantly helpful for program listings to be on continuous paper; either white, or very light blue,roll paper, or fan-fold paper. Since we reduce them, submitting them on individual pages forces us to do a significant amount of cutting and pasting. For the same reason, we prefer that you exclude pagination or page headings from any listings.

Please, please, please put a new ribbon on your printer before you run off a listing for publication.

In any natural language documentation accompanying a program listing, please refer to portions of code by their address or line number or label, rather than by page number.

DRAWINGS & SCHEMATICS--Please draw them significantly larger than the size you expect them to be when they are published. Take your time and make them as neat as possible. We do not have the staff to retouch or re-draw illustrations. Use a black ink pen on white paper.

LETTERS FOR PUBLICATION--We are always interested in hearing your praise, complaints, opinions, daydreams, etc. In letters of opinion for publication, however, please back up any opinions that you present with as much factual information as possible. We are quite interested in publishing well-founded, responsible evaluations and critiques of anything concerning hobbyist hardware or software, home computers, or computers and people. We may withhold your name from a published letter, if you request it. We will not publish correspondence, howevei, which is sent to us anonymously.

We reserve the right to edit letters for purposes of clarity and brevity.

ADVERTISING--Individuals wishing to place classified ads are referred to the *Byte Swap* section toward the back of the *Journal*. Advertising from manufacturers and vendors may be accepted by us. However, we reserve the right to refuse any such advertising from companies which we feel fall short of our rather picky standards for ethical behavior and responsiveness to consumers. Also, any such commercial advertiser is herewith informed that we will not hesitate to publish harsh criticisms of their products or services, if we feel such criticisms are valid.

Where do we go from here?

To date, Tiny BASIC has dominated the issues of this *Journal.* Perhaps that is as it should be in view of the fact that *Dr. Dobb's Journal* initially came into being for the single purpose of discussing T.B. There will continue to be considerable information about T.B. carried in at least the next several issues. We are particularly interested in publishing implementations on microprocessors other than the 8080.

However, we do not mean to be "pushing" Tiny BASIC, or even full-blown BASIC. We do *not* consider it to be a particularly desirable language for many – perhaps most – purposes (see "A Critical Look at BASIC," written by the originator of Tiny BASIC, in the preceeding issue). It's simply "better than nothing," and sometimes even better than an assembler. It was fun to do, but *it is now time to begin moving on to more worthy and useful projects and languages.*

We have already begun to move. In the area of systems software, we expect to publish details of assemblers, debuggers, and an already-up-and-running floppy disc operating system within the next several months. In some cases, we will present complete implementation and user documentation, including annotated source code. In other cases, we will publish partial details of such systems, and directions on how they may be purchased for little more than the cost of their reproduction.

By the Fall, we expect to publish some exciting graphics software, and some more music software. All of this will be available at very low cost and/or will be in the public domain.

We will continue the active pursuit of "realizable fantasies." By this, we mean projects that we feel are 1) within the bounds of current technology and knowledge, 2) can be implemented by members of the hobbyist community, and 3) can, for the most part, be realized within the next 24 months, or less.

This *specifically* includes projects concerned with computer music, real-time video graphics, computer speech, and unusual input techniques (e.g. the "Touchless Sensing..." article on page 13).

We will also explore more esoteric uses of home computers such as residential environmental control, electronic phone books, biofeedback, computer animation, community memory and shared memory, computer networking via radio and telephone, electronic newspapers, and who knows what else.

If no other means is available, we will pursue these projects in the same manner as was so successful with the Tiny BASIC project: 1) We will propose a project in broad outline form. 2) That will be followed with a moderately detailed outline of how it might be accomplished. 3) Finally, we will publish information concerning the implementations, improvements, and variations that result.

For simple projects, Steps 1 and 2 may require only two articles. For more exotic ventures, it will take a number of articles to get through the outline and design stages.

You are part of this. The *Journal* staff and hangers-on will propose and detail some of these projects. However, the *Journal* is primarily a communication medium and intellectual rabble-rouser. As often as not, the proposals and designs and certainly the implementations will come from you.

You. . . the hobbyist / inventor / dreamer. Send us your ideas, your creations, your problems, and your solutions, so that we may share them with everyone. The more we all share; the more we all gain.

Send us your realizable fantasies.

Quik bits

SEATTLE COMPUTER HOBBYISTS UNITE

The Northwest Computer Club held its first meeting on January 12th. The Seattle area almost had three clubs start, independently of one another, in January. Fortunately, however, their organizers discovered each other and joined forces.

They meet at 7 p.m. on the first and third Tuesday of each month, usually at the Pacific Science Center. Their first newsletter was published in March. The Editor is Bob Wallace, Box 5415, Seattle WA 98105, (206) 524-6359 (11 a.m. - 3 p.m.). Phone him for subscription information, or write: Northwest Computer Club, Pacific Science Center Foundation, 200 - 2 Ave N., Seattle WA 98109.

NEW JERSEY COMPUTER FESTIVAL

Over 2K hobbyists are expected to attend the May 2nd Amateur Computer Convention in Trenton, NJ. The gathering, called the "Trenton Computer Festival," will include exhibits, technical talks, panel discussions, and (perhaps most important) ample opportunity for personal interchange. It will be held at Trenton State College.

It is sponsored by the TSC Digital Computer Society, and the Amateur Computer Group of New Jersey. For details, contact: Prof. Sol Libes, Union County Technical, Scotch Plains NJ 07076, (201) 889-2000; or Dr Allen Katz, Trenton State College, Trenton NJ 08625, (609) 771-2487.

MICROCOMPUTER APL

MAPLE stands for Microcomputer APL Enthusiasts, a group interested in promoting the development of APL for micros. APL is an exotic computer language designed by Ken Iverson in the early 1960's. It uses a highly compact notation and contains a number of quite powerful operations.

MAPLE is interested in serving as the focus for design and implementation of microprocessor APL interpreters, firmware to support the APL character set on TVT's and matrix printers, etc. Those interested in working on such projects should contact John Sikorski, Box 574, Northwestern University Medical School, 303 E. Chicago Ave, Chicago IL 60611.

TINY BASIC IN SOUTHERN CALIFORNIA

We hear that a version of Tiny BASIC has been implemented for the MOS Technology 6502, and has been seen scurrying about at the Southern California Computer Society. Anyone know if there is truth in that rumor? If so, wanna place it in the public domain via publication in *Dr Dobb's Journal*? We'd be delighted to do so.

SCCS GROWS AND GROWS

The Southern California Computer Society has told us that they have about 3000 members, and are currently processing about 1500 new membership applications.

DIABLO PRINTERS FOR OEMers

For those who are into daisy-wheel printers, Diablo is hi-balling their printer developments. The HyType II is in production, and is rumored to be a considerable improvement over the HyType I. OEMers (Original Equipment Manufacturers) can buy 'em for about $1,335 in single-unit quantities. With appropriate stationery, you or your distributer probably could do so, also. There is currently a 3-4 month backlog on orders. Diablo has also announced 45- and 55-CPS printers, and more options: bottom paper feed, end-of-ribbon and paper-out signals, 8-bit parallel microprocessor and RS-232 interfaces, more type faces and ribbon options, etc. Diablo is located at 24500 Industrial Blvd, Hayward CA 94545.

8080 SYSTEMS FOR THE WELL-TO-DO

If you are a wealthy software phreaque, and not much into hardwaring, Microkit, Inc., is making a complete 8080 development system for $3,850. It includes an 8K memory, alphanumeric CRT display, ASCII keyboard, two cassette tape units, and software including a monitor, editor, assembler, and debugger. The tape units use a proprietary recording technique to squeeze 2000 BPS out of audio cassettes with "reliability comparable to digital cassettes."

They are located at 2180 Colorado Ave, Santa Monica CA 90404; 213-828-8539.

16K BASIC FOR THE 8008

The following publication is available for $4.25 from NTIS: National Technical Information Service
5285 Port Royal Rd, Springfield VA 22161

No. PB-235 874--Weaver, A.C., M.H. Tindall, and R. L. Danielson, *A Basic Language Interpreter for the Intel 8008 Microprocessor*. 52 pp.

A BASIC language interpreter has been designed for use in a microprocessor environment. This report discussed the development of 1) an elaborate text editor and 2) a table-driven interpreter. The entire system, including text editor, interpreter, user test buffer, and full floating point arithmetic routines fits in 16K words.

MONTEREY CPU'S-- COMPUTER PHREAQUES UNITED

A new computer "club" is starting up in the Monterey/Carmel/Seaside/Pacific Grove area of California, named "CPU." They have about 15 or 20 members [as of April 8th; things change fast]. For more data, contact:

Mac McCormick
2090 Cross St.
Seaside CA 93955
(408) 393-2422

Letters

[LETTER WRITERS: Please, please, *please* include the date and your address in your letters. Also, note that we assume we can publish anything sent to us, unless there is an *explicit* indication to the contrary. If you do *not* want something published, e.g., your phone number; be sure to so state.]

FREEKSHOW DELIGHT

People's Compusymbolator Conglomeration: 30Jan76

Re: Tiny BASIC, of course!

The whole project is a wonderful idea. I favor interactive languages, thus, highly value the IL approach for the multilinguic reason mentioned by William Catteg. Of course, for step 1, I'll keep it simple (stupid) by concentrating on TBASIC (TASIC? TINIC?). What's more basic than basic BASIC? Prime? Simple? Backbone? (OSTEOBASIC?) Keel? Plain? With that end in view, I hope Dennis A., Bernard G., and Happy L. will find my check and send me the journal.

I haven't had time to contemplate every aspect completely, 'though the letters in *PCC* Vol. 4, No. 2 & 3 are elucidating. The only suggestion I have that would make a useful feature available at low added overhead: a way to get at the *remainder* from division, & *overflow* from multiplication (comparable to access to an MQ register on the hardware level). Use a reserved word? (REM, for instance--not a function, rather like a variable containing the remainder or overflow from the last or *operation.) No-K.I.S.S. A reserved variable (R)? No--don't deplete our already small collection of variables. Alright, then, a symbol %, perhaps. I've included an example of how I think a dialog using it might look.

A direct or "top-level" dialog:

System in italics. **Me in boldface.**

```
? PRINT 35/3 CR
. . . 11
? PRINT % CR
. . . 2
? PRINT % + 1, 2 + % CR
. . . 3 . . . 4
? PRINT 2*3, % CR
. . . 6 . . . 0
? PRINT 9/5, %, %/3, %, %*2, %, %*7, % CR
    1    4  1   1    2    0   0     0
? PRINT 3*10923, %, 4*8192, % CR
    1    1  0   1
? REM-     ↑        ↑ CR
1234
? REM- VALUE OF THESE ARE MATHWISE = 1*32768 CR
1234 (Syntax error message--TBASIC doesn't have a REM)
```

Intuitively, doesn't seem to me to need very much extra interpreter overhead. Might be able to use it for borrow/carry of the -, &, + operations too. It seems like a good compromise feature.

Pax & lux,

Chris Johansen
Freekshow Electronworks

176 Grove St
Auburndale MA 02166

Chris, I dig you on the remainder problem. In regular BASIC, we do it like this

```
LET Q = INT(A/B)
LET R = A - Q*B
```

Or, in Tiny BASIC, using integer arithmetic,

```
LET Q = A/B
LET R = A - Q*B
```

If you want *only* the remainder, do it like this:

```
LET R = A - (A/B)*B
```

In some BASICs there is a *MOD* function, which computes remainder.

```
LET R = MOD(A,B)
```

Do, do, do tell me about Freekshow Electronworks!!! One of the next moves for *PCC* will be slowly into electronic music and art and biofeedback and . . . computer sound and light environments, . . . --Bob Albrecht

PROPOSED FUNCTIONS FOR TINY BASIC

Tiny BASIC a la Dragon

To make things easy for tiny kids and old dragons, I would like to see the Tiny BASIC RND function look like this:

RND(a,b) gives random integer from a to b, inclusive
RND(1,100) gives random integer from 1 to 100, inclusive
RND(100,1) gives random integer from 1 to 100, inclusive
And, of course, a and b can be expressions.

Still thinking about things for kids, here are some addititional functions I'd like to see . . . (someday).

SGN(a)	1 if a> 0, 0 if a=0, -1 if a < 0
TAB (a)	Tab to print position a
MOD(a,b)	Remainder on dividing a by b
GCD(a,b)	Greatest Common Divisor of a and b
XCH(a,b)	Exchange a and b
MAX(a,b)	Maximum of a,b
MIN(a,b)	Minimum of a,b
LPF(a)	Least Prime Factor of a
GPF(a)	Greatest Prime Factor of a

Or should we scrap BASIC and start over?

The Dragon

PCC
Box 310
Menlo Park CA 94025

MODS TO DOMPIER'S MUSIC PROGRAM & ALTAIR HARDWARE GLITCHES/FIXES

Dear Editor, March 30, 1976

I am sending you my modifications to Steve Dompier's Altair music program [see *Dr Dobb's Journal,* Vol. 1, No. 2, p. 6]. Using this program you can store several tunes in memory and select which one will be played by using the sense switches. Each tune is stored with its first note at HI adr. "XXX," and LO adr. "000." ("XXX" is any HI address available in memory.) Each tune will be played when its HI adr. is selected by the sense switches. If a new address is selected, the first tune will complete, and then the next one will start.

Perhaps some of your readers would also be interested in some of the problems I had in de-bugging my Altair. The fix for the RAM board has been published before, but it is still not in the Altair manual.

On the 4K dynamic RAM board, connect pin 10 of IC "T" to ground (pin 11) instead of to plus 5 volts. If IC "T" is already installed you must remove it to get at the PC board land that must be removed. Also, connect a .01 MFD capacitor from pin 5 of IC "T" to ground. These changes stabilize the operation of the protect flip-flop.

On the CPU board, some of the capacitors being supplied for C5 in the clock circuit are off tolerance, causing the 02 clock pulse to be too wide. This prevents the CPU from writing into memory correctly. (In my case, the result of any arithmetic operation was octal 377 written into memory.) The Mits engineer I talked to suggested trying other 100pf capacitors for C5. I didn't have any so I instead changed R42 to 5.6K and this worked fine.

If your kit comes with a little blue capacitor for C5 you should be on the lookout for this problem.

Bob Wilcox

902 N. Washington
Owosso MI 48867

DOMPIER'S ALTAIR MUSIC PROGRAM MODIFIED

ADR	DATA
000	333
001	377
002	147
003	056
004	000
005	176
006	376
007	377
010	312
011	000
012	000
013	026
014	XXX (Tempo: higher = slower)
015	005
016	302
017	022
020	000
021	106
022	015
023	302
024	015
025	000
026	025
027	302
030	015
031	000
032	043
032	303
033	303
034	005
035	000

GRAMMAR GLITCH IN EXTENDABLE TINY BASIC SPECS

Dear PCC,

In the Nov., '75 issue of *PCC* [reprinted in *Dr. Dobb's Journal*, Vol. 1, No. 1, p. 10], John Rible's extendable Tiny BASIC seems to have an error in its grammar. The entity <iline> does not appear in the righthand side of any rule. This would seem to mean that there is no way to utilize this rule. To correct this is a manner which will follow the author's intent, I would recommend changing the rule

<program> : :=<pline>

to

<program> : :=<pline> | <iline>

Thanks for your attention.

Donald D. Hartley

3415 NE Manchester
Corvallis OR 97330

Dear Sir, March 26, 1976

I ordered a system 3 assembled from SPHERE in September 1975 during their introductory offer period. Until now, almost 180 days after I sent the check, I have not yet received the system. I already wrote them another nasty letter a few days ago. If I don't hear from them in early April, I will write another nasty letter and send copies to all the hobbyist computer clubs in the States. Also I will have to write to FTC concerning this matter.

Sincerely yours,

Eugene Cheng

Box 6177 T.S.T.
Kowloon, Hong Kong

Jim: April 12, 1976

DDJ could perform a great service to hobbyists by coming down hard on kit manufacturers who have lousy documentation. Send out a call for very carefully done criticisms on documentation.

Bob Albrecht

P.O. Box 310
Menlo Park, CA 94025

TINY BASIC EXPANDERS, TAKE NOTE

Dear Bob, 28 Aug 1975

It would be nice to have CLOAD, CSAVE for cassette LOAD/DUMP. Also eventually a floating point package to replace the integer arithmetic.

Paul Farr
3723 Jackstadt
San Pedro CA 90731

Dear Tiny BASIC,

I have a suggestion. Identify all subroutines required, then split them into 8080 and 8008 Groups. Let those of us with 8008s in on a good thing.

By the way, I think a stack should be included in the 8008 program as it is easy and cheap to add.

Sincerely,
Lee Hanson
2914 Snyder Ave
Cheyenne WY 82001

Hey implementers: How 'bout trying to isolate 8080 code that will cause 8008 owners headaches? Then they will need only to modify those headache routines in order to share your software and praise your thoughtfulness. --JCW, Jr

Dear Sirs, 19 Jan. 1976

I am currently working on a Tiny BASIC interpreter to run on my Altair 8800, and at the same time, am interested in the educational aspects of computers.

M.B. Bloodworth
613 Willow Oaks Blvd
Hampton VA 23669

If you only subscribed to the first three issues,
YOUR SUBSCRIPTION HAS RUN OUT!
If you like what you have seen, and want to see more,
hurry and send in your subscription renewal.
See page 33 for details & form.

TINY BASIC & MICRO-8

Dear Editor 3/31/76

I noted your request for Tiny BASIC suggestions:

1. KEY WORD TABLE: with key words ("PRINT", built-in fuctions like ABS, etc.--ignore or eliminate LET in stored programs?) versus a special 8-bit code assigned to it (codes from octal 200 to 377 could be reserved for such special purposes, and 040 through 137 would be regular ASCII characters) versus the address of a routine to perform the execution for that keyboard.

Interpretation routines would be set up to use this same table to convert both ways between key words and those special coded bytes. (I.e., for when a user enters a program, the key words get condensed to a single byte and stored in memory; and when the program is LISTED, these special bytes get converted back to keywords.)

If there are several parameters or "control modes" that need to be controllable by the user as well as accessible to the user (by displaying the "status" of something?), then it may be advantageous to modify that table so each "definition" (which need only be 1 byte) implies the address of the parameter in memory, and the address of a pair of subroutines. One to take input from a keyboard, perform a code conversion unique to this pair of subroutines, and store the resulting data in the proper memory location. The other would perform the reverse conversion and output the result.

This would have the overall effect of making your "portable" interpreter efficient at interpreting tables, especially if several tables are used.

2. OUTPUT PAGE WIDTHS: you will, no doubt, find it necessary to allow for different page widths (line lengths) on different output devices, etc. TVT-I & II have 32 characters per line; and have no need of carriage returns if you want to continue on the next line, after storing the last character on the previous line. Note, however, that carriage returns on TVT-I (I don't know about TVT-II--haven't studied the RE schematics in detail) do not blank the characters they skip over (in the original version, anyway).

It will be highly desirable not to split words/numbers between lines, therefore it is necessary to more than just have Tiny BASIC call a user-defined subroutine to output characters. The user may also want to output to more than one device in the same session--further complicating the problem of different line lengths. I suggest you have 2 routines:

a) One that is given a string of characters to be outputted without splitting between lines. (Say, with beginning address in HL and end address in DE, or length? or 1 register?) This routine would then take appropriate action depending on whether this additional segment will fit on the current line, by making use of access to the *current* line length accessible to it--but not to the program that called it.

b) Another--user-defined--subroutine that handles the actual output characters, which is separate from the user-defined line length parameter. (I have implemented a scheme very similar to this on the IBM 360 and the-then RCA Spectra 70--which have the same user = non-privileged instructions, but the I/Omacros are quite different--in which the same program could be used in either batch or time sharing mode as well as accommodate a variety of page widths on printers and terminals.)

3. INFIX (ALGEBRAIC) EXPRESSION INTERPRETATION: If you want, I can supply information on an algorithm that uses stacks for result numbers and saving binary operators that have to be delayed one operand/expression before execution--without having to scan the algebraic expression more than once.

While I find your Tiny BASIC project intriguing, I am not interested enough to spend the money to subscribe to yet another journal. Pop, (Victor W. Amoth) doesn't seem to think computer hobbyists need high level languages, even though his programming experience is almost entirely confined to BASIC on GE time sharing--he's still very "green" at programming in machine language on the Mark-8.

My expertise runs the full range from hardware through software to continued fraction series for transcendental functions. I'm interested in further developing the "asynchronous I/O ports" I implemented. They make hardware automatically take care of "waiting," etc., and make possible my 180 cps TVT-I

[please continue on page 31]

PROGRAM REPOSITORY & TAPE DUPLICATION FACILITY

A PUBLIC DOMAIN ALTERNATIVE TO MANUFACTURERS' USER GROUPS

The Community Computer Center (CCC) will act as a repository for program tapes; both source tapes and binary tapes. Everyone wishing to contribute programs to the public domain may do so by forwarding appropriate paper tapes to CCC. In particular, if you are hesitant about submitting a program for publication in *Dr. Dobb's Journal* because you don't want to hassle with its distribution, you are encouraged to forward the tapes to CCC and the documentation to the *Journal* for publication.

The CCC will thus serve as a desirable alternative and supplement to the User Groups that are controlled and operated by many of the processor manufacturers, some of whom charge up to $100 for "membership" and access to the programs that their *customers* developed and offered to the User Group, without compensation.

There is *no* membership fee for access to the tapes from the Community Computer Center. Instead, one pays only for the duplication and mailing costs:

> Duplication charge: $1/ounce or fraction thereof, for tapes (weighed after punching on fanfold tape)
> (Add 6% tax for orders mailed to a California address)
> Postage & handling: $0.50 on orders of $5 and less
> $1 on orders exceeding $5
> Payment must accompany all orders. Orders will be mailed First Class, within 3 days of receipt.

Lists of available tapes will be published, periodically, in *Dr. Dobb's Journal*, as well as being available from CCC:

Community Computer Center
1919 Menalto Avenue
Menlo Park, CA 94025
(415)326-4444

The following source tapes are currently available. They are programs written for the version of BASIC that is implemented for the HP 2000F minicomputers, and are discussed in *What To Do After You Hit Return* (available from the PCC Bookstore, $6.95).

Number Guessing Games	$12
Number	2
Abase	3
Trap	2
Stars	2
Clocks	3
Bagels	2
Quadgt	3
Button	2
Word Games	$10
Letter	2
Abagel	3
Hangmn	3
Madlib	6
Word	2
"Nimlike" Games	$11
23Mtch	2
Batnum	3
Nim	4
Chomp	3
Zot	5
Hide-n-Seek in 2D	$ 4
Hurkle	2
Mugwmp	2
Snark	2
Pattern Games	$11
Dangle	2
Sunsgn	3
Biosin	3
Mandal	3
Life	3
Amaze	3
Board Games	$11
Qubic5	5
Gomoku	4
Teaser	3
Rover	5
Welcome to the Caves	$ 9
Caves1	5
Wumpus	4
Caves2	5
Business & Social Science	$22
Hamrbi	3
King	5
Civil2	7
Market	5
Stock	5
Policy	4
Polut	4
Science Fiction Games	$12
Trader	10
Sttr1	9
Last Chapter	$10
Crash	4
Lunar	3
Revers	2
Zeros	3
Taxman	3
The following games are in Dartmouth BASIC	
Motie	5
Rescue	5

For historical reasons,
CCC maintains a different price schedule for postage and handling on this particular set of tapes:
duplication charge and tax, as above
postage and handling:
$0.50 on orders under $10
$1.00 on order of $10 or more

SIGNETICS 2650 KIT FOR UNDER $200

[from Roy Blacksher, MOS Microprocessor Applications Manager, Signetics, 811 E. Arques, Sunnyvale CA 94086; (408) 739-7700]

The Signetics Adaptable Board Computer, ABC 1500, is a modular microcomputer containing a CPU, memory, I/O ports and support circuitry. It is designed to cover a broad range of applications from software development to system hardware prototyping. Cost performance trade-offs have been carefully considered to achieve maximum flexibility and allow the card to be tailored to a variety of individual requirements.

The basic configuration consists of the 2650 microprocessor, 512 bytes of read/write memory (four 2112 static RAM's), 1K bytes of 2608 ROM with PIPBUG*, two 8T31 I/O ports and buffering on data, address and control lines. A single +5 volt supply will be required to power the card and communicate with a serial 20 ma current-loop terminal.

Modifications to the basic system can be easily made to allow for various memory configurations and operating modes. Unused plated-through holes are provided for the PROM memory chips (82S115's). Other options are jumper selectable.

The ABC 1500 is sold either as a completely assembled and tested card (2650 PC1500) or in kit form (2650KT9500). The kit is priced below $200.

FEATURES

-- Expandable printed circuit card: unused area on card filled with plated-through holes on .300-inch centers for wirewrap sockets.

-- 1K bytes of PIPBUG ROM (in socket).

-- 512 bytes of RAM

-- Two latched I/O ports

-- Four non-extended I/) read/write user strobes.

-- Tri-state buffers on data, address and control lines.

-- Serial input/output port.

-- Single +5 volt supply requirement (1.7A max.) for card and 20 ma current loop interface (±12 volt supply for RS 232 interface).

-- Simple memory and I/O port decoding with two 16-pin dips.

-- Interrupt and single step capability.

-- Simple clock configured from dual monostable multivibrator.

-- 24K memory expansion capability.

-- Directly compatible with 4K RAM card (2650PC2000) and power supply demonstration base (2650DS2000).

-- Card dimensions: 8" x 6.875" with a 100-pin connector along the 8" dimension.

*PIPBUG is a basic monitor having the following commands:

ALPHA CHARACTER INPUT	COMMAND
A	Alter memory
B	Set breakpoint
C	Clear breakpoint
D	Dump memory to papertape
G	Go to address
L	Load memory from papertape
S	See and alter registers

Note: the program is entered by resetting the card. The terminal will then respond with an asterisk (*).

PUBLIC INTEREST SATELLITE ASSOCIATION

The Public Interest Satellite Association (PISA) was formed in October, 1975, as a non-profit national organization to explore how satellite communications technology can be adapted to meet the long-distance telecommunications needs of non-profit users.

For the past fifteen years, satellites have been providing global links via television, radio, telephone, data, telex and facsimile for business, industry, and the military. Up to now, though, the technology, for a number of reasons, has been beyond the reach of public groups, despite the fact that satellites have been developed with nearly $80 billion of public funds. But recent technical breakthroughs in the field promise to greatly reduce satellite costs, and make the technology available for low-cost public use. To spearhead the public effort that will be required to turn this potential into reality, PISA has been formed.

PISA's goals are to:

1) Help non-profit groups understand the many facets of satellite technology;

2) Assist these groups in examining their long-distance communications costs, and in determining how satellites--and what kinds of satellites--may better serve their needs; and

3) Explore ways the technology can be used by them to form new networks of information exchange, and to improve their outreadh to the public-at-large.

In March, 1975, PISA received grants from the Stern Fund and the Ottinger Foundation to permit the following first steps to be taken:

1) Conduct a survey of the communications needs, uses, and costs of non-profit organizations;

2) Prepare written material informing these groups about satellites, the potential benefit they hold for the non-profit community, and what must be done to realize this potential;

3) Design one or more demonstration projects, using available NASA experimental satellites, to give non-profit groups some experience with the technology; and

4) Plan PISA's organization structure.

For additional information, write or call:
PISA
55 W 44 Street
New York NY 10036
(212) 661-2540

DON'T KEEP IT A SECRET!

Let us know what exciting new software and systems you are working on. We'll tell everyone else (if you wish). Maybe someone is also working on the same thing. You can work together and get results twice as fast. Or, may be someone else has already done it; no reason for everyone to reinvent the wheel.

Our 'Want' list

Careful, detailed comparison and contrast of the several versions of Tiny BASIC we are publishing. Systems software for the public domain, including:

– Tiny BASIC versions for the

INTEL 8008
Motorola/AMI 6800
RCA COSMAC
SIGNETICS 2650
MOS Technology 6502
Fairchild F-8

– Tiny block-structured languages for Microprocessors

PASCAL-like
ALGOL-like

– Resident structured and unstructured assemblers

Any old assemblers
PL360-like
Macro-assemblers

– Interactive Debuggers

– Graphics Software

For the TV Dazzler
For any TV interface (including schematics)

– Music software

Like Dompier's program (*DDJ*, V. 1, No. 2)
Like Wright's Alpha Numeric Music (PCC Bookstore)

– File systems for cassettes

This is a partial list. It will change before the ink drys. We welcome your suggestions for additions.

COMPUTER PROCESS FOR RAPID PRODUCTION OF MUSICAL COMPOSITIONS

[reprinted from Stanford University's March 31, 1976, *Campus Report*]

A complete cycle of music production from the composer's mind to the page the musicians play–has been developed at Stanford's Center for Research in Music and Acoustics.

Here's how it works:

Prof. Leland Smith, working at the Artificial Intelligence Laboratory on Arastradero Road, types a composition into the computer.

The computer then transmits all the necessary parts either directly to a Xerox copier or to a plotter. The latter makes a king-sized reproduction of the score which can be reduced in size mechanically.

Either copy produces an engraving-quality format from the Xerox in about 15 seconds.

The same procedure, done the old way by a music publisher, might take as long as two years, with the necessary engraving, printing, binding, and publishing. At Stanford it can take less than two weeks, including final editing.

The advantage of Smith's system is that it eliminates the need for copyists. The computer supplies all the parts for the instruments based on the master copy typed into the PDP-10 computer.

At the moment, the process is strictly for academic purposes. It allows composers like Smith to prepare works for performance or enables graduate students to prepare scores for their degree requirements.

Smith feels it is inevitable that such a system will become the standard method for the publication of music.

But Smith's work on music printing has been done without formal sponsorship–literally, on his own time.

He sometimes gets to the Lab at 4 a.m. to take advantage of the quiet and the availability of the computer.

Michael McNabb, a Stanford graduate student in music now studying in Paris, wrote an impressionistic piece called "Solstice", which was premiered by the Stanford Symphony under Prof. Mark Starr a few weeks ago.

It was prepared and produced entirely by computer, with Smith's help.

"It took longer to rehearse than it did to edit it," Smith said.

One of his own projects shows how a computer can help.

Francesco Bonporti, an obscure 18th

century Italian composer, once had the misfortune to get his work accidentally mixed up with that of the great Johann Sebastian Bach.

This came about when Bach, taken with Bonporti's ingenious "inventions" for violin and string bass, hand copied the latter's work. When someone else included four of them in Bach's collected works, they were credited to Bach's genius until researchers discovered the error.

Using the computer printing method, Smith developed and expanded Bonporti's "Inventio Septima" ("Seven Inventions,") adding a double scherzo of his own, based on Bonporti's original.

Smith published it under his own "logo," the San Andreas Press, with the credit line: "Graphic Realization by PDP-10 Computer."

The computer printed the entire score and the title page, including a "snapshot" of an oaktree against rolling Peninsula hills—the "San Andreas" monogram.

Smith has produced computer scores for Renaissance and Baroque chamber groups of ancient instruments in the original notation—square instead of round notes; or special notation for the 17th century lute.

Students in Prof. George Houle's classes in early music already are finding this handy for producing music required for their master's degrees.

The computer is coupled with a video display screen, which presents a five-line music staff on the operator's command. The notes appear in response to the proper typing on the keyboard.

These are fed into the computer which transmits them direct to the Xerox or to the "Calcomp" plotter, whichever is desired.

The plotter, about 40 inches wide, has two parallel metal arms across the width. On these, a special ink-laden pen travels sedately back and forth, placing the notes on the treble or bass staff while the drum moves up or down to accommodate the notation.

To the casual observer it looks as though a giant musical Ouija board was in action, operated by an invisible hand.

Smith foresses the day when hundreds of computer-produced scores, reduced to digital form, can be stored in the Library of Congress.

From any place in the country, he predicts, a musician could dial up the Library's computer, code the correct numerals for Dvorak's Fifth Symphony, for example, and have the full orchestral score delivered by telecopier.

The cost could be billed to his phone or be provided for by a coin-in-the-slot arrangement. The computer in the Library of Congress could assess the royalties due the composer, if necessary, and credit the amount to his account.

The Stanford computer's value as a research tool has no limits either, Smith feels.

One doctoral candidate already has started a computer-developed thesis, working on a method which could produce thematic catalogues of the works of the classical composers—a job of monumental drudgery if attacked in the traditional manner.

His project will be so comprehensive that it will be able to compare composers' themes, where and when they were used, down to the book, page, and line of the original score. It also will cite the places where the same themes have been used or adapted to other compositions.

Anything the computer does can be stored on magnetic tape for permanent instant recall, or erasure and reuse.

The Smith system could quite readily be adopted by music publishers. "It would cost them only about $130,000 to set up this system," Smith says, "but they seem to be afraid or reluctant to make the change."

Smith, 50, is a native of Oakland who was elected to Phi Beta Kappa as an undergraduate at UC-Berkeley. He also earned his master's degree in music at Berkeley, where he studied under the noted composer Roger Sessions.

He took additional postgraduate work at the Paris Conservatory under Olivier Messaien.

Smith taught at Mills and the University of Chicago before coming to Stanford in 1958. He has received many commissions for his original compositions, which include "Orpheus" for harpsichord, harp, and guitar; a string trio, and an opera, "Santa Claus," as well as "Three Pacifist Songs."

While he has been extremely busy in the last few years developing the Center's comprehensive program for editing and printing computer music, he has also found time to produce a piano trio, a "Rhapsody for Flute and Computer," "Arabesque for Small Orchestra," "Six Bagatelles for Piano," a suite for mixed trio, and two motets for mixed chorus.

Almost all of these have been performed at Stanford, the Cabrillo Music Festival, or at other universities.

An accomplished pianist, clarinetist, and bassoonist as well, he has played with the Chicago and San Francisco symphony orchestras. His papers on the computerization of music have appeared in professional journals.

IT CAN TALK...BUT CAN IT SING?

Votrax is proposing making the guts of this English language synthesizer system available in kit form for $1K. More details, next issue.

Note that the system described below is a turn-key, off-the-shelf item that has been on the market for several years.

The VOTRAX Model VS-6 is a new departure in voice response technology. This unique system combines low unit cost, unlimited vocabulary, operational simplicity and low data requirements to provide the ultimate in flexibility and cost effectiveness. The price of the VS-6 with parallel buffered interface is $3605 in single-unit quantity. Purchase prices are discounted for quantity buys starting at two units. Maximum discount is over 50%.

The VS-6 is programmed to speak based on phonetic coding principles. Each eight-bit command word selects one of 61 phonemes (sounds) and one of four levels of inflection (pitch). Utterances are "spelled" phonetically to produce all combinations of words and phrases required by the application. Since words and phrases are stored in the form of digital information in some storage medium, such as magnetic disc or solid-state memory, there is virtually no limitation as to the amount of vocabulary VOTRAX can produce. One well-known computer services company reports a vocabulary in excess of 300,000 words. The value of unlimited vocabulary is that the same low-cost VOTRAX unit can be used for any and all applications.

The use of phonetic coding in the VOTRAX VS-6 permits the production of speech at uniquely low data rates. A rule of thumb indicates that the number of phonemes per word is approximately equal to the number of letters per word. At eight data bits per phoneme command, VOTRAX can achieve continuous speech from input as low as 150 bps.

The VOTRAX VS-6 was developed to fit into a wide variety of applications and physical environments. A complete range of interface types and options makes VOTRAX compatible with virtually all computers, from the largest business mainframes to the smallest microprocessors. The small amount of data and limited controls required to drive VOTRAX permit installation at almost any point in a communications network: host computer, communications concentrator, communications multiplexor, or computer terminal. Data rates of 110 to 9600 bps also allow VOTRAX to fit in with a minimum of change to existing systems. Operating temperature and humidity specifications are such that specially conditioned environments are not required. Applications include: Computer Timesharing, Education, Handicapped Aids, Instrumentation, Manufacturing, Military and Training Simulators.

Electrical

Input Power Requirements	115 VAC ±10%, 47-420 Hz, 0.25 Amps
Input Power Fuse	3AG - 1/2 Amp, 125 Volts
Audio Output	100-5000 Hz, 6 Volts Peak, Nominal
Audio Output Drive Capability	0.5 Watts into an 8 Ohm load

Environmental

Operating Temperature	0^o C. to 50^o C.
Storage Temperature	-20^o C. to 70^o C.
Operating Humidity	0 to 95% with no condensation

Command Word

6-bits: 64 selections available, includes phonemes, pauses and control functions

2-bits: 4 levels of inflection available

If you are interested in having this available in $1K kits, write to:

John McDaniel
Vocal Interface Div.
4340 Campus Dr.
Suite 212
Newport Beach CA 92660
(714) 557-9181

TOUCHLESS SENSING FOR UNDER $100

We just spoke with a representative for a manufacturer of low-cost proximity sensors (about $95@ in groups of 50; $133@ in single units), and turned him on to the hobbyist movement. These sensors are capable of determining the presence or absence of materials some distance away. They can "see" water flowing from a pipe or through a semi-transparent tube, doors that are opened or closed, people, hands, fingers, spokes of a rotating wheel, etc. Their range is from at least 24 inches for sensing highly reflective material, or 40 inches for minimally reflective material, up to about 30 feet when a reflector is used beyond the material "under surveillance." They can even "see" through materials that we normally think of as being opaque (e.g., cardboard, skin, thin wood panels, etc.) much like you can see the glow of a flashlight that you have stuck in your mouth--for some obscure reason--through your cheeks.

We will carry much more extensive information on this within the next several issues. In the meantime, if you are interested in such devices being made available through distributors, mail-order hobbyist sales, and computer stores, write to the manufacturer and tell them so. You might also tell them the maximum that you would be willing to pay for such sensors. Please do not ask them for literature, schematics, etc., however, unless you are planning on purchasing them in quantity. We will be furnishing such information in forthcoming issues; the company is not set up to deal with very small retail sales . . . and we want them to be happy with the hobbyist community . . . and eager to enter our marketplace. We do *not* want them to avoid the hobbyist market because they feel they can't deal with the end users.

Just let 'em know you are very interested in their making the products available at the lowest possible price, to the hobby community, via the already-existent retail distributors (and, of course, group buys can be set up at any time).

Send your quick statements of interest to: Anthony Lazzara, President, Scientific Technology, Inc., 1201 San Antonio Rd, Mountain View CA 94043.

DESCRIPTION

The STI Model AL3093 is a self-contained, complete, sensitive non-contact proximity and retro-reflective sensor system component. All circuitry is totally sealed in the shockproof 4.4 cm (1¾") by 10.1 cm (4.06") long aluminum housing.

The AL3093 responds to any surface or object entering its field of view, irrespective of material. It also detects certain changes of color or texture.

Range of the AL3093 is up to 102 cm (40 in) as a proximity sensor. When used with a retro-reflective target, range is up to 9.8 m (30 ft). Long range units are available that "see" clear plate glass or 3 mil clear mylar or liquid surfaces at more than 102 cm (40 in).

The AL3093 can be mounted anywhere, indoors or out, submerged or in a vacuum. Interference from ambient light, environmental contaminants and thin film accumulations of dust, oil, etc., is virtually impossible in normal operation. A form of automatic gain control (AGC) maintains the modulated beam sensitivity under changing operating conditions.

SPECIAL FEATURES

- Responsive to virtually all objects and materials, many color and texture changes.
- Simple to set up with adjustable, wide sensitivity range—visible alignment indicator—no focusing.
- Range to 102 cm (40") in proximity mode, to 9.8 m (30') as a retro-reflective control.
- Long, maintenance-free life—solid state throughout, never a bulb to change. Circuit protected output.
- Operates anywhere—rugged, sealed unit is completely self-contained.
- Invisible modulated beam unaffected by ambient light, even bright sun.
- Automatic compensation for fog, dust and other atmospheric or ambient conditions.
- Versatile system component—available in custom O.E.M. configurations.

OPERATION

The STI Model AL3093 is simple to set up and operate, requiring neither focusing nor critical adjustment. A visible LED indicator glows brightly when the sensor is aligned on target and permits visual monitoring during operation. A potentiometer provides range and target sensitivity adjustment.

Maintenance requirements are practically non-existent. There are no lamps or other components that deteriorate rapidly or periodically in the all solid state circuitry. Service life is conservatively rated at 10 years.

Any number of sensors may be interconnected for simultaneous or sequential operation. Outputs can be ANDed, ORed, or arranged in any other logic sequence.

USES

The STI AL3093 is useful for every type of non-contact sensing application within its wide range capabilities. Major uses include sensing, counting, routing, positioning, inspecting, measuring, code reading, web monitoring and performing a wide variety of other automated process control functions. Additional applications include safety controls, perimeter or intrusion protection or alarms and many, many others where visible movements or changes must be sensed automatically.

A series of externally mounted relay and switch outputs, including delays, latches, and other control circuits are available for use with the AL3093. Externally mounted transformers for any input voltage are also optional.

SPECIFICATIONS

ELECTRICAL & PERFORMANCE SPECIFICATIONS

Sensing Range—screwdriver adjustable
Maximum Range
Proximity Mode*
40 in. (102 cm) (90% reflectance surface)
24 in. (61 cm) (18% reflectance surface)
Retro-reflective Mode
30 ft. (9.8 m)

*Color and texture affect range in Proximity Mode. Measurements made with Kodak standard (visible) reflectance test cards.

Input Power

Normally 12 VAC or VDC, or 24 VDC at 200 mA. Externally mounted transformers available for other input voltages.

Operating Temperature Range

−50°C to +70°C (−60°F to +160°F)

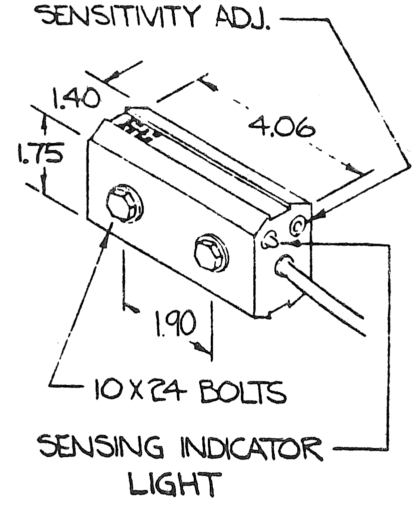
Control Options

Time delays, one shots, alarm latches and other modular control options are available for remote, external mounting.

Output

+10 VDC active pulldown—will sink 100 ma (current shutdown protection approximately 200 mA) or source 1 mA. Output may be pulled up to higher voltages, e.g. 12 VDC for MOS-type logic, without damage.

MECHANICAL SPECIFICATIONS



Response Time

Turn-On 0.0005 sec.; Turn-Off 0.01 sec.; Counting speed 6,000 CPM; Normal Cycle Life 10 billion.

Cabling

Standard 1.5 m (5 feet) 5-conductor for input and output leads. Additional length to 150 m (500 feet) and flexible armored conduit available.

Circuitry

Totally solid state, encapsulated. Withstands shock of 100 g @ 10 milliseconds.

Multiple Sensor Options

Any number of units may be ANDed or ORed through external logic circuitry. Specify requirements.

Parser saves pain

Harvey E. Hahn
630 N. Lincoln Ave., Apt 208
Addison IL 60101

February 24, 1976

In reading *PCC* [article, below] I was intrigued by your parsing subroutine, which avoids the direct input of the user (which can prevent game players, etc., from initiating control commands to BASIC itself). This would appear to be very useful in situations where inadvertent input commands could upset or destroy a program, particularly by someone who is not conversant with programming or computer languages. It would appear to be a useful "safety" feature to incorporate in BASIC interpreters.

yet another BASIC BOMBOUT!
or
How we learned to live with the INPUT statement

[reprinted from *PCC*, Vol. 3, No. 3 (Jan., 1975)]

Sometimes in the old days, often in the middle of a game, and usually to somebody new to computers, our terminals would say:

```
    ERROR xx IN LINE xxx
    READY
```

(By which the computer meant: "You typed the wrong thing when I asked for INPUT so I've kicked you out of the program. Out of the goodness of my heart, I've described what you did wrong (i.e. ERROR xx) and where the error happened (i.e. IN LINE xxx). To understand it, all you have to do is look at page xx in the reference manual, then look at the program listing (wherever that is or type LIST), and with your thorough knowledge of BASIC (oh, you say you don't speak the language – well, ask somebody then), you can figure out where you went wrong. Naturally, the READY means you're in BASIC so if you type some "random" number (like the input you tried to type in the first place), you might wipe out a line in the program and then . . .
To pick up again where you left off, type GOTO xxx – by the way, I zero all variables so you can't really start where you left off so you may as well start over. Be more careful next time!!")

Games encourage non-standard responses – like, I THOUGHT YOU WERE 'IT' when the terminal is asking, WHERE DO YOU THINK THE HURKLE IS HIDING?. People were being heavily discouraged from exploring and seeing *what would happen if*.

Suppose the terminals would print something like, I'M CONFUSED – I NEED 2 COORDINATES FROM 0 TO 9. Then the *computer* is the dummy – *it* doesn't understand *me*. "Watch me get the computer all confused." Quite different than feeling upset because the program has to be reloaded (on our 10 cps reader – no mass storage, alas) because a few random lines were erased. Blahh!

Our current solution happened in three stages.

1. A subroutine for all input. *Pass* the number and types (numeric or string) of inputs wanted to the subroutine. Input the entire user response into a character string and parse it. One special input was always recognized – STOP (the user could type STOP anytime to stop the game). We never bothered to tell our game players about Control C (remember, we never wanted a game player to give commands directly to BASIC). *Return* the inputs and a condition code to the calling routine; 1 = STOP, 2 = couldn't find all the inputs you wanted, 3 = o.k. The '2' would cause a "helpful" message to be sent to the player and the input would again be requested.

   The parsing of the inputted string was complicated because there was no direct way to convert from string to ascii (*ascii* is the numeric representation of a character) and numeric operations (like subtraction) could not be performed with strings. If we could compute

   ```
       T = C$ - "0",
   ```

   we'd almost be done; T would equal the digit in C$ (from 0 to 9) (you still need to check if T is from 0 to 9 to see if C$ actually is a digit). For numeric input, we used a FOR-NEXT loop variable as a pointer into an internal character string. If a match were found with the input character, the value of the FOR-loop variable was the ascii representation. (?!?).

   The problem (and the reason that step 1 was not our final solution) was that *it took a lot of time* to parse the input. People got really impatient, especially with multiple terminals running.

2. We eliminated the parsing subroutine. We tried all programs having line numbers greater than 1000 (hopefully, it would be harder to accidently erase a line since most inputs to the games were less than 1000). The player was supposed to ignore an ERROR when (and generally *when*, rather than *if*, for first-timers) it occurred and blindly type RUN.

   It was faster than before but it didn't solve much – "What does ERROR xx IN LINE xxx mean"? And a player couldn't continue where the game aborted because of the zero-all-variables insanity of our BASIC. So, . . .

3. One night, after everyone was asleep and all was quiet, it happened. Did you know that if you compute

   ```
       T = C$
   and X = INT(LOG(ABS(T))) + SGN(T)
   ```

   that X will be unique for each possible ascii character (on DEC EDU20, at least)? This gives you a *unique* index into an array where the ascii value of each character can be stored.

   So, we redid phase 1 with a streamlined, razzle-frazzle lookup that would gladden the heart of the most hardened hacker. And – our method of parsing INPUT isn't perceptibly slower to the user, even with multiple terminals, than good (or is it bad) ol' INPUT.

THE END (We hope.)

KEYBOARD LOADER FOR OCTAL CODE VIA THE TVT-2

Jack O. Coats, Jr, 213 Argonaut, No. 27, El Paso TX 79912
El Paso Computer Group

This program is being used in a modified form by the EPCG (El Paso Computer Group) for loading machine language programs that have been coded in octal. The program does no character validation so if you enter an invalid character it will be processed just like a valid character (the digits 0 to 7, and 0 to 3 in the most significant digit). This program should work without modification for an eight-level ASR-33 or similar device.

The program will be loaded, beginning in location 000 111. Once loaded, a program may be started by typing "$" as input to this keyboard loader.

The status input port is port no. 1, and the data I/O port, no. 0. In the status word, the high order (left-most) bit is the not-ready flag for the output port. It is high when the output port is busy and low when the port is ready to accept more output. The right-most bit is used for the input port status bit. It is low when the port is ready to present input, and high while there is no new data available. It is assumed that the input status bit is reset to the high state after data is input.

All input and output are done by subroutines GET and PUT, respectively. If any other I/O routines are desired, these routines must be replaced. For the GET routine, the character is returned in the accumulator. For the PUT routine, the character is passed to it in the accumulator. These routines may be called from any user routine as a subroutine as long as the conventions are observed.

These routines are not optimized for either memory or time. However, they are a starting place for those who need or desire a crude alternative to the panel switches.

```
ADDR  DATA  LABEL  SYM   OPERAND       COMMENT

000                ORG   0
000         STACK  EQU   <your choice>High memory address
000         RUN    EQU   END+1         Start of program entered
000   061   START  LXI   SP,STACK
001   377
002   000
003   041          LXI   H,RUN         Where do I store it?
004   111
005   000
006   076          MVI   A,CR          Output a carriage
007   015                                return
010   315          CALL  PUT
011   063
012   000
013   076          MVI   A,LF          Output a line feed
014   012
015   315          CALL  PUT
016   063
017   000
020   076          MVI   A,A'*'        Output an asterisk
021   052
022   315          CALL  PUT
023   063
024   000
025   257          XRA   A
026   006          MVI   B,(-3)        Get minus the
027   375                                character count
030   007   LOOP   RLC                 Rotate it left 3 bits
031   007          RLC
032   007          RLC
033   117          MOV   C,A           Store it in Reg. C
034   315          CALL  GET           Get a character
035   077
036   000
037   315          CALL  PUT           Write out the
040   063                                character
041   000
042   376          CPI   A'$'          Compare to the run
043   044                                signal character
044   312          JZ    RUN           Run the program
045   111                                entered
046   000
047   346          ANI   7             Mask out unwanted
050   007                                bits
051   201          ADD   C             Add it in to the running
052   004          INR   B             Is that all?       total
053   302          JNZ   LOOP          No: go to loop
054   030
055   000
056   167          MOV   M,A           Store it in memory
057   043          INX   H             Increment the address
060   303          JMP   GO            Go again
061   000
062   000
063   365   PUT    PUSH  PSW           Keep the chtr
064   333   P1     IN    STATUS        Get the status
065   001                  PORT
066   346          ANI   OUTMASK       Is it ready?
067   200
070   302          JNZ   P1            No; go to P1
071   064
072   000
073   361          POP   PSW           Retrieve the charctr
074   323          OUT   DATA          Write the data
075   000                  PORT
076   311          RET                 Go back
077   333   GET    IN    STATUS        Get the status
100   001                  PORT
101   346          ANI   INMASK        Is it what we want?
102   001
103   302          JNZ   GET           No; return to get
104   077
105   000
106   333          IN    DATA          Get the data
107   000                  PORT
110   311   END    RET
110                END
```

BREAKPOINT ROUTINE FOR 6502s

John Zeigler
8 Seaview Dr., Pittsburg CA 94565
(415) 894-3661

[This routine was distributed at the Homebrew Computer Club meeting, March 17, 1976. It is reprinted with the author's permission.]

This routine is entered via a software breakpoint. It is entered when the processor encounters a 00 op-code. Upon entering, the program counter is printed, followed by the active flags, accumulator, X index register, & index register, and stack pointer, terminated by a carriage return and line feed. It then waits for the user to type in a new op-code. Upon receiving that op-code, the original 00 code is replaced with the op-code that was input, the stack is returned to pre-interrupt status, and execution of the original program continues from the breakpoint.

To use this routine, it is necessary to load the interrupt vector, FFFE and FFFF, with 64 and 02, respectively, and place the 00 breakpoint op-code in the desired location. The following storage is required: 0000-0007, 0200-02E3, FFFE-FFFF. Note: This routine calls subroutines located in the TIM Monitor.

```
                    BUG PRØGRAM LISTING        VERSIØN 1


0200  85 07        NEG     STA 07      ;SAVE MØDIFIED P STATUS
0202  A9 4E                LDA #$4E    ;LØAD A WITH 'N'
0204  20 C6 72             JSR WRT     ;TYPE 'N'
0207  A5 07                LDA 07      ;RESTØRE MØDIFIED P
0209  4C 7F 02             JMP V       ;RETURN TØ PRØG. V
020C  85 07        ØVERFL  STA 07      ;SAVE MØDIFIED P
020E  A9 56                LDA #$56    ;LØAD A WITH 'V'
0210  20 C6 72             JSR WRT     ;TYPE 'V'
0213  A5 07                LDA 07      ;RESTØRE MØDIFIED P
0215  4C 82 02             JMP B       ;RETURN TØ PRØG. B
0218  85 07        BRK     STA 07      ;SAVE MØDIFIED P
021A  A9 42                LDA #$42    ;LØAD A WITH 'B'
021C  20 C6 72             JSR WRT     ;TYPE 'B'
021F  A5 07                LDA 07      ;RESTØRE MØDIFIED P
0221  4C 86 02             JMP D       ;RETURN TØ PRØGRAM D
0224  85 07        DEC     STA 07      ;SAVE MØDIFIED P
0226  A9 44                LDA #$44    ;LØAD A WITH 'D'
0228  20 C6 72             JSR WRT     ;TYPE 'D'
022B  A5 07                LDA 07      ;RESTØRE MØDIFIED P
022D  4C 89 02             JMP I       ;RETURN TØ PRØGRAM I
0230  85 07        IRQDIS  STA 07      ;SAVE MØDIFIED P
0232  A9 49                LDA #$49    ;LØAD A WITH 'I'
```

```
0234   20 C6 72            JSR WRT      ;TYPE 'I'
0237   A5 07               LDA 07       ;RESTØRE MØDIFIED P
0239   4C 8C 02            JMP Z        ;RETURN TØ PRØGRAM Z
023C   85 07       ZERØ    STA 07       ;SAVE MØDIFIED P
023E   A9 5A               LDA #$5A     ;LØAD A WITH 'Z'
0240   20 C6 72            JSR WRT      ;TYPE 'Z'
0243   A5 07               LDA 07       ;RESTØRE MØDIFIED P
0245   4C 8F 02            JMP C        ;RETURN TØ PRØGRAM C
0248   85 07       CARRY   STA 07       ;SAVE MØDIFIED P
024A   A9 43               LDA #$43     ;LØAD A WITH 'C'
024C   20 C6 72            JSR WRT      ;TYPE 'C'
024F   A5 07               LDA 07       ;RESTØRE MØDIFIED P
0251   4C 92 02            JMP CØNT     ;RETURN TØ PRØGRAM CØNT
0254   85 00               STA 00       ;SAVE A IN 00
0256   86 01               STX 01       ;SAVE X IN 01
0258   84 02               STY 02       ;SAVE Y IN 02
025A   68                  PLA          ;PULL P ØT A
025B   85 03               STA 03       ;SAVE P IN 03
025D   68                  PLA          ;PULL PCL TØ A
025E   85 04               STA 04       ;SAVE PCL IN 04
0260   68                  PLA          ;PULL PCH TØ A
0261   85 05               STA 05       ;SAVE PCH IN 05
0263   BA                  TSX          ;MØVE S TØ X
0264   86 06               STA 06       ;SAVE S IN 06
0266   D8                  CLD          ;NØT DECIMAL MØDE
0267   20 8A 72            JSR CRLF     ;DØ A CRLF
026A   20 CF 02            JSR MØDPC    ;CØRRECT PCL & PCH
026D   A5 05               LDA 05       ;LØAD A WITH PCH
026F   20 B1 72            JSR WRØB     ;TYPE PCH IN HEX
0272   A5 04               LDA 04       ;LØAD A WITH PCL
0274   20 B1 72            JSR WRØB     ;TYPE PCL IN HEX
0277   20 77 73            JSR SPACE    ;SPACE 1 CHARACTER
027A   A5 03               LDA 03       ;LØAD A WITH P
027C   2A                  RØL A        ;RØTATE N FLAG TØ CARRY
027D   B0 81               BCS NEG      ;BRANCH IF N FLAG SET
027F   2A          V       RØL A        ;RØTATE V FLAG TØ CARRY
0280   B0 8A               BCS ØVERFL   ;BRANCH IF V FLAG SET
0282   2A          B       RØL A        ;RØTATE PAST UNUSED BIT
0283   2A                  RØL A        ;RØTATE B FLAG TØ CARRY
0284   B0 92               BCS BRK      ;BRANCH IF B FLAG SET
0286   2A          D       RØL A        ;RØTATE D FLAG TØ CARRY
0287   B0 9B               BCS DEC      ;BRANCH IF D FLAG SET
0289   2A          I       RØL A        ;RØTATE I FLAG TØ CARRY
```

```
028A   B0 A4               BCS IRQDIS  ;BRANCH IF I FLAG SET
028C   2A          Z       RØL A       ;RØTATE Z FLAG TØ CARRY
028D   B0 AD               BCS ZERØ    ;BRANCH IF Z FLAG SET
028F   2A          C       RØL A       ;RØTATE C FLAG TØ CARRY
0290   B0 B6               BCS CARRY   ;BRANCH IF C FLAG SET
0292   20 77 73    CØNT    JSR SPACE   ;SPAGE 1 CHARACTER
0295   A5 00               LDA 00      ;GET A
0297   20 B1 72            JSR WRØB    ;TYPE A
029A   20 77 73            JSR SPACE   ;SPACE 1 CHARACTER
029D   A5 01               LDA 01      ;GET X
029F   20 B1 72            JSR WRØB    ;TYPE X
02A2   20 77 73            JSR SPACE   ;SPACE 1 CHARACTER
02A5   A5 02               LDA 02      ;GET Y
02A7   20 B1 72            JSR WRØB    ;TYPE Y
02AA   20 77 73            JSR SPACE   ;TYPE SPACE
02AD   A5 06               LDA 06      ;GET S
02AF   20 B1 72            JSR WRØB    ;TYPE S
02B2   20 8A 72            JSR CRLF    ;DØ A CRLF
02B5   20 B3 73            JSR RDHEX   ;READ VALID ØPCØDE
02B8   A2 00               LDX #$00    ;PREPARE TØ LØAD ØPCØDE
02BA   81 04               STA (04,X)  ;STØRE CØRRECT ØPCØDE
02BC   A6 06               LDX 06      ;GET S
02BE   9A                  TXS         ;RESTØRE STACK PØINTER
02BF   A5 05               LDA 05      ;GET PCH
02C1   48                  PHA         ;RESTØRE PCH TØ STACK
02C2   A5 04               LDA 04      ;GET PCL
02C4   48                  PHA         ;RESTØRE PCL TØ STACK
02C5   A5 03               LDA 03      ;GET P
02C7   48                  PHA         ;RESTØRE P TØ STACK
02C8   A4 02               LDY 02      ;RESTØRE Y
02CA   A6 01               LDX 01      ;RESTØRE X
02CC   A5 00               LDA 00      ;RESTØRE A
02CE   40                  RTI         ;RETURN TØ PRØGRAM
02CF   A5 04       MØDPC   LDA 04      ;LØAD PCL IN A
02D1   F0 07               BEQ ALTER1  ;BRANCH IF PCL = 0
02D3   C6 04       ALT1    DEC 04      ;SET PCL = PCL-1
02D5   F0 08               BEQ ALTER2  ;BRANCH IF PCL = 0
02D7   C6 04       ALT2    DEC 04      ;SET PCL = PCL-2
02D9   60                  RTS         ;RETURN FRØM SUBRØUTINE
02DA   C6 05       ALTER1  DEC 05      ;SET PCH = PCH-1
02DC   4C D3 02            JMP ALT1    ;JUMP TØ ALT1
02DF   C6 05       ALTER2  DEC 05      ;SET PCH = PCH-1
02E1   4C D7 02            JMP ALT2    ;JUMP TØ ALT2
                           END
```

DENVER TINY BASIC FOR 8080s
A 2nd version that includes 1-D arrays

F.J. Greeb, 1915 S. Cape Way, Denver CO 80227, (303) 986-6651

[An earlier release of Fred's Tiny BASIC was submitted to the Denver Amateur Computer Society. This release is a considerably improved version.]

This is a version of Tiny BASIC based on the design notes which have been published in People's Computer Company newspaper, and in the *Journal*. The program is written in 8080 assembly language for a system utilizing a TV-Typewriter and a Suding-type cassette tape interface. The program requires approximately 2.75K bytes of memory, including storage space for variables.

COMMAND SET

LET
PR (print)
GOTO
GOSUB
RET (return)
IN (input)
IF
CLEAR
LIST
RUN
END
DIM
REM
CLRS
SIZE
TAPE
LOAD

DIM -- allows single-dimensioned variables (only single letter variables may be dimensioned)
REM -- remarks follow
CLRS -- clears screen on TVT
SIZE -- prints number of bytes used, and number remaining (does not include dimensioned-variable storage areas, which are above the program)
Control -- X input in response to an INPUT statement returns control to the Tiny BASIC monitor.

FEATURES AND RESTRICTIONS

Integer Arithmetic only, +/- 32767 maximum range
Single letter variables optionally followed by the numbers 1 to 6
1-dimensional variables
Only one function available RND(X); random number generator, returns a value between 0 and +32767. If X ≠ 0, initialize the routine and return a random number. If X = 0, return a random number.
Multiple statements per line allowed using a colon (:) separator.
Strings ok in print statements; string variables not allowed.
Direct mode operation (except that GOSUB and INPUT will not operate in the direct mode)
Built-in editor for creation/modification of programs
Full line erase using a ?. No single character erase.
Dump and load programs to/from cassette tape
Implied THEN in IF statements. The THEN clause may have any recognizable Tiny BASIC statements. Multiple statements following an IF THEN clause will be executed only if the relational clause is satisfied.
Single byte line numbers, 2 to 255
Zone spacing suppression on PRINT statements using a semi-colon (;)
Expressions may be input (e.g., 3 * 5/2 is a valid input)

ARITHMETIC OPERATIONS

+, -, *, / allowed. Expressions are evaluated from left to right with multiply/divide precedence unless otherwise parenthesized.

Too deeply nested parentheses is the most common cause of error number 45. The expression complexity which can be handled is a function of the program being processed. Variables and expression operands are stored in a common memory block, with variable values entered from the bottom up, and expression operands from the top down. If overlap occurs, the error message is output. If only a few variables have been referenced, a very complex expression can be handled. If the maximum allowable number of variables (120) have been referenced, arithmetic expressions must be kept very simple.

COMMAND MODE

A "greater-than" symbol (>) is output indicating that the interpreter is awaiting a command from the keyboard. Commands entered with a line number will be entered in proper numerical sequence in the program area. Commands entered without a line number will be executed immediately if possible. Errors encountered in the direct mode will be output as mmm AT O since there is no line number associated with them.

The LIST command is optionally followed by two numbers (LIST mmm nnn). If no numbers are entered, the entire file will be displayed on the TVT. If LIST mmm is entered, line mmm will be listed. If both mmm and nnn are entered, the listing will be from line number mmm to nnn, inclusive. If mmm or nnn do not exist, the first line number greater than the input numbers will be used as limits.

LIST, RUN, CLEAR, TAPE (Output a program to cassette), and LOAD (Input a program from cassette), are designed to be used primarily in the command mode. If these commands are included in a program, they will execute properly, but upon completion (with the exception of RUN, which will simply restart the program), they will return control to the monitor portion of the program (i.e., a "> " will be output as a prompt, and no further statements will be executed until a command is input).

OTHER FEATURES & A SAMPLE PROGRAM

Some other features of the system are best illustrated by the following sample program:

```
  5 GOSUB 200
 10 PR “INPUT X,Y”;
 20 IN X,Y
 22 IF X=0 GO TO 230
 23 IF Y=0 GO TO 230
 25 IF X <0 LET X=-X
 30 IF Y<0 Y=-Y
 40 IF X> =100 X=X/7 :GOTO 40
 50 IF Y> 120 Y=Y/111:GOTO 50
 60 IF X<> 0 IF Y<> 0 Z=RND(X*Y)
 65 IF Z> 100 Z=Z/8: GOTO 65
 67 C2 = 0
 70 PR
 75 PR “I MADE A NEW NUMBER”
 80 IF C> 5 GOSUB 200
 85 PR “GUESS MY NUMBER”;
 90 IN C1
 95 C2 = C2 + 1 : C = C + 1
100 IF C1 = Z GOTO 160
110 IF C1 <Z GOTO 130
120 PR C1,; “IS TOO HIGH”
125 GOTO 80
130 PR C1,; “IS TOO LOW”
140 GOTO 80
160 PR “***** THAT’S IT *****”
163 PR “YOU TOOK”,;C2,;“GUESSES”
165 PR “INPUT 1 TO TRY AGAIN”
170 IN C1
175 IF C1 = 1 GO TO 5
180 END
200 CLRS
210 LET C=0
220 RET
230 PR “YOU CAN’T USE ZERO”
235 GOSUB 200
240 GOTO 10
```

Line 20 illustrates multiple inputs. The input values must be separated by a single character (normally a comma, but this is not required), and the entire input string of numbers terminated by a carriage return. The input routine outputs question mark as a prompt to indicate it is awaiting input data. A question mark input will erase the entire line of input.

Line 30, and several others, illustrate the implied LET statement. LET X=8 and X=8 both produce the same result. Using the LET statement speeds up execution. Omitting the LET saves space in the program memory area.

Lines 40, 50, and 65 illustrate a special use of multiple statements per line. The statements following the colon will execute only if the relational operator is satisfied. Thus, each of these statements will loop on themselves until the variable value is reduced below the relational limit.

Line 60 illustrates chaining of relational statements. The final statement will be executed only if both relational operators are satisfied, which, for this program, will always be true.

Line 70 will print a carriage return. This statement will only work with a C/R terminator, and will produce a syntax error if followed by a colon for multi-statement lines.

Lines 85 and 130 illustrate zone spacing suppression. Only the semicolon is required to suppress zone spacing. Zones are eight columns wide, which is convenient for a TVT. Zone 5 then starts a new line. Leading zeros are suppressed on numerical output.

Line 200 illustrates a special feature included for the TVT. CLRS calls a clear-screen routine, to avoid overwriting old data. Scrolling would be nicer, but my TVT won’t do that.

Throughout the program, blanks may be included or omitted freely. In general, blanks may be used or omitted between variables, constants, commands, etc., to make the program more readable, or save memory space. 10X=3 works just as well as 10 X = 3 but it doesn’t look as nice. GOTO and GOSUB may also be separated by blanks if desired. Blanks do act as separators.

CONVERSION TO OTHER SYSTEMS

Conversion to other 8080 systems should be fairly straightforward. The program was assembled with a starting location of 000 003 (split octal), but could be relocated elsewhere. The only routine not contained within the program is CRLF (output a carriage return). This routine is contained in a small monitor PROM in my system, which is also the reason for the starting location not being 000 000. This location is normally loaded with a jump instruction so that the monitor PROM is entered when the system is reset. All variable storage locations are provided within the 2.75K memory allocation. The 8080 stack for subroutine calls and push/pop operations is external to the program. I use a 128 byte ram dedicated to this purpose.

The main conversion problem will occur in the I/O portions. My TVT uses hardware control of the 8080 ready line, and will operate directly with an IN or OUT instruction. If it is necessary to modify this approach, the best technique would probably be to change the IN and OUT instructions to CALL instructions, and write subroutine suitable for the particular I/O device. The IN instruction is used only in the one input subroutine (DTIN), but the OUT instruction is used in several routines (DTIN, DECA, CNVV, PRS, LIST, and ERRS).

The tape routines for the TAPE and LOAD commands are based on software timing control of a Suding-type cassette interface. They would have to be replaced if a different type of interface was used. (Note: the output to tape routine does not include the usual 5 second delay at the start; data transmission begins immediately.) The timing constants used produce a data rate of approximately 660 baud in my 8080 system operating with a 1.25 MHz clock and no memory wait states.

No change is required to utilize Teletype lingth I/O lines. The input buffer accepts a 72-character input line, and will store it in memory properly. This also allows program lines which are longer than the 32-character TVT capability to be processed properly. Program lines are terminated by a carriage return and not by any fixed length.

Another variable which may require changing is MMAX, (used in the editor portion, subroutine RPIN), which sets the maximum memory size (high portion of address only). The Tiny BASIC program to be processed is stored above the interpreter, and is limited to a maximum address of MMAX. This value is currently set to octal 040, corresponding to my 8K system.

For conversion to non-8080 systems, good luck. Conversion of the code from the listing should be faster than writing a new program, if you are familiar with 8080 assembly language.

Some is bound to ask how I get my listings since I have no hard-copy device. My assembler produces a listing on a cassette. This is then processed by another system which has a printer.

All TVT I/O is handled through subroutine calls for ease of conversion to other systems. The two 3-byte subroutines TVTI at 002 156 and TVT0 at 002 161) may be replaced by JUMPs to more complex I/O routines. If the new routines are placed at the end of the program, the value of TOPL which specifies the first available memory location must be changed. I think the only reference to this symbol is at location 000 014, where the EOF pointer is initialized. No other changes should be required to change the I/O procedures.

ERROR DETECTION

Errors detected during execution of a Tiny BASIC program will cause an output of the form mmm AT nnn, where mmm is the error number, and nnn is the line number where the error was detected. The following errors are detected by the interpreter program:

10 - Syntax error
15 - Invalid line number (<2 or > 255) detected by editor also
20 - Memory overflow (program too large)
25 - End of file detected
30 - Attempt to transfer to a non-existing line number (GOTO or GOSUB)
35 - GOSUBs nested too deep (8 maximum)
40 - Too many variables (120 maximum)
45 - A-stack/V-stack overflow. Combination of number of variables and expression complexity too great.
50 - RET with no GOSUB
55 - No closing quote on string print
60 - Relational operator error (=, <, >, <=, >=, ><, <>)
65 - Missing right parenthesis
70 - Undefined variable in expression evaluation
75 - Add/Subtract overflow
80 - Multiply overflow
85 - Attempt to divide by zero
90 - End statement detected
95 - Empty A-stack on pop operation
100 - Input line too long (72 characters + C/R maximum)
105 - Dimensioned-variable error

PLANNED MODIFICATIONS

(Things I would like to add)

More Commands
- FOR NEXT loops
- Multiple-dimensioned variables
- String variables
- Floating point arithmetic and I/O routines
- More functions
- Etc., etc.

I haven't really devoted any time to them yet. Any help, suggestions, routines, or whatever anyone cares to contribute (especially a printer) will be greatly appreciated.

[A collage to two letters from Fred; February 21st, and April 2nd]

Dear Dennis and Jim,

Excuse the lack of detailed comments in the assembly listing. I have an 8K system, and an assembler which requires 4K. Even with only the few comments, and Tab capability in the source code generation, the source code requires around 14K, which is assembled in four blocks.

There are a few misprints in the listing (they are obvious, the entire line is moved to the left), but I don't think that will cause any problems if someone wants to implement the Tiny BASIC interpreter.

I would like to implement a different-format language, structured more specifically for the small system. I haven't formalized all of the details yet, but I anticipate using the following approach: 1) Separate editor and interpreter program. This is not as convenient, but it allows a much more sophisticated text edit capability without sacrificing memory space during execution. 2) Only referenced lines (GOTO, GOSUB) numbered. Without the resident editor, line numbers are not nearly as useful. 3) Partial symbol table formation prior to execution. Numbered line addresses stored in the symbol table to reduce execution time for GOTO/GOSUB statements. 4) Scan off all blanks at load time, except in string prints, to reduce program memory requirements.

I will probably also go to an IL type of program rather than direct coding in assembly language, since I am beginning to understand it and appreciate its features after numerous readings of the *PCC* articles, and the first *Journal*.

I have been programming in assembly language and high level languages for some time, but this was my first attempt at implementing a new language for a machine. The Tiny BASIC design articles have been a tremendous help. I don't think that I would have been as far as I am now without their help.

I have a couple game programs running in my Tiny BASIC. If I figure out how to get a listing of them, I will send them along. The program that I use to generate the assembler listings will not handle programs written in (Tiny) BASIC, since the line numbers are stored in Binary rather than ASCII.

If you're interested in it, I also have a fairly sophisticated text editor program. It is a string/line-oriented editor modeled after the PDP-9 text editor. It has 28 different commands.

--Fred

YES! We would be *delighted* to publish your Text Editor. Send it along ASAP, and keep up the good work. The more everyone shares, the more everyone gains. --JCW, Jr.

```
 000 003                    *    TINY BASIC INTERFRETER
 000 003                    *    INTEGER ARITHMETIC
 000 003                    *    WITH RND FUNCTION
 000 003                    *
 000 003   061 200 347      STRT LXI    SP,STAK
 000 006   315 220 340           CALL   CLRS
 000 011   315 061 000           CALL   INIT  INITIALIZE
 000 014   041 261 013           LXI    H,TOPL
 000 017   066 001               MVI    M,1
 000 021   042 315 011           SHLD   EFPN
 000 024   257              ERNT XRA    A
 000 025   062 325 011           STA    LNUM
 000 030   036 077               MVI    E,'?'
 000 032   076 076               MVI    A,'>'
 000 034   315 151 000           CALL   DTIN+8
 000 037   041 147 013           LXI    H,IBUF
 000 042   042 323 011           SHLD   APNT
 000 045   315 231 000           CALL   NTST  TEST FOR #
 000 050   332 164 002           JC     STM  NO #, XCT
 000 053   315 352 000           CALL   RPLN  EDIT
 000 056   303 024 000           JMP    ERNT
 000 061                    *  INITIALIZATION ROUTINE
 000 061   041 357 011      INIT LXI    H,SYMT
 000 064   006 170               MVI    B,NSYM
 000 066   315 131 000           CALL   CLER
 000 071   062 341 011           STA    CHCT
 000 074   052 315 011           LHLD   EFPN
 000 077   043                   INX    H
 000 100   042 321 011           SHLD   NMLC
 000 103   041 147 013           LXI    H,ASTR
 000 106   042 327 011           SHLD   ASTK
 000 111   041 147 012           LXI    H,VSTR
 000 114   042 331 011           SHLD   VSTK
 000 117   041 346 011           LXI    H,RSTR-1
 000 122   167                   MOV    M,A
 000 123   043                   INX    H
 000 124   167                   MOV    M,A
 000 125   042 333 011           SHLD   RSTK
 000 130   311                   RET
 000 131                    *  CLER - ZERO'S MEMORY
 000 131   257              CLER XRA    A
 000 132   167                   MOV    M,A
 000 133   043                   INX    H
 000 134   005                   DCR    B
 000 135   302 132 000           JNZ    CLER+1
 000 140   311                   RET
 000 141                    *  DTIN - INPUT ROUTINE
 000 141   036 077          DTIN MVI    E,'?'
 000 143   173                   MOV    A,E
 000 144   315 161 002           CALL   TVTO
 000 147   076 040               MVI    A,' '
 000 151   315 161 002           CALL   TVTO
 000 154   041 147 013      DTN1 LXI    H,IBUF
 000 157   345                   PUSH   H
 000 160   006 112               MVI    B,IBLN
 000 162   315 131 000           CALL   CLER
 000 165   341                   POP    H
 000 166   006 110               MVI    B,IBLN-2
 000 170   315 156 002      DTN2 CALL   TVTI
 000 173   273                   CMP    E
 000 174   312 154 000           JZ     DTN1
 000 177   376 030               CPI    18H
 000 201   302 215 000           JNZ    $+9
 000 204   061 200 347           LXI    SP,STAK
 000 207   315 076 340           CALL   CRLF
 000 212   303 024 000           JMP    ERNT
 000 215   167                   MOV    M,A
 000 216   376 015               CPI    13
 000 220   310                   RZ
 000 221   005                   DCR    B
 000 222   372 303 011           JM     ILTL
 000 225   043                   INX    H
 000 226   303 170 000           JMP    DTN2
 000 231                    *  NTST - TEST INPUT FOR LINE #
 000 231   315 271 000      NTST CALL   SBLK
 000 234   315 307 000           CALL   TSTN
 000 237   330                   RC
 000 240   104                   MOV    B,H
 000 241   115                   MOV    C,L
 000 242   315 322 000           CALL   ADEC
 000 245   174                   MOV    A,H
 000 246   267                   ORA    A
 000 247   302 147 011           JNZ    ERRM
 000 252   175                   MOV    A,L
 000 253   376 002               CPI    2
 000 255   332 147 011           JC     ERRM
 000 260   062 326 011           STA    FNUM
 000 263   140                   MOV    H,B
 000 264   151                   MOV    L,C
 000 265   042 323 011           SHLD   APNT  SET APNT
 000 270   311                   RET
```

```
000 271
000 271   052 323 011      SBLK LHLD   APNT
000 274   176                   MOV    A,M
000 275   376 040               CPI    ' '
000 277   300                   RNZ
000 300   043                   INX    H
000 301   042 323 011      SBL1 SHLD   APNT
000 304   303 274 000           JMP    SBLK+3
000 307                    *  TSTN - TEST FOR NUMERIC
000 307   052 323 011      TSTN LHLD   APNT
000 312   176                   MOV    A,M
000 313   376 060          TSN1 CPI    '0'
000 315   330                   RC
000 316   376 072               CPI    '9'+1
000 320   077                   CMC
000 321   311                   RET
000 322                    *  ADEC - CONVERT ASCII NUMBER
000 322                    *              TO BINARY
000 322   041 000 000      ADEC LXI    H,0
000 325   012                   LDAX   B
000 326   315 313 000           CALL   TSN1
000 331   330                   RC
000 332   124                   MOV    D,H
000 333   135                   MOV    E,L
000 334   051                   DAD    H
000 335   051                   DAD    H
000 336   031                   DAD    D
000 337   051                   DAD    H
000 340   326 060               SUI    48
000 342   137                   MOV    E,A
000 343   026 000               MVI    D,0
000 345   031                   DAD    D
000 346   003                   INX    B
000 347   303 325 000           JMP    ADEC+3
000 352                    * RPLN - REPLACE LINE
000 352   315 115 001      RPLN CALL LNFD
000 355   302 016 001           JNZ    INSL
000 360   345                   PUSH   H
000 361   345                   PUSH   H
000 362   043                   INX    H
000 363   315 141 001           CALL   NXTL
000 366   321                   POP    D
000 367                    *  DELETE OLD LINE
000 367   176              RPL1 MOV    A,M
000 370   022                   STAX   D
000 371   023                   INX    D
000 372   043                   INX    H
000 373   376 002               CPI    2
000 375   322 367 000           JNC    RPL1
001 000   033                   DCX    D
001 001   353                   XCHG
001 002   042 315 011           SHLD   EFPN
001 005   321                   POP    D
001 006   052 323 011           LHLD   APNT
001 011   176                   MOV    A,M
001 012   376 015               CPM    13
001 014   310                   RZ
001 015   353                   XCHG
001 016                    * INSERT NEW LINE - COUNT
001 016                    * CHARACTERS IN NEW LINE
001 016   353              INSL XCHG
001 017   052 323 011           LHLD   APNT
001 022   001 001 000           LXI    B,1
001 025   176              INS1 MOV    A,M
001 026   014                   INR    C
001 027   043                   INX    H
001 030   376 015               CPI    13
001 032   302 025 001           JNZ    INS1
001 035   052 315 011           LHLD   EFPN
001 040   345                   PUSH   H
001 041   011                   DAD    B
001 042   174                   MOV    A,H
001 043   376 040               CPI    MMAX
001 045   322 154 011           JNC    ERMO
001 050   042 315 011           SHLD   EFPN  NEW EOF
001 053   301                   POP    B
001 054                    *  MOVE ALL LINES UP
001 054   012              INS2 LDAX   B
001 055   167                   MOV    M,A
001 056   170                   MOV    A,B
 222                   SUB    D
001 060   053                   DCX    H
001 061   013                   DCX    B
001 062   302 054 001           JNZ    INS2
001 065   171                   MOV    A,C
001 066   074                   INR    A
001 067   223                   SUB    E
001 070   302 054 001           JNZ    INS2
```

```
001 073                  *    INSERT NEW LINE
001 073   072 326 011          LDA   FNUM
001 076   022                  STAX  D
001 077   023                  INX   D
001 100   052 323 011          LHLD  APNT
001 103   176            INS3 MOV   A, M
001 104   022                  STAX  D
001 105   043                  INX   H
001 106   023                  INX   D
001 107   376 015              CPI   13
001 111   302 103 001          JNZ   INS3
001 114   311                  RET
001 115                  *  LNFD - LINE FINDER
001 115   041 261 013    LNFD LXI   H, TOPL
001 120   072 326 011          LDA   FNUM
001 123   107                  MOV   B, A
001 124   176            LNF1 MOV   A, M
001 125   376 002              CPI   2
001 127   330                  RC
001 130   270                  CMP   B
001 131   320                  RNC
001 132   043                  INX   H
001 133   315 141 001          CALL  NXTL
001 136   303 124 001          JMP   LNF1
001 141                  * NXTL - GET NEXT LINE START
001 141   176            NXTL MOV   A, M
001 142   043                  INX   H
001 143   376 015              CPI   13
001 145   310                  RZ
001 146   322 141 001          JNC   NXTL
001 151   053                  DCX   H
001 152   311                  RET
001 153                  *  RND - RANDOM NUMBER GEN
001 153   315 154 005    RND  CALL  ASPP
001 156   175                  MOV   A, L
001 157   264                  ORA   H
001 160   312 171 001          JZ    GEN
001 163   062 345 011          STA   LORD
001 166   042 343 011          SHLD  HORD
001 171   072 345 011    GEN  LDA   LORD
001 174   016 017              MVI   C, 15
001 176   107                  MOV   B, A
001 177   346 041              ANI   33     BITS 19 & 24
001 201   352 205 001          JPE   GEN1
001 204   067                  STC
001 205   052 343 011    GEN1 LHLD  HORD
001 210   315 256 061          CALL  HLRS
001 213   042 343 011          SHLD  HORD
001 216   170                  MOV   A, B
001 217   037                  RAR
001 220   015                  DCR   C
001 221   302 176 001          JNZ   GEN+5
001 224   062 345 011          STA   LORD
001 227   076 177              MVI   A, 7FH
001 231   244                  ANA   H
001 232   147                  MOV   H, A
001 233   315 134 005          CALL  ASPH
001 236   311                  RET
001 237                  * HLCM - HL COMPLEMENT
001 237   175            HLCM MOV   A, L
001 240   057                  CMA
001 241   157                  MOV   L, A
001 242   174                  MOV   A, H
001 243   057                  CMA
001 244   147                  MOV   H, A
001 245   043                  INX   H
001 246   311                  RET
001 247                  * HLLS - HL LEFT SHIFT
001 247   175            HLLS MOV   A, L
001 250   027                  RAL
001 251   157                  MOV   L, A
001 252   174                  MOV   A, H
001 253   027                  RAL
001 254   147                  MOV   H, A
001 255   311                  RET
001 256                  *  HLRS - HHL RIGHT SHIFT
001 256   174            HLRS MOV   A, H
001 257   037                  RAR
001 260   147                  MOV   H, A
001 261   175                  MOV   A, L
001 262   037                  RAR
001 263   157                  MOV   L, A
001 264   311                  RET
001 265                  *  BUML - BINARY MULTIPLY
001 265   345            BUML PUSH  H
001 266   041 000 000          LXI   H, 0
001 271   042 337 011          SHLD  PRD2
001 274   006 020              MVI   B, 16
001 276   052 335 011    BUM1 LHLD  PRD1
001 301   315 256 001          CALL  HLRS
001 304   042 335 011          SHLD  PRD1
001 307   052 337 011          LHLD  PRD2
```

```
001 312   322 320 001          JNC   BUM2
001 315   321                  POP   D
001 316   031                  DAD   D
001 317   325                  PUSH  D
001 320   315 256 001    BUM2 CALL  HLRS
001 323   042 337 011          SHLD  PRD2
001 326   005                  DCR   B
001 327   302 276 001          JNZ   BUM1
001 332   321                  POP   D
001 333   052 335 011          LHLD  PRD1
001 336   315 256 001          CALL  HLRS
001 341   311                  RET
001 342                  *  BUDV - BINARY DIVIDE
001 342   315 237 001    BUDV CALL  HLCM
001 345   345                  PUSH  H
001 346   006 021              MVI   B, 17
 267                  ORA   A
001 351   052 337 011    BUD1 LHLD  DVD2
001 354   315 247 001          CALL  HLLS
001 357   042 337 011          SHLD  DVD2
001 362   005                  DCR   B
001 363   312 014 002          JZ    BUD2
001 366   052 335 011          LHLD  DVD1
001 371   315 247 001          CALL  HLLS
001 374   042 335 011          SHLD  DVD1
001 377   321                  POP   D
002 000   073                  DCX   SP
002 001   073                  DCX   SP
002 002   031                  DAD   D
002 003   322 351 001          JNC   BUD1
002 006   042 335 011          SHLD  DVD1
002 011   303 351 001          JMP   BUD1
002 014   321            BUD2 POP   D
002 015   311                  RET
002 016                  *  SPNZ - SPACE TO NEXT ZONE
002 016   072 341 011    SPNZ LDA   CHCT
002 021   107                  MOV   B, A
002 022   326 010              SUI   8
002 024   312 032 002          JZ    $+3
002 027   322 022 002          JNC   SPNZ+4
002 032   117                  MOV   C, A
002 033   015                  DCR   C
002 034   076 040              MVI   A, ' '
002 036   014            SPN3 INR   C
002 037   362 051 002          JP    SPN4
002 042   315 161 002          CALL  TVTO
002 045   004                  INR   B
002 046   303 036 002          JMP   SPN3
002 051   170            SPN4 MOV   A, B
002 052   062 341 011          STA   CHCT
002 055   311                  RET
002 056                  *  VSIN - INCREMENT VSTK
002 056   315 065 002    VSIN CALL  STOV
002 061   042 331 011          SHLD  VSTK
002 064   311                  RET
002 065                  *  STOV - CHECK FOR OVERFLOW
002 065   052 327 011    STOV LHLD  ASTK
002 070   353                  XCHG
002 071   052 331 011          LHLD  VSTK
002 074   043                  INX   H
002 075   043                  INX   H
002 076   175                  MOV   A, L
002 077   223                  SUB   E
002 100   174                  MOV   A, H

   232                  SBB   D
002 102   322 214 011          JNC   STOF
002 105   311                  RET
002 106                  * TAPE INPUT ROUTINE
002 106   016 001        TPIN MVI   C, 1
002 110   021 010 000          LXI   D, 8
002 113   333 001              IN    TAPU
002 115   241                  ANA   C
002 116   302 113 002          JNZ   TPIN+5
002 121   006 300              MVI   B, 192
002 123   005                  DCR   B
002 124   302 123 002          JNZ   $-4
002 127   333 001        TPI2 IN    TAPU
002 131   241                  ANA   C
002 132   202                  ADD   D
002 133   017                  RRC
002 134   127                  MOV   D, A
002 135   006 200              MVI   B, 128
002 137   005                  DCR   B
002 140   302 137 002          JNZ   $-4
002 143   035                  DCR   E
002 144   302 127 002          JNZ   TPI2
```

```
002 147   167                  MOV   M,A
002 150   271                  CMP   C
002 151   310                  RZ
002 152   043                  INX   H
002 153   303 110 002          JMP   TPIN+2
002 156   333 000         TVTI IN    TVT
002 160   311                  RET
002 161   323 000         TVTO OUT   TVT
002 163   311                  RET
002 164                   *  END BLOCK 1
002 164                   *  STMT - STATEMENT PROCESSOR
002 164   021 226 002     STMT LXI   D,LTMS
002 167   315 076 004          CALL  TST
002 172   315 303 004     STM1 CALL  TSTV
002 175   332 067 011          JC    ERRS
002 200   021 222 002          LXI   D,EQMS
002 203   315 076 004          CALL  TST
002 206   315 226 006          CALL  EXPR
002 211   315 150 004          CALL  DONE
002 214   315 052 005          CALL  STOR
002 217   303 164 004          JMP   NXT
002 222   275             EQMS DB    '='+128
002 223   303 067 011          JMP   ERRS
002 226   114 105         LTMS DW    'LE'
002 230   324                  DB    'T'+128
002 231   021 310 002          LXI   D,GOMS
002 234   315 076 004          CALL  TST
002 237   021 256 002          LXI   D,TOMS
002 242   315 076 004          CALL  TST
002 245   315 226 006          CALL  EXPR
002 250   315 150 004          CALL  DONE
002 253   303 216 004          JMP   XFER
002 256   124             TOMS DB    'T'
002 257   317                  DB    'O'+128
002 260   021 302 002          LXI   D,SBMS
002 263   315 076 004          CALL  TST
002 266   315 226 006          CALL  EXPR
002 271   315 150 004          CALL  DONE
002 274   315 250 004          CALL  SAV
002 277   303 216 004          JMP   XFER
002 302   123 125         SBMS DW    'SU'
002 304   302                  DB    'B'+128
002 305   303 067 011          JMP   ERRS
002 310   107             GOMS DB    'G'
002 311   317                  DB    'O'+128
002 312   021 043 003          LXI   D,PRMS
002 315   315 076 004          CALL  TST
002 320   021 013 003     PRT1 LXI   D,QUMS
002 323   315 076 004          CALL  TST
002 326   315 204 005          CALL  PRS
002 331   021 345 002     PRT2 LXI   D,CMMS
002 334   315 076 004          CALL  TST
002 337   315 016 002          CALL  SPNZ
002 342   303 320 002          JMP   PRT1
002 345   254             CMMS DB    ','+128
002 346   021 375 002          LXI   D,SMMS
002 351   315 076 004          CALL  TST
002 354   052 323 011          LHLD  APNT
002 357   176                  MOV   A,M
002 360   376 015              CPI   13
002 362   312 005 003          JZ    SMM2
002 365   376 072              CPI   ':'
002 367   302 320 002          JNZ   PRT1
002 372   303 005 003          JMP   SMM2
002 375   273             SMMS DB    ';'+128
002 376   315 076 340          CALL  CRLF
003 001   257                  XRA   A
003 002   062 341 011          STA   CHCT
003 005   315 150 004     SMM2 CALL  DONE
003 010   303 164 004          JMP   NXT
003 013   242             QUMS DB    '"'+128
003 014   052 323 011          LHLD  APNT
003 017   176                  MOV   A,M
003 020   376 015              CPI   13
003 022   312 376 002          JZ    SMMS+1
003 025   376 072              CPI   ':'
003 027   312 376 002          JZ    SMMS+1
003 032   315 226 006          CALL  EXPR
003 035   315 105 005          CALL  PRNV
003 040   303 331 002          JMP   PRT2
003 043   120             PRMS DB    'P'
003 044   322                  DB    'R'+128
003 045   021 107 003          LXI   D,IFMS
003 050   315 076 004          CALL  TST
003 053   315 226 006          CALL  EXPR
003 056   315 074 006          CALL  RELP
003 061   315 226 006          CALL  EXPR
003 064   315 324 007          CALL  CMPR
003 067   322 164 002          JNC   STMT
```

```
003 072   052 323 011         IFNX LHLD  APNT
003 075   315 141 001              CALL  NXTL
003 100   053                      DCX   H
003 101   042 323 011              SHLD  APNT
003 104   303 164 004              JMP   NXT
003 107   111                 IFMS DB    'I'
003 110   306                      DB    'F'+128
003 111   021 166 003              LXI   D,INMS
003 114   315 076 004              CALL  TST
003 117   257                      XRA   A
003 120   062 341 011              STA   CHCT
003 123   315 141 000              CALL  DTIN
003 126   315 303 004         INM1 CALL  TSTV
003 131   332 067 011              JC    ERRS
003 134   315 025 006              CALL  NCOV
003 137   315 052 005              CALL  STOR
003 142   021 153 003              LXI   D,CMM1
003 145   315 076 004              CALL  TST
003 150   303 126 003              JMP   INM1
003 153   254                 CMM1 DB    ','+128
003 154   257                      XRA   A
003 155   062 341 011              STA   CHCT
003 160   315 150 004              CALL  DONE
003 163   303 164 004              JMP   NXT
003 166   111                 INMS DB    'I'
003 167   316                      DB    'N'+128
003 170   021 204 003              LXI   D,RTMS
003 173   315 076 004              CALL  TST
003 176   315 150 004              CALL  DONE
003 201   303 066 005              JMP   RSTO
003 204   122 105             RTMS DW    'RE'
003 206   324                      DB    'T'+128
003 207   021 220 003              LXI   D,ENMS
003 212   315 076 004              CALL  TST
003 215   303 271 011              JMP   ENDM
003 220   105 116             ENMS DW    'EN'
003 222   304                      DB    'D'+128
003 223   021 234 003              LXI   D,LSMS
003 226   315 076 004              CALL  TST
003 231   303 310 010              JMP   LIST
003 234   114 111 123         LSMS DW    'LIS'
003 237   324                      DB    'T'+128
003 240   021 265 003              LXI   D,RNMS
003 243   315 076 004              CALL  TST
003 246   315 061 000              CALL  INIT
003 251   041 261 013              LXI   H,TOPL
003 254   176                      MOV   A,M
003 255   376 002                  CPI   2
003 257   332 147 011              JC    ERRM
003 262   303 204 004              JMP   NXT1-4
003 265   122 125             RNMS DW    'RU'
003 267   316                      DB    'N'+128
003 270   021 301 003              LXI   D,CLMS
003 273   315 076 004              CALL  TST
003 276   303 003 000              JMP   STRT
003 301   103 114 105 101     CLMS DW    'CLEA'
003 305   322                      DB    'R'+128
003 306   021 317 003              LXI   D,TPMS
003 311   315 076 004              CALL  TST
003 314   303 114 005              JMP   TAPE
003 317   124 101 120         TPMS DW    'TAP'
003 322   305                      DB    'E'+128
003 323   021 345 003              LXI   D,LDMS
003 326   315 076 004              CALL  TST
003 331   041 261 013              LXI   H,TOPL
003 334   315 106 002              CALL  TPIN
003 337   042 315 011              SHLD  EFPN
003 342   303 024 000              JMP   ERNT
003 345   114 117 101         LDMS DW    'LOA'
003 350   304                      DB    'D'+128
003 351   021 005 004              LXI   D,DMSG
003 354   315 076 004              CALL  TST
003 357   315 303 004              CALL  TSTV
003 362   322 310 011              JNC   DMER
003 365   021 376 003              LXI   D,DMC2
003 370   315 076 004              CALL  TST
003 373   303 357 003              JMP   $-15
003 376   254                 DMC2 DB    ','+128
003 377   315 150 004              CALL  DONE
004 002   303 164 004              JMP   NXT
004 005   104 111             DMSG DW    'DI'
004 007   315                      DB    'M'+128
004 010   021 024 004              LXI   D,SZEM
004 013   315 076 004              CALL  TST
004 016   315 254 007              CALL  SZER
004 021   303 024 000              JMP   ERNT
004 024   123 111 132         SZEM DW    'SIZ'
004 027   305                      DB    'E'+128
004 030   021 041 004              LXI   D,RMKS
004 033   315 076 004              CALL  TST
004 036   303 072 003              JMP   IFNX
```

```
004 041    122 105             RMKS DW     'RE'
004 043    315                      DB     'M'+128
004 044    021 067 004              LXI    D,CLRM
004 047    315 076 004              CALL   TST
004 052    315 220 340              CALL   CLRS
004 055    257                      XRA    A
004 056    062 341 011              STA    CHCT
004 061    315 150 004              CALL   DONE
004 064    303 164 004              JMP    NXT
004 067    103 114 122         CLRM DW     'CLR'
004 072    323                      DB     'S'+128
004 073                        *  END OF STATEMENT PROCESSOR
004 073                        *  IF MORE OPERATIONS ARE ADDED
004 073                        *  INPUT TESTS HERE
004 073                        *
004 073                        *  DEFAULT IS   LET
004 073                        *
004 073    303 172 002              JMP    STM1
004 076                        *  TST ROUTINE - STRING COMP
004 076                        *  ALTERNATE RETURN IF NO MATCH
004 076    006 001             TST  MVI    B,1
004 100    052 323 011              LHLD   APNT
004 103    032                 TST1 LDAX   D
004 104    027                      RAL
004 105    322 112 004              JNC    TST2
004 110    005                      DCR    B
004 111    077                      CMC
004 112    037                 TST2 RAR
004 113    276                      CMP    M
004 114    043                      INX    H
004 115    023                      INX    D
004 116    302 132 004              JNZ    TST3
004 121    170                      MOV    A,B
004 122    267                      ORA    A
004 123    302 103 004              JNZ    TST1
004 126    315 301 000              CALL   SBL1
004 131    311                      RET
004 132                        * SET ALT.  RETURN
004 132    170                 TST3 MOV    A,B
004 133    267                      ORA    A
004 134    312 145 004              JZ     TST5
004 137    032                 TST4 LDAX   D
004 140    023                      INX    D
004 141    027                      RAL
004 142    322 137 004              JNC    TST4
004 145    353                 TST5 XCHG
004 146    321                      POP    D
004 147    351                      PCHL          ALT. RET
004 150                        *  DONE - TEST FOR C/R OR :
004 150    315 271 000         DONE CALL   SBLK
004 153    376 015                  CPI    13
004 155    310                      RZ
004 156    376 072                  CPI    ':'
004 160    310                      RZ
004 161    303 067 011              JMP    ERRS
004 164                        * NXT - SETUP FOR NEXT LINE #
004 164    052 323 011         NXT  LHLD   APNT
004 167    176                      MOV    A,M
004 170    043                      INX    H
004 171    376 072                  CPI    ':'
004 173    312 210 004              JZ     NXT1
004 176    176                      MOV    A,M
004 177    376 002                  CPI    2
004 201    332 161 011              JC     EOFR
004 204    062 325 011              STA    LNUM
004 207    043                      INX    H
004 210    315 301 000         NXT1 CALL   SBL1
004 213    303 164 002              JMP    STMT
004 216                        *  XFER - NEW LINE FOR GO
004 216    315 154 005         XFER CALL   ASPP
004 221    174                      MOV    A,H
004 222    267                      ORA    A
004 223    302 147 011              JNZ    ERRM
004 226    175                      MOV    A,L
004 227    376 002                  CPI    2
004 231    332 147 011              JC     ERRM
004 234    062 326 011         XFE1 STA    FNUM
004 237    315 115 001              CALL   LNFD
004 242    302 175 011              JNZ    ERML
004 245    303 204 004              JMP    NXT1-4
004 250                        *  SAV - SAVE RETURN LINE #
004 250    315 141 001         SAV  CALL   NXTL
004 253    332 161 011              JC     EOFR
004 256    106                      MOV    B,M
004 257    041 357 011              LXI    H,RSTR+8
004 262    353                      XCHG
004 263    052 333 011              LHLD   RSTK
004 266    175                      MOV    A,L
004 267    223                      SUB    E
004 270    174                      MOV    A,H
004 271    232                      SBB    D
004 272    322 202 011              JNC    GSER
```

```
004 275    160                      MOV    M,B
004 276    043                      INX    H
004 277    042 333 011              SHLD   RSTK
004 302    311                      RET
004 303                        *  TSTV - TEST FOR VARIABLE
004 303    016 000             TSTV MVI    C,0
004 305    052 323 011              LHLD   APNT
004 310    176                      MOV    A,M
004 311    376 101                  CPI    'A'
004 313    330                      RC
004 314    376 133                  CPI    'Z'+1
004 316    077                      CMC
004 317    330                      RC
004 320    107                      MOV    B,A
004 321    043                      INX    H
004 322    176                      MOV    A,M
004 323    376 050                  CPI    '('
004 325    302 336 004              JNZ    $+6
004 330    043                      INX    H
004 331    016 340                  MVI    C,0E0H
004 333    303 357 004              JMP    TSV1
004 336    376 061                  CPI    '1'
004 340    332 357 004              JC     TSV1
004 343    376 067                  CPI    '7'
004 345    322 357 004              JNC    TSV1
004 350    043                      INX    H
004 351    346 007                  ANI    7
004 353    017                      RRC
004 354    017                      RRC
004 355    017                      RRC
004 356    117                      MOV    C,A
004 357    315 301 000         TSV1 CALL   SBL1
004 362    076 037                  MVI    A,1FH
004 364    240                      ANA    B
004 365    261                      ORA    C
004 366    107                      MOV    B,A
004 367    016 377                  MVI    C,-1
004 371    041 356 011              LXI    H,SYMT-1
004 374    043                 TSV2 INX    H
004 375    014                      INR    C
004 376    176                      MOV    A,M
004 377    267                      ORA    A
005 000    312 017 005              JZ     TSV3
005 003    171                      MOV    A,C
005 004    376 170                  CPI    NSYM
005 006    322 207 011              JNC    SMOE
005 011    176                      MOV    A,M
005 012    270                      CMP    B
005 013    302 374 004              JNZ    TSV2
005 016    074                      INR    A
005 017    160                 TSV3 MOV    M,B
005 020    365                      PUSH   PSW
005 021    365                      PUSH   PSW
005 022    026 000                  MVI    D,0
005 024    171                      MOV    A,C
005 025    027                      RAL
005 026    137                      MOV    E,A
005 027    041 147 012              LXI    H,VSTR
005 032    031                      DAD    D
005 033    170                      MOV    A,B
005 034    326 340                  SUI    0E0H
005 036    322 141 007              JNC    TSV4
005 041    315 134 005              CALL   ASPH
005 044    361                      POP    PSW
005 045    314 056 002              CZ     VSIN
005 050    361                      POP    PSW
005 051    311                      RET
005 052                        *  STOR - STOR VAR.  VALUE
005 052    315 154 005         STOR CALL   ASPP
003 055    345                      PUSH   H
005 056    315 154 005              CALL   ASPP
005 061    321                      POP    D
005 062    163                      MOV    M,E
005 063    043                      INX    H
005 064    162                      MOV    M,D
005 065    311                      RET
005 066                        *  RSTO - NEW # FOR RETURN
005 066    052 333 011         RSTO LHLD   RSTK
005 071    053                      DCX    H
005 072    176                      MOV    A,M
005 073    267                      ORA    A
005 074    312 221 011              JZ     RNER
005 077    042 333 011              SHLD   RSTK
005 102    303 234 004              JMP    XFE1
005 105                        *  PRNV - PRINT VARIABLE
005 105    315 154 005         PRNV CALL   ASPP
005 110    315 256 005              CALL   DECA
005 113    311                      RET
005 114                        *  TAPE - OUTPUT TO TAPE
005 114    041 261 013         TAPE LXI    H,TOPL
005 117    176                      MOV    A,M
005 120    315 252 010              CALL   TAPO
```

```
005 123   376 002              CPI   2
005 125   332 024 000          JC    ERNT
005 130   043                  INX   H
005 131   303 117 005          JMP   TAPE+3
005 134                   *  END BLOCK 2
005 134                   *  ASPH - PUSH HL TO ASTK
005 134   345             ASPH PUSH  H
005 135   315 065 002          CALL  STOV
005 140   033                  DCX   D
005 141   341                  POP   H
005 142   175                  MOV   A,L
005 143   022                  STAX  D
005 144   033                  DCX   D
005 145   174                  MOV   A,H
005 146   022                  STAX  D
005 147   353                  XCHG
005 150   042 327 011          SHLD  ASTK
005 153   311                  RET
005 154                   *  ASPP - POP HL FROM ASTK
005 154   052 327 011     ASPP LHLD  ASTK
005 157   353                  XCHG
005 160   041 147 013          LXI   H,ASTR
005 163   315 237 001          CALL  HLCM
005 166   031                  DAD   D
005 167   332 276 011          JC    SUFE
005 172   353                  XCHG
005 173   126                  MOV   D,M
005 174   043                  INX   H
005 175   136                  MOV   E,M
005 176   043                  INX   H
005 177   042 327 011          SHLD  ASTK
005 202   353                  XCHG
005 203   311                  RET
005 204                   *  PRS - PRINT STRING
005 204   052 323 011     PRS  LHLD  APNT
005 207   053                  DCX   H
005 210   176                  MOV   A,M
005 211   376 042              CPI   '"'
005 213   302 207 005          JNZ   PRS+3
005 216   043                  INX   H
005 217   072 341 011          LDA   CHCT
005 222   107                  MOV   B,A
005 223   176             PRS1 MOV   A,M
005 224   043                  INX   H
005 225   376 015              CPI   13
005 227   312 226 011          JZ    CRER
005 232   376 042              CPI   '"'
005 234   312 246 005          JZ    PRS3
005 237   004                  INR   B
005 240   315 161 002          CALL  TVTO
005 243   303 223 005          JMP   PRS1
005 246   170             PRS3 MOV   A,B
005 247   062 341 011          STA   CHCT
005 252   315 301 000          CALL  SBL1
005 255   311                  RET
005 256                   *  DECA & CNVV - OUTPUT #
005 256   174             DECA MOV   A,H
005 257   267                  ORA   A
005 260   362 302 005          JP    DEC1
005 263   076 055              MVI   A,'-'
005 265   315 161 002          CALL  TVTO
005 270   072 341 011          LDA   CHCT
005 273   074                  INR   A
005 274   062 341 011          STA   CHCT
005 277   315 237 001          CALL  HLCM
005 302   001 005 000     DEC1 LXI   B,5
005 305   021 360 330          LXI   D,-10000
005 310   315 344 005          CALL  CNVV
005 313   021 030 374          LXI   D,-1000
005 316   315 344 005          CALL  CNVV
005 321   021 234 377          LXI   D,-100
005 324   315 344 005          CALL  CNVV
005 327   021 366 377          LXI   D,-10
005 332   315 344 005          CALL  CNVV
005 335   021 377 377          LXI   D,-1
005 340   315 344 005          CALL  CNVV
005 343   311                  RET
005 344   305             CNVV PUSH  B
005 345   006 057              MVI   B,'0'-1
005 347   004                  INR   B
005 350   031                  DAD   D
005 351   174                  MOV   A,H
005 352   027                  RAL
005 353   322 347 005          JNC   CNVV+3
005 356   353                  XCHG
005 357   315 237 001          CALL  HLCM
005 362   031                  DAD   D
005 363   170                  MOV   A,B
005 364   301                  POP   B
005 365   376 060              CPI   '0'
005 367   312 010 006          JZ    CNV2
```

```
005 372   015             CNV1 DCR   C
005 373   315 161 002          CALL  TVTO
005 376   072 341 011          LDA   CHCT
006 001   074                  INR   A
006 002   062 341 011          STA   CHCT
006 005   006 200              MVI   B,128
006 007   311                  RET
006 010   200             CNV2 ADD   B
006 011   362 020 006          JP    CNV3
006 014   220                  SUB   B
006 015   303 372 005          JMP   CNV1
006 020   015             CNV3 DCR   C
006 021   312 014 006          JZ    CNV3-4
006 024   311                  RET
006 025                   *  NCOV - INPUT # TO BINARY
006 025   052 323 011     NCOV LHLD  APNT
006 030   345                  PUSH  H
006 031   052 317 011          LHLD  TMP1
006 034   072 341 011          LDA   CHCT
006 037   267                  ORA   A
006 040   302 046 006          JNZ   NCO2
006 043   041 147 013          LXI   H,IBUF
006 046   315 301 000     NCO2 CALL  SBL1
006 051   315 226 006          CALL  EXPR
006 054   315 271 000          CALL  SBLK
006 057   043                  INX   H
006 060   042 317 011          SHLD  TMP1
006 063   174                  MOV   A,H
006 064   062 341 011          STA   CHCT
006 067   341                  POP   H
006 070   042 323 011          SHLD  APNT
006 073   311                  RET
006 074                   *  RELP - RELATIONAL OP TEST
006 074   021 112 006     RELP LXI   D,M0
006 077   315 076 004          CALL  TST
006 102   056 000              MVI   L,0
006 104   046 000         REL1 MVI   H,0
006 106   315 134 005          CALL  ASPH
006 111   311                  RET
006 112   275             M0   DB    '='+128
006 113   021 156 006          LXI   D,M4
006 116   315 076 004          CALL  TST
006 121   021 134 006          LXI   D,M1
006 124   315 076 004          CALL  TST
006 127   056 002              MVI   L,2
006 131   303 104 006          JMP   REL1
006 134   275             M1   DB    '='+128
006 135   021 150 006          LXI   D,M3
006 140   315 076 004          CALL  TST
006 143   056 003              MVI   L,3
006 145   303 104 006          JMP   REL1
006 150   276             M3   DB    '>'+128
006 151   056 001              MVI   L,1
006 153   303 104 006          JMP   REL1
006 156   274             M4   DB    '<'+128
006 157   021 222 006          LXI   D,M41
006 162   315 076 004          CALL  TST
006 165   021 200 006          LXI   D,M5
006 170   315 076 004          CALL  TST
006 173   056 005              MVI   L,5
006 175   303 104 006          JMP   REL1
006 200   275             M5   DB    '='+128
006 201   021 214 006          LXI   D,M6
006 204   315 076 004          CALL  TST
006 207   056 003              MVI   L,3
006 211   303 104 006          JMP   REL1
006 214   274             M6   DB    '<'+128
006 215   056 004              MVI   L,4
006 217   303 104 006          JMP   REL1
006 222   276             M41  DB    '>'+128
006 223   303 233 011          JMP   REER
006 226                   *  EXPR - EXP. EVALUATOR
006 226                   *  CAN BE CALLED RECURSIVELY
006 226   021 253 006     EXPR LXI   D,E0
006 231   315 076 004          CALL  TST
006 234   315 332 006          CALL  TERM
006 237   315 154 005          CALL  ASPP
006 242   315 237 001          CALL  HLCM
006 245   315 134 005          CALL  ASPH

   303 271 006           JMP   E1
006 253   255              E0  DB    '-'+128
006 254   021 265 006          LXI   D,E01
006 257   315 076 004          CALL  TST
006 262   303 266 006          JMP   E01+1
006 265   253             E01  DB    '+'+128
006 266   315 332 006          CALL  TERM
006 271   021 310 006     E1   LXI   D,E2
006 274   315 076 004          CALL  TST
006 277   315 332 006          CALL  TERM
006 302   315 022 010          CALL  IADD
006 305   303 271 006          JMP   E1
```

```
006 310   253                   E2   DB    '+'+128
006 311   021 330 006                LXI   D,E3
006 314   315 076 004                CALL  TST
006 317   315 332 006                CALL  TERM
006 322   315 011 010                CALL  ISUB
006 325   303 271 006                JMP   E1
006 330   255                   E3   DB    '-'+128
006 331   311                        RET
006 332                         *  TERM - TERM EVALUATOR
006 332                         *  CAN BE CALLED RECURSIVELY
006 332   315 376 006           TERM CALL  FACT
006 335   021 354 006                LXI   D,I1
006 340   315 076 004                CALL  TST
006 343   315 376 006                CALL  FACT
006 346   315 157 010                CALL  MULT
006 351   303 335 006                JMP   TERM+3
006 354   252                   I1   DB    '*'+128
006 355   021 374 006                LXI   D,I2
006 360   315 076 004                CALL  TST
006 363   315 376 006                CALL  FACT
006 366   315 070 010                CALL  DIVD
006 371   303 335 006                JMP   TERM+3
006 374   257                   I2   DB    '/'+128
006 375   311                        RET
006 376                         *  FACT - GET FACTORS
006 376   315 103 007           FACT CALL  FNTS
007 001   320                        RNC
007 002   315 303 004                CALL  TSTV
007 005   332 026 007                JC    F0
007 010   312 245 011                JZ    UDVE
007 013   315 154 005                CALL  ASPP
007 016   136                        MOV   E,M
007 017   043                        INX   H
007 020   126                        MOV   D,M
007 021   353                        XCHG
007 022   315 134 005           FAC1 CALL  ASPH
007 025   311                        RET
007 026   315 307 000           F0   CALL  TSTN
007 031   332 053 007                JC    F1
007 034   104                        MOV   B,H
007 035   115                        MOV   C,L
007 036   315 322 000                CALL  ADEC
007 041   120                        MOV   D,B
007 042   131                        MOV   E,C
007 043   353                        XCHG
007 044   315 301 000                CALL  SBL1
007 047   353                        XCHG
007 050   303 022 007                JMP   FAC1
007 053   021 077 007           F1   LXI   D,F11
007 056   315 076 004                CALL  TST   TEST FOR (
007 061   315 226 006                CALL  EXPR  RECURSIVE CALL
007 064   021 073 007                LXI   D,FE1
007 067   315 076 004                CALL  TST
007 072   311                        RET
007 073   251                   FE1  DB    ')'+128
007 074   303 240 011                JMP   RPER
007 077   250                   F11  DB    '('+128
007 100   303 067 011                JMP   ERRS
007 103                         *  FNTS - FUNCTION TEST
007 103                         *  RND ONLY FUNCTION INITIALLY
007 103   021 133 007           FNTS LXI   D,RNDM
007 106   315 076 004                CALL  TST
007 111   315 226 006                CALL  EXPR   RECURSIVE
007 114   315 153 001                CALL  RND
007 117   021 127 007                LXI   D,RFMS
  007 122   315 076 004                CALL  TST
                       ORA   A
007 126   311                        RET
007 127   251                   RPMS DB    ')'+128
007 130   303 240 011                JMP   RPER
007 133   122 116 104           RNDM FW    'RND'
007 136   250                        DB    '('+128
007 137   067                        STC
007 140   311                        RET
007 141                         * DIM SETUP & HANDLING

   345                   TSV4 PUSH  H
007 142   315 226 006                CALL  EXPR
007 145   021 156 007                LXI   D,RPTV
007 150   315 076 004                CALL  TST
007 153   303 162 007                JMP   $+4
007 156   251                   RPTV DB    ')'+128
007 157   303 240 011                JMP   RPER
007 162   315 154 005                CALL  ASPP
007 165   257                        XRA   A
007 166   264                        ORA   H
007 167   372 310 011                JM    DMER
007 172   265                        CRA   L
007 173   312 310 011                JZ    DMER
```

```
007 176   353                        XCHG
007 177   341                        POP   H
007 200   361                        POP   PSW
007 201   302 237 097                JNZ   TSV6
007 204                         * NEW VAR.
007 204   325                        PUSH  D
007 205   353                        XCHG
007 206   052 321 011                LHLD  NMLC
007 211   353                        XCHG
007 212   163                        MOV   M,E
007 213   043                        INX   H
007 214   162                        MOV   M,D
007 215   341                        POP   H
007 216   051                        DAD   H         TIMES 2
007 217   031                        DAD   D
007 220   042 321 011                SHLD  NMLC
007 223   174                        MOV   A,H
007 224   376 040                    CPI   MMAX
007 226   322 154 011                JNC   ERMO
007 231   361                        POP   PSW
007 232   315 056 002                CALL  VSIN
007 235   067                        STC
007 236   311                        RET
007 237                         * EXISTING DIM VAR.
007 237   033                   TSV6 DCX   D
007 240   353                        XCHG
007 241   051                        DAD   H
007 242   031                        DAD   D
007 243   136                        MOV   E,M
007 244   043                        INX   H
007 245   126                        MOV   D,M
007 246   353                        XCHG
007 247   315 134 005                CALL  ASPH
007 252   361                        POP   PSW
007 253   311                        RET
007 254                         * SIZE CMMD
007 254   052 315 011           SZER LHLD  EFPN
007 257   353                        XCHG
007 260   041 261 013                LXI   H,TOPL
007 263   315 237 001                CALL  HLCM
007 266   031                        DAD   D
007 267   315 256 005                CALL  DECA
007 272   076 005                    MVI   A,5
007 274   062 341 011                STA   CHCT
007 277   315 016 002                CALL  SPNZ
007 302   026 040                    MVI   D,MMAX
007 304   036 000                    MVI   E,0
007 306   052 315 011                LHLD  EFPN
007 311   315 237 001                CALL  HLCM
007 314   031                        DAD   D
007 315   315 256 005                CALL  DECA
007 320   315 076 340                CALL  CRLF
007 323   311                        RET
007 324                         *  END BLOCK 3
007 324                         *  CMPR - COMPARE 2 VALUES
007 324   315 154 005           CMPR CALL  ASPP
007 327   345                        PUSH  H
007 330   315 154 005                CALL  ASPP
007 333   353                        XCHG
007 334   341                        POP   H
007 335   325                        PUSH  D
007 336   315 134 005                CALL  ASPH
007 341   315 011 010                CALL  ISUB
007 344   315 154 005                CALL  ASPP
007 347   301                        POP   B
007 350                         * HERE WITH X-Y IN HL
007 350   174                        MOV   A,H
007 351   267                        ORA   A
007 352   302 365 007                JNZ   CMP0
007 355   265                        ORA   L
007 356   171                        MOV   A,C
007 357   312 000 010                JZ    CMP2
007 362   376 003                    CPI   3
007 364   311                        RET
007 365   171                   CMP0 MOV   A,C
007 366   362 362 007                JP    $-7
007 371   376 001                    CPI   1
007 373   330                        RC
007 374   376 004                    CPI   4
007 376   077                        CMC
007 377   311                        RET
010 000   376 000               CMP2 CPI   0
010 002   310                        RZ
010 003   376 002                    CPI   2
010 005   310                        RZ
010 006   376 005                    CPI   5
010 010   311                        RET
010 011                         *  ISUB/IADD - ADD - SUBTRACT
010 011   315 154 005           ISUB CALL  ASPP
010 014   315 237 001                CALL  HLCM
010 017   303 025 010                JMP   IADD+3
```

```
010 022   315 154 005      IADD CALL  ASPP
010 025   174                   MOV   A,H
010 026   346 200               ANI   128
010 030   037                   RAR
010 031   107                   MOV   B,A
010 032   345                   PUSH  H
010 033   315 154 005           CALL  ASPP
010 036   174                   MOV   A,H
010 037   346 200               ANI   128
010 041   200                   ADD   B
010 042   321                   POP   D
010 043   031                   DAD   D
010 044   037                   RAR
010 045   107                   MOV   B,A
010 046   174                   MOV   A,H
010 047   027                   RAL
010 050   170                   MOV   A,B
010 051   037                   RAR
010 052   376 200               CPI   128
010 054   312 252 011           JZ    AOFE
010 057   376 160               CPI   112
010 061   312 252 011           JZ    AOFE
010 064   315 134 005           CALL  ASPH
010 067   311                   RET
010 070                    * DIVD - INTEGER DIVIDE
010 070   315 154 005      DIVD CALL  ASPP
010 073   175                   MOV   A,L
010 074   264                   ORA   H
010 075   312 264 011           JZ    DZER
010 100   076 200               MVI   A,128
010 102   244                   ANA   H
010 103   107                   MOV   B,A
010 104   374 237 001           CM    HLCM
010 107   345                   PUSH  H
010 110   315 154 005           CALL  ASPP
010 113   076 200               MVI   A,128
010 115   244                   ANA   H
010 116   200                   ADD   B
010 117   062 342 011           STA   TEMP
010 122   174                   MOV   A,H
010 123   267                   ORA   A
010 124   374 237 001           CM    HLCM
010 127   042 337 011           SHLD  DVD2
010 132   041 000 000           LXI   H,0
010 135   042 335 011           SHLD  DVD1
010 140   341                   POP   H
010 141   315 342 001           CALL  BUDV
010 144   072 342 011           LDA   TEMP
010 147   267                   ORA   A
010 150   304 237 001           CNZ   HLCM
010 153   315 134 005           CALL  ASPH
010 156   311                   RET
010 157                    * MULT - INTEGER MULTIPLY
010 157   315 154 005      MULT CALL  ASPP
010 162   076 200               MVI   A,128
010 164   244                   ANA   H
010 165   107                   MOV   B,A
010 166   374 237 001           CM    HLCM
010 171   345                   PUSH  H
010 172   315 154 005           CALL  ASPP
010 175   076 200               MVI   A,128
010 177   244                   ANA   H
010 200   200                   ADD   B
010 201   062 342 011           STA   TEMP
010 204   174                   MOV   A,H
010 205   027                   RAL
010 206   334 237 001           CC    HLCM
010 211   042 335 011           SHLD  PRD1
010 214   341                   POP   H
010 215   315 265 001           CALL  BUML
010 220   174                   MOV   A,H
010 221   027                   RAL
010 222   332 257 011           JC    MOFE
010 225   353                   XCHG
010 226   052 337 011           LHLD  PRD2
010 231   175                   MOV   A,L
010 232   264                   ORA   H
010 233   302 257 011           JNZ   MOFE
010 236   353                   XCHG
010 237   072 342 011           LDA   TEMP
010 242   267                   ORA   A
010 243   304 237 001           CNZ   HLCM
010 246   315 134 005           CALL  ASPH
010 251   311                   RET
010 252                    * TAPO - TAPE OUT ROUTINE
010 252   016 011          TAPO MVI   C,9
010 254   267                   ORA   A
010 255   027                   RAL
010 256   323 001          TAP1 OUT   TAPU
010 260   006 200               MVI   B,128
```

```
010 262   005              TAP2 DCR   B
010 263   302 262 010           JNZ   TAP2
010 266   037                   RAR
010 267   015                   DCR   C
010 270   302 256 010           JNZ   TAP1
010 273   037                   RAR
010 274   067                   STC
010 275   027                   RAL
010 276   323 001               OUT   TAPU
010 300   006 377               MVI   B,255
010 302   005              TAP3 DCR   B
010 303   302 302 010           JNZ   TAP3
010 306   037                   RAR
010 307   311                   RET
010 310                    * LIST - LIST FILE ON TVT
010 310   076 001          LIST MVI   A,1
010 312   062 326 011           STA   FNUM
010 315   076 377               MVI   A,255
010 317   062 325 011           STA   LNUM
010 322   315 307 000           CALL  TSTN
010 325   332 370 010           JC    LIS1
010 330   104                   MOV   B,H
010 331   115                   MOV   C,L
010 332   315 322 000           CALL  ADEC
010 335   175                   MOV   A,L
010 336   062 326 011           STA   FNUM
010 341   062 325 011           STA   LNUM
010 344   140                   MOV   H,B
010 345   151                   MOV   L,C
010 346   315 301 000           CALL  SBL1
010 351   315 307 000           CALL  TSTN
010 354   332 370 010           JC    LIS1
010 357   104                   MOV   B,H
010 360   115                   MOV   C,L
010 361   315 322 000           CALL  ADEC
010 364   175                   MOV   A,L
010 365   062 325 011           STA   LNUM
010 370   315 220 340      LIS1 CALL  CLRS
010 373   315 115 001           CALL  LNFD
010 376   176                   MOV   A,M
010 377   376 002               CPI   2
011 001   332 024 000           JC    ERNT
011 004   345                   PUSH  H
011 005   046 000               MVI   H,0
011 007   157                   MOV   L,A
011 010   376 144               CPI   100
011 012   322 035 011           JNC   LIS2
011 015   076 040               MVI   A,' '
011 017   315 161 002           CALL  TVTO
011 022   175                   MOV   A,L
011 023   376 012               CPI   10
011 025   322 035 011           JNC   LIS2
011 030   076 040               MVI   A,' '
011 032   315 161 002           CALL  TVTO
011 035   315 256 005      LIS2 CALL  DECA
011 040   341                   POP   H
011 041   043                   INX   H
011 042   176              LIS3 MOV   A,M
011 043   315 161 002           CALL  TVTO
011 046   043                   INX   H
011 047   376 015               CPI   13
011 051   302 042 011           JNZ   LIS3
011 054   106                   MOV   B,M
011 055   072 325 011           LDA   LNUM
011 060   220                   SUB   B
011 061   322 376 010           JNC   LIS1+6
011 064   303 024 000           JMP   ERNT
011 067                    * ERRS - ERROR HANDLING
011 067   056 012          ERRS MVI   L,10
011 071   046 000          ERR1 MVI   H,0
011 073   061 200 347           LXI   SP,STAK
011 076   315 076 340           CALL  CRLF
011 101   315 256 005           CALL  DECA
011 104   076 040               MVI   A,' '
011 106   315 161 002           CALL  TVTO
011 111   076 101               MVI   A,'A'
011 113   315 161 002           CALL  TVTO
011 116   076 124               MVI   A,'T'
011 120   315 161 002           CALL  TVTO
011 123   076 040               MVI   A,' '
011 125   315 161 002           CALL  TVTO
011 130   072 325 011           LDA   LNUM
011 133   157                   MOV   L,A
011 134   046 000               MVI   H,0
011 136   315 256 005           CALL  DECA
011 141   315 076 340           CALL  CRLF
011 144   303 024 000           JMP   ERNT
```

```
011 147   056 017         ERRM MVI   L,15
011 151   303 071 011          JMP   ERR1
011 154   056 024         ERMO MVI   L,20
011 156   303 071 011          JMP   ERR1
011 161   072 325 011     EOFR LDA   LNUM
011 164   267                  ORA   A
011 165   312 024 000          JZ    ERNT
011 170   056 031              MVI   L,25
011 172   303 071 011          JMP   ERR1
011 175   056 036         ERML MVI   L,30
011 177   303 071 011          JMP   ERR1
011 202   056 043         GSER MVI   L,35
011 204   303 071 011          JMP   ERR1
011 207   056 050         SMOE MVI   L,40
011 211   303 071 011          JMP   ERR1
011 214   056 055         STOF MVI   L,45
011 216   303 071 011          JMP   ERR1
011 221   056 062         RNER MVI   L,50
011 223   303 071 011          JMP   ERR1
011 226   056 067         CRER MVI   L,55
011 230   303 071 011          JMP   ERR1
011 233   056 074         REER MVI   L,60
011 235   303 071 011          JMP   ERR1
011 240   056 101         RPER MVI   L,65
011 242   303 071 011          JMP   ERR1
011 245   056 106         UDVE MVI   L,70
011 247   303 071 011          JMP   ERR1
011 252   056 113         AOFE MVI   L,75
011 254   303 071 011          JMP   ERR1
011 257   056 120         MOFE MVI   L,80
011 261   303 071 011          JMP   ERR1
011 264   056 125         DZER MVI   L,85
011 266   303 071 011          JMP   ERR1
011 271   056 132         ENDM MVI   L,90
011 273   303 071 011          JMP   ERR1
011 276   056 137         SUFE MVI   L,95
011 300   303 071 011          JMP   ERR1
011 303   056 144         ILTL MVI   L,100
011 305   303 071 011          JMP   ERR1
011 310   056 151         DMER MVI   L,105
011 312   303 071 011          JMP   ERR1
011 315                   *VARIABLE DEFINITIONS
011 315                   SP   EQU   6
011 315                   PSW  EQU   6
011 315                   TVT  EQU   0
011 315                   TAPU EQU   1
011 315                   CRLF EQU   0E03EH
011 315                   CLRS EQU   0E090H
011 315                   STAK EQU   0E780H
011 315                   NSYM EQU   120
011 315                   MMAX EQU   20H
011 315                   IBLN EQU   74
011 315                   *
011 315                   *  STORAGE AREAS
011 315                   *
011 315                   EFPN DS    2
011 317                   TMP1 DS    2
011 321                   NMLC DS    2
011 323                   APNT DS    2
011 325                   LNUM DS    1
011 326                   FNUM DS    1
011 327                   ASTK DS    2
011 331                   VSTK DS    2
011 333                   RSTK DS    2
011 335                   PRD1 DS    2
011 337                   PRD2 DS    2
011 341                   CHCT DS    1
011 342                   TEMP DS    1
011 343                   DVD1 EQU   PRD1
011 343                   DVD2 EQU   PRD2
011 343                   HORD DS    2
011 345                   LORD DS    1
011 346                        DS    1
011 347                   RSTR DS    8
011 357                   SYMT DS    120
012 147                   VSTR DS    256
013 147                   ASTR EQU   $
013 147                   IBUF DS    74
013 261                   TOPL EQU   $
013 261                   *
013 261                   *   END OF PROGRAM
013 261                   *
```

TV DAZZLER

SOFTWARE CONTEST

Sponsored by People's Computer Company
P.O. Box 310, Menlo Park, Ca. 94025

FIRST PRIZE: $500 certificate for hardware from CROMEMCO

SECOND PRIZE: $250 certificate for hardware from CROMEMCO

OBJECT: Develop a program resulting in a new and interesting display using the Cromemco TV Dazzler. (The Dazzler is an interface that permits a home color TV set to be a graphic terminal for certain microcomputers.)

RULES:

- All entries must use the Cromemco Dazzler display and must not require more than 20K of computer memory.
- All entries will be judged by People's Computer Company on
  1 — originality
  2 — general user appeal
  3 — clarity of documentation
- Entries should include source code and object code on punched paper tape. A listing of an appropriate bootstrap loader should also be provided.
- Software should be compatible with MITS REV 1 serial I/O port convention for I/O requirements (i.e., data transfer is on port 1, bit 7 [active low] of input port 0 is used to indicate receiver ready, and bit 0 [active low] of input port 0 is used to indicate transmitter empty).

Microcomputers can be incredibly versatile. The Dazzler adds the dimension of full-color graphic display to the microcomputer.

What can you develop? — games? — business? — education? — art? — others?

SEND ALL ENTRIES TO: PEOPLE'S COMPUTER CO
P.O. Box 310
Menlo Park, Ca. 94025

ENTRIES MUST BE RECEIVED BY SEPT. 30, 1976 interface, as well as resolving some bugs in my Micro-8 Vol. 2, Issue 1, page 11 article. I made the mistake of not indicating that just because you haven't encountered these bugs in your Mark-8 in no way means they aren't in your system. In software, I'm interested in writing a "suffix" notation programmable calculator, some sort of relocatable loader, and, perhaps, some sort of pseudo-assembler.

I'm disappointed that there doesn't seem to be any place or journal that effectively supports the Mark-8. I think there is a tremendous need for national journals specializing in individual microcomputers or at least individual microprocessors--and teaching programming, solving problems, creating hardware and software for that particular machine. This would be very valuable for the individual user with that machine.

Sincerely yours,
Thomas R. Amoth

228 Fox Rd
Media PA 19063
(215) 566-1068

Dear Tom, We will try to publish everything of value that we receive concerning the Mark-8. There is a *need* for machine-specific journals, however the market isn't yet there to support them. (It costs much bucks to publish a quality periodical.) Of course, there are the manufacturer's newsletters, and user groups, but it seems to me they don't meet hobbyist needs; particularly not inexpensively. We're gonna try.

Send us your software as you get it running so we can share it with all Micro-8 owners. --JCW, Jr

APL'S APPEAL

Dear Dragons:

I have an Altair 8K system (the 8K currently on vacation in Albuquerque due to MITS' recall order).

Incidentally, my favorite language is APL, although I know more BASIC than APL. It seems to me that a limited knowledge of APL (i.e., just a few of its features) allows greater creative freedom than knowing BASIC intimately, and is somewhat easier to attain. My initial bias against APL (and what I see as your continuing bias) comes from my background--I started off on FORTRAN, so BASIC (an "extended subset of FORTRAN" as Jean Sammet might call it) seems as natural as English. And old FORTRAN hand would likely see BASIC as the ideal language for beginners. You really should look into APL, and how it can be implemented on small machines.

At least as a beginning, BASIC looks like fun, and is easily suited to small machines. Tiny BASIC looks like even more fun, since very little has been written on languages for small machines. (A friend of mine recently said, "Why bother? You can always get a few 'K' cheap." This is the worst argument I've heard in favor of inefficient programming.) Thus my interest in your journal. After all, my pie-in-the-sky 8080 APL system has to start out with a few "basic" steps.

Sincerely,
Ed Luwish

419 Simons Ave
Hackensack NJ 07601

6800 Tiny BASIC FOR $5

Dear folks at PCC & Readers of *DDJ* 2 April 1976

I have gotten a version of Tiny BASIC up and running on the 6800. It largely follows the logic and philosophy outlined in the PCC articles (saved a lot of time!), but I have enhanced it in the following ways: two-byte line numbers, LIST can specify a range, semicolon formatting on PRINT, REM added, INPUT accepts expressions, and RND and USR functions (= machine language function call) are available.

The interpreter fits into a little less than 2K bytes (may be ROM) and uses a single JMP to each of three user-supplied I/O routines (character input, character output, and break test). I did this as a commercial venture (software is my living), but I am asking only $5 for a hex tape (Motorola format) and 20 page User's Manual. Please specify RAM-based (ORG at 0100) or ROM-base (ORG at E000, I/O preset for AMI "PROTO" board). When I have more time, and if there is sufficient interest, I will publish the IL code (I made a few changes), and show how to add extra functions. How about an assembler written in Tiny?

For a copy of this TINY BASIC for the Motorola and AMI 6800, send your name, address, and $5 to:

Tom Pittman
P.O. Box 23189
San Jose, CA 95153

PS As was noted in the TB articles, there is no such thing as a free lunch. Software comes in the lunch category, but perhaps I can offer you a cheap sandwich.

Editor's notes: Tom has a good reputation around the local Homebrew crowd. We believe that he will back his product. We would be quite interested in hearing from those who purchase his Tiny BASIC; we'd like to hear their praise and their complaints (if any).

If you wish for him to publish his Intermediate Language code (IL) in the *Journal*, write him and encourage him to do so *soon*.

Tom – What do you mean by, "an assembler in Tiny?" I *hope* that you *don't* mean an assembler that is written in Tiny BASIC.

ERRATA

The author of the 6800 version of Tiny BASIC was incorrectly given, in one place in the February issue, as being Tim Pitmann. His *correct* name and address is:

Tom Pittman
Box 23189
San Jose CA 95153
(408) 578-4944

Anyone out there know anything about Arrow Microcomputer Systems in Farmingdale, NY? We'd like their address (none was given in their ad we saw), and any other tidbits you might know about them. --JCW, Jr

Directories of:

_____ Users of home computers and their equipment
_____ Computer stores and distributers
_____ Manufacturers of computer kits
_____Computer clubs
_____Sources of used equipment
_____Microprocessor and minicomputer manufacturers

Source code listings and documentation: For which microprocessors? ____________________

_____ Nearly full-sized (much less can be published)
_____Reduced as in recent issues (more difficult to read, but more info included in each issue)

What kind of software would you like to see developed and placed in the public domain?

Importance Rating Software Description

__________ ______________________________
__________ ______________________________
__________ ______________________________
__________ ______________________________
__________ ______________________________

____
____
____

Place
13-cent
stamp
here

DR DOBB'S JOURNAL OF
COMPUTER CALISTHENICS & ORTHODONTIA
PCC
BOX 310
MENLO PARK CA 94025

To use this as a self-mailer: 1. Fold it so *this* third covers the *top* third. 2. Place the proper postage, above. 3. If you are subscribing, insert your check so that it crosses a fold. 4. Staple this closed *with a single staple*, making sure that the staple pierces the check.
(Better still, stick all of this in your own envelope, and mail it to us.)

__________What else would you like to see us publish? Please use another page or ten, if you need them.__________

____
____
____
____
____
____
____

BOOKSTORE

ACTIVE FILTER COOKBOOK
Don Lancaster. 1975. 240 pp. $14.95.

ADVANCED APPLICATIONS FOR POCKET CALCULATORS
Jack Gilbert. 1975. 304 pp. $5.95.

ALPHA-NUMERIC MUSIC WITH AMPLITUDE CONTROL
Malcolm T. Wright. 1975. 23 pp. (Xeroxed). $2.

BIOFEEDBACK AND THE ARTS
Edited by David Rosenboom. 1976. 163 pp. $12.95. (Hardbound).

BASIC
Albrecht, Finkel, and Brown. 1973. 325 pp. $3.95.

BASIC BASIC
James S. Coan. 1970. 256 pp. $3.95.

BASIC PROGRAMMING
Kemeny & Kurtz. 1961, 1971. 150 pp. $6.95.

THE BUGBOOK I & II with INSTRUCTOR'S WORKBOOK
Rony, Larsen, & Braden. 1974. Two volumes plus workbook. $16.95.

THE BUGBOOK III
Rony, Larsen, & Titus. 1975. $14.95.

CALCULATOR CALCULUS
George McCarty. 1975. 254 pp. $8.75.

COMPUTER LIB/DREAM MACHINES
Theodore Nelson. 1974. 186 pp. $7.

COMPUTERS & COMPUTATION
Scientific American. 1950 - 1971. 280 pp. $6.

ELECTRONIC PROJECTS FOR MUSICIANS
Craig Anderton. 1975. 134 pp. $6.95.

FUNDAMENTALS & APPLICATIONS OF DIGITAL LOGIC CIRCUITS
Sol Libes. 1975. 192 pp. $5.98.

FUN & GAMES WITH THE COMPUTER
Edwin R. Sage. 1975. 360 pp. $5.95.

GAMES, TRICKS, & PUZZLES FOR A HAND CALCULATOR
Wallace P. Judd. 1974. 100 pp. $2.95.

GAMES WITH THE POCKET CALCULATOR
Thiagaragan & Stolovitch. 1976. 64 pp. $2.

GETTING THE MOST OUT OF YOUR ELECTRONIC CALCULATOR
William L. Hunter. 1974. 204 pp. $4.95.

HOW TO BUILD A HOUSE SIMPLY FOR 1/3 THE COST
William Zink. 1975. 107 pp. $5.50.

INTRODUCTION TO MICROCOMPUTERS
Adam Osborne & Associates, Inc. 1975. 384 pp. $7.50.

MATH WRITING & GAMES IN THE OPEN CLASSROOM
Herbert R. Kohl. 1974. 252 pp. $2.45.

MY COMPUTER LIKES ME WHEN I SPEAK IN BASIC
Bob Albrecht. 1972. 64 pp. $2.

101 BASIC COMPUTER GAMES
Edited by David Ahl. 1974. 250 pp. $7.50.

PRINCIPLES & PRACTICE OF ELECTRONIC MUSIC
Gilbert Trythall. 1973. 214 pp. $6.95.

PROBLEMS FOR COMPUTER SOLUTION
Gruenberger & Jaffray. 1965. $7.95.

PROBABILITY
D.J. Koosis. 1973. 163 pp. $2.95.

PROFESSOR GOOGOL'S MATH PRIMER
Sam Valenza Jr. 1973. 144 pp. $3.25.

PROGRAMMING PROVERBS
Henry F. Ledgard. 1975. 134 pp. $5.95.

PCC GAMES' PROGRAM LISTINGS
PCC. 1974. 31 pp. $2.

STATISTICS
D.J. Koosis. 1972. 282 pp. $3.95.

TEACH YOURSELF BASIC, VOLUMES 1 & 2
Tecnica Education Corp. 1970. 64 pp. each. $1.95 each.

THE ENERGY PRIMER
Portola Institute. 1974. 200 pp. $5.50.

TTL COOKBOOK
Don Lancaster. 1974. 328 pp. $7.95.

II CYBERNETIC FRONTIERS
Stewart Brand. 1974. 96 pp. $2.

WHOLE EARTH EPILOG
Edited by Stewart Brand. 1974. 318 pp. $4.

WHAT TO DO AFTER YOU HIT RETURN
or PCC'S FIRST BOOK OF COMPUTER GAMES
PCC. 1975. 157 pp. $6.95.

DRAGON SHIRTS
Nancy Herter. 1974. $3.50.

List title and quantity for each item you wish to order. (Orders to be shipped within California require a sales tax remittance of 6%.) For orders less than $10, add $1 for postage and handling; for orders $10 and more, add $2. Send your order, along with your check or money order, to: PCC, Box 310, Menlo Park CA 94025. Thank you.

byte swap

We are experimenting with offering a "Want Ad" section. We will continue to do it as long as we can afford it (in terms of staff time and printing costs). Note: the charge for running an ad will undoubtedly increase as our circulation (and printing costs) increases.

Please follow these instructions in submitting ads. Ads received in other than this form cannot be accepted, and will be returned to the sender.

1. *Type* the ad, with a blank space between each line, in lines no more than 50 character positions in length.
2. Include at least your name and address as part of the ad. "Blind" ads will not be accepted.
3. Compute the charge on the basis of $1 per line or partial line, per issue.
4. Forward the typed copy and a check or money order payable to "PCC," to: DDJ Byte Swap, PCC, Box 310, Menlo Park CA 94025. Do *not* send cash. Your cancelled check is your receipt. Payment *must* accompany the ad.

SAVE MY MARRIAGE! *Buy my new assembled IMSAI 8080, loaded 22 slot mother board, 8k Ram, regular price, $1835.00. Will sell to highest bidder above $1700.00. Also, IMSAI 8080 kit, still in box, large mother board, regular price $578.00. Will sell to highest bidder above $547.00. Send bids to: Eric Stewart, 664 Via Alamo, San Lorenzo CA 94580.*

DR DOBB'S JOURNAL OF
COMPUTER CALISTHENICS & ORTHODONTIA
PCC
Box 310
Menlo Park CA 94025