dr. dobb's journal of

$1.50

COMPUTER Calisthenics & Orthodontia

Running Light Without Overbyte

May, 1976 | Box 310, Menlo Park CA 94025 | Volume 1, Number 5

A REFERENCE JOURNAL FOR USERS OF HOME COMPUTERS

In This Issue. . .

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.

DR DOBB'S JOURNAL OF
COMPUTER CALISTHENICS & ORTHODONTIA

Volume 1, Number 4; April, 1976
Box 310, Menlo Park CA 94025

Publisher
People's Computer Company
1010 Doyle, Menlo Park, California
(415) 323-3111

Editor
Jim C. Warren, Jr.

Contributing Editors
F. J. Greeb
Lichen Wang

Watchdogs
Bob Albrecht
Dennis Allison

Underdog
Rosehips Malloy

Circulation & Subscriptions
Mary Jo McPhee

Bulk Sales
Dan Rosset

Printer
Nowels Publications, Menlo Park 94025

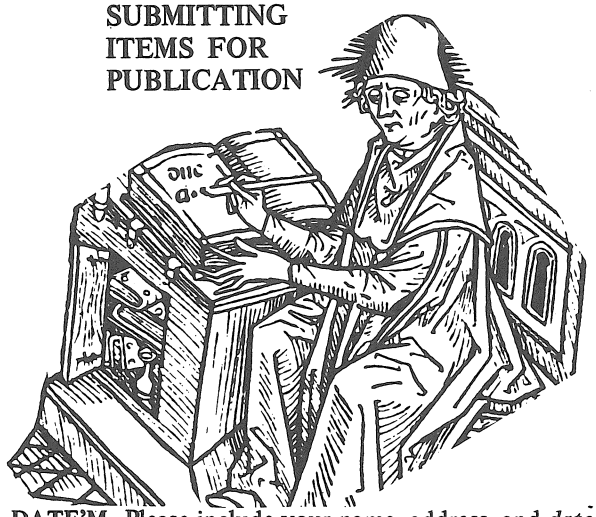
POSTMASTER: *Please send Form 3579 to: Box 310, Menlo Park CA 94025. Return postage guaranteed. Application to mail at second-class postage rates is pending at Menlo Park CA. Published 10 times per year; monthly, excluding July & December.*

U.S. subscriptions:
(Subscription blank is on page 33.)
$1.50 for a single issue.
$3 for the first three issues.
$10 per year.

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

Discounts available for bulk orders.

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, 8½ x 11 inch, white paper. If we can't read it; we can't publish it. Remember that we will be retyping 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-folded paper. Since we reduce the copy in size, submitting it on individual pages forces us to do a significant amount of extra 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 so request. We will not publish correspondence, however, which is sent to us anonymously.

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

ADVERTISING—Advertising from manufacturers and vendors may be accepted by us. However, we reserve the right to refuse any 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.

COPYRIGHT MANIA: It's mine; it's mine, and you can't play with it!

During the past year or so, People's Computer Company has received several letters-with-enclosures from one Calvin N. Mooers of Rockford Research, Inc. in Cambridge Mass. We initiated the rather unfortunate contact by asking him for information about an interesting but relatively obscure computer language that he had developed called TRAC. (Note: TRAC is, at the least, a registered trademark, and probably patented, copyrighted, and marked with infra-red dye to boot.) What we have since received from this person, however, appears to primarily be concerned with copyrights, patents, trade-marks, and the like. We don't really know because we didn't take the time to wade through all of it. He has sent us a copyrighted price list for his software and documentation that included an entire paragraph about its copyright protection and registered service mark, a mimeographed policy on copyrights and trade-marks, an article concerning a $3 million suit against some companies that purportedly have used Mooers' language, and *two* copies of a major article that Mooers wrote concerning software copyrighting (we hesitate to give the actual name and source of the article—we might be sued for reprinting the title without the author's permission). Oh yes, he also included some information about his computer language. Incidentally, he explicitly prohibited us from publishing most of his letters . . . which saved us at least several microseconds in our reaching the decision not to reprint them. Needless to say, Mooers has shown great interest in (preoccupation with?) the manner in which many hobbyists obtain their software, and has written us concerning this topic . . . but we can't let you know what he said because he prohibited us from printing or paraphrasing it.

If you are interested in the topic of proprietary software, you might look up Mooers' name in some readers' guide to computer science literature.

There is an interesting clincher to this little story: Enclosed with Mooers' most recent paper deluge concerning copyright protection were *reproduced copies of two articles from* Computerworld *newspaper.* Yes, Computerworld is copyrighted and includes an explicit prohibition against reproduction of material appearing in it unless written permission is obtained. No, the copies that Mooers included with his letter did *not* include any indication that he had obtained such permission.

[Editor's Note: We have no quarrel with copyrighting, whether it is applied to publications or to software. You may note that most of the PCC publications are copyrighted, including *Dr. Dobb's Journal*. We do object, however, to the incredible teapot tempest that has recently been raised concerning proprietary software and the hobbyist community. We also object to the blanket indictment that has been laid on *all* hobbyists: ". . . most of you steal your software" (the essence of Bill Gate's widely publicized February 3rd, open-letter to hobbyists).

1. We feel that it unjustifiably casts a shadow on the entire hobbyist community.

2. We know there are many hobbyists who are *not* thieves in spite of the fact that copyrighted software is as easy to copy as are copyrighted newspaper articles.

3. We feel this proprietary preoccupation is a waste and misuse of time and energies of talented software professionals.

4. We feel there are differences between marketing software to hobbyist/consumers for their entertainment, and marketing software to the business and industrial community where it is used directly or indirectly for financial gain. We find it unreasonable and impractical to attempt to sell software to hobbyists when its price is half the cost of their hardware systems. It is unreasonable because it's too expensive. It is impractical because, at best, software is very difficult to protect against reproduction.

Furthermore, it is naive to attempt to market software to hobbyists via a royalty agreement with a hardware manufacturer. The analogy comes to mind of someone developing an excellent and useful reference book, then attempting to market it via a royalty agreement with a manufacturer of copying machines, knowing that the manufacturer is going to place the book next to their copiers with a sign saying, "If you want a copy of this book, you must send us $350."

We feel there are only two practical choices in marketing software for the hobbyist community: 1) Charge very little for your software, and depend on volume sales for your profit; 2) Charge a great deal for your software, and sell it to your *only controllable* marketplace: the hardware manufacturers. They need it to enhance their hardware. They are also the only ones who are making sufficient profit to be able to afford your high price.

We believe that we are exemplifying these viewpoints relevant to the copyrighting of PCC publications. Please note the statement of *Reprint Privileges* inside the front page of each issue of *Dr. Dobb's Journal*. You may also note that, since we are marketing to the non-profit-making hobbyists, our subscription rate is $10/year. Compare this to, for instance, the $28/year that *Microcomputer Digest* charges for a smaller, monthly publication, or the $10 to $40 that DataPro charges for a *single copy* of some of their slender reports. But, these latter publications are being marketed to the highly profitable business and industrial computer communities. We have no particular quarrel with this. We are simply pointing out that one adjusts to the realities of one's chosen marketplace.

Finally, to those software professionals and hardware manufacturers who choose to provide low-cost software to the hobbyist community: We wish to actively encourage your efforts. As you develop such software, if you will forward information about it, we will be pleased to publicize it, without cost. We believe that you are taking the right track in this new and exciting area.] --Jim C. Warren, Jr.

THE SIX PROJECT STAGES

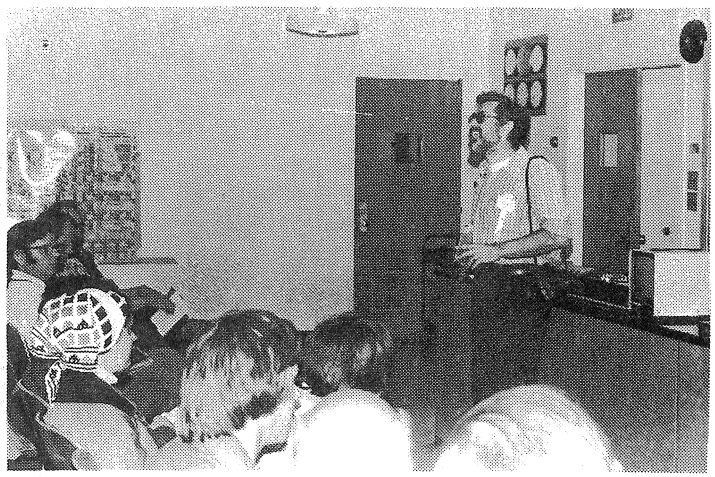
Wild Enthusiasm
Disillusionment
Total Confusion
Search for the Guilty
Punishment of the Innocent
Promotion of the Nonparticipants

THE 1976 TRENTON COMPUTER FESTIVAL

by Sol Libes

[Editor's Note: The 1976 Trenton Computer Festival was the first manufacturer-independent computer convention of national scope for hobbyists. It was held on May 2nd.]

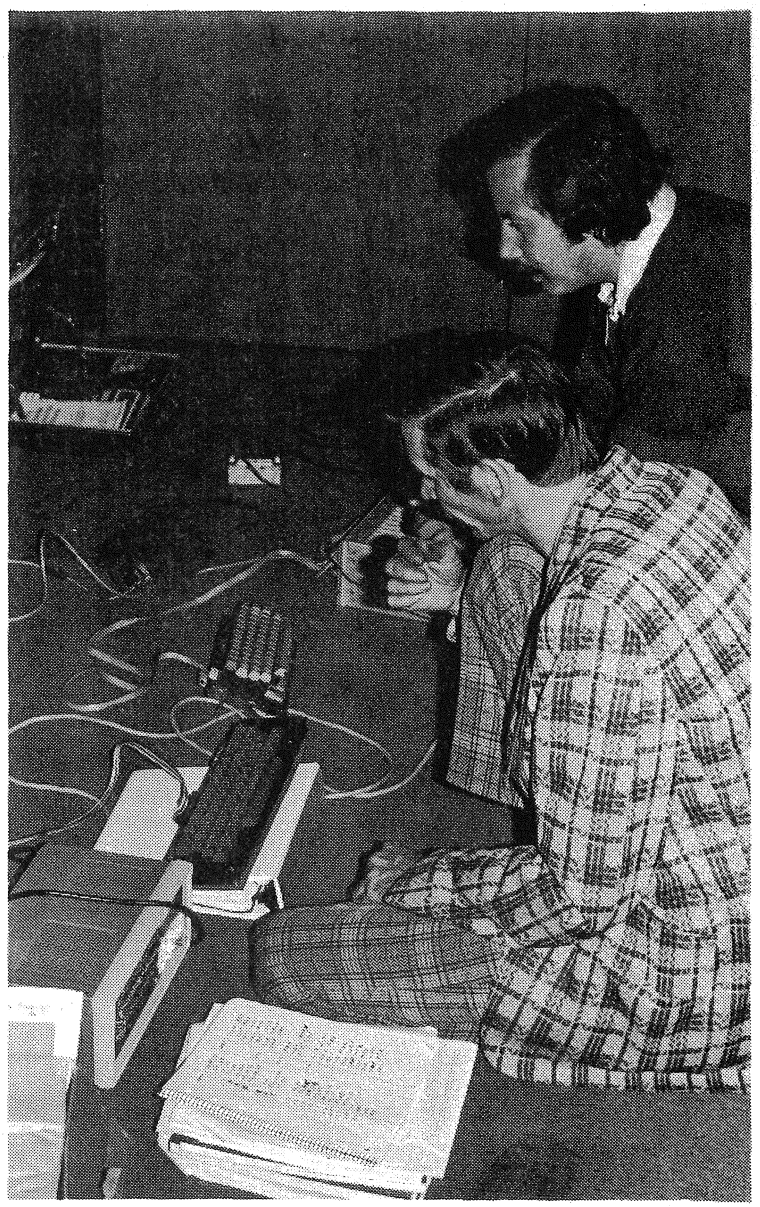
Back in November 1975 when Al Katz and I conceived of a Computer Festival, if you had said that 1,500 people would attend, I would have said you didn't know what you were talking about. And, if you had said that we would have 45 exhibitors, I would have thought that you belonged in the "cuckoo nest." But, it all happened on May 2nd at Trenton State College in New Jersey.



SOL LIBES, ACG-NJ president, spoke at the "Computer Club Congress".

What started out as a small affair for about 300 people (we only had about 100 members at the time), six to nine exhibitors, and a flea market, exploded. Considering that we did not anticipate such a large affair, it is incredible that everything went so well.

FLEA MARKET scene



PAT DEITTMAN and associate prepared for demonstration on use of microcomputers by radio amateurs.

The Festival turned into a National Convention—the first of its kind. People came from California, Colorado, Florida, North Carolina, Virginia, Washington DC, Maryland, Ohio, Pennsylvania, Massachusetts, Illinois, Indiana, New York, Connecticut, New Hampshire, Michigan, and, of course, New Jersey—18 states in all!

There were 26 speakers with such well known authorities as Dr. Robert Suding, Hal Chamberlin, and Dave Ahl. They spoke on subjects such as the Z-80 Mpu—including demo, computer music, computer graphics, writing software, 16-bit mpu's, etc.

The exhibitors included DEC, Motorola, RAC, Digital

Group, dealers for MITS, IMS, Sphere, E&L, HAL Communications, and many others.

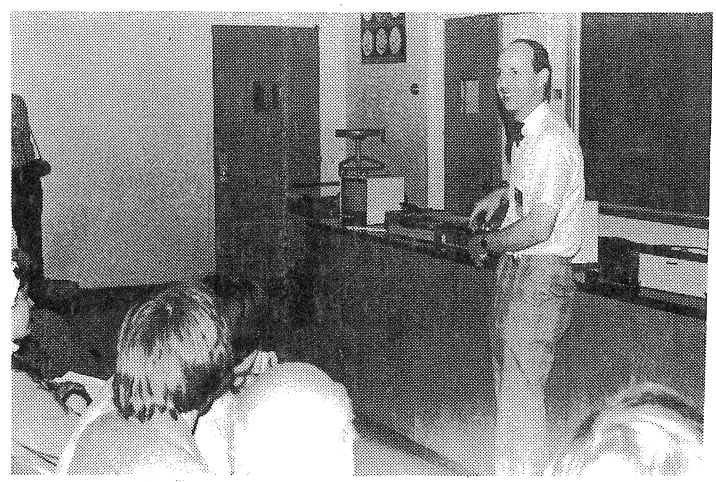
Amateurs set up demo's of their systems. There was a free program-copying service for programs in the public domain. There was a huge outdoor flea market.

One exhibitor, who had brought 14 CRT terminals (@$500 apiece) to the festival, sold 12 within minutes after the doors opened. By noon, all his stock was gone and he closed up his booth!

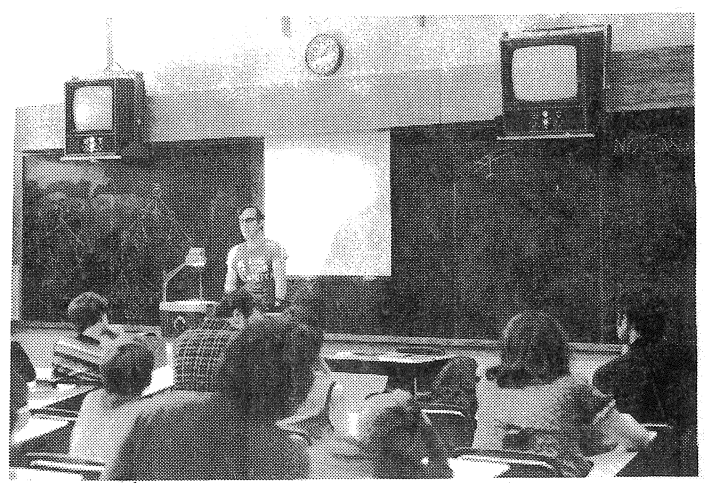
Of course, I was very busy during the Festival, but I did manage to go to two talks—and they were great. I heard Hal Chamberlin's talk on computer music. Hal is doing some radically new things using his IMP 16-bit mpu and a special Fourier hardware system to create music that is better than that I have heard from systems using other techniques. Hal played a tape of the Bach Toccata & Fugue in D Minor which was programmed on a 16-bit machine and processed through his Fourier circuitry to produce a rendition which

MORE flea market

HAL CHAMBERLIN (left), Computer Hobbyist **editor, demonstrated his new techniques for computer music.**



DR. ROBERT SUDING talked about, and demonstrated his new Z-80 microcomputer system.

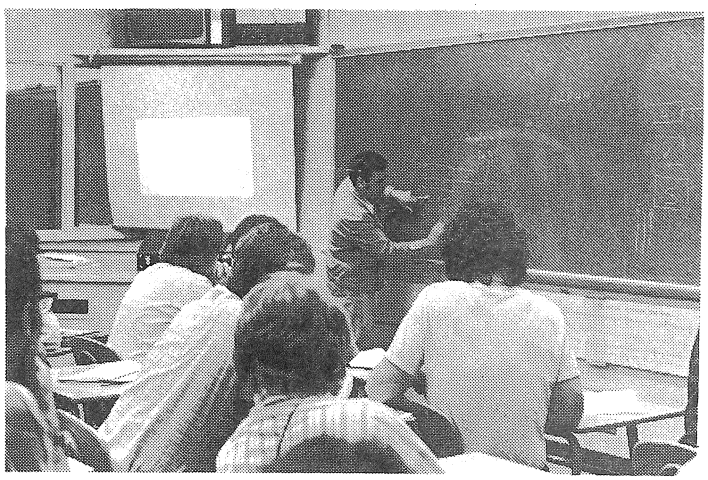


DAVE AHL (Creative Computing **editor) spoke on computer games.**

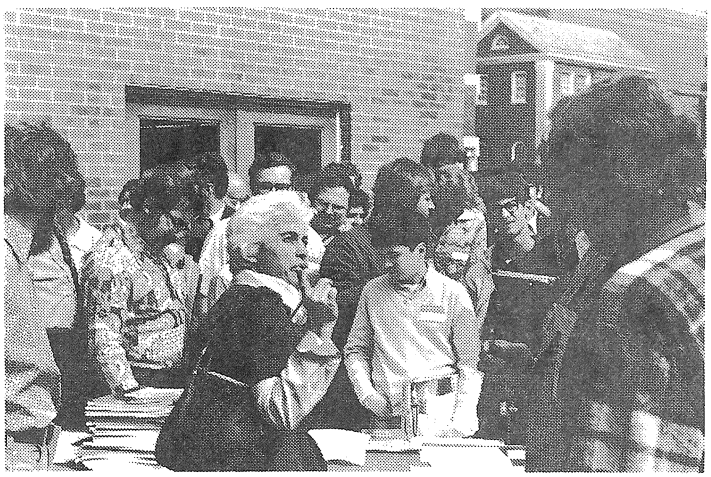
sounded like the best musician I have ever heard.

The other talk I heard was Dr. Robert Suding's concerning the new Zilog Z-80 Mpu. Bob had received an engineering sample of the Z-80 about 6 weeks ago and quickly and easily reworked a Digital Group CPU board to accept the Z-80 (it is *not* pin compatible with the 8080). Bob showed his system with a very impressive CRT display of all those Z-80 double registers.

All I can say is, if you were not there, you really missed the event of the year (there must have been about 100 Mpu-based systems up and running).



ED GERRI spoke on computer graphics.



100 door prizes—worth about $2K—were handed out at the festival.

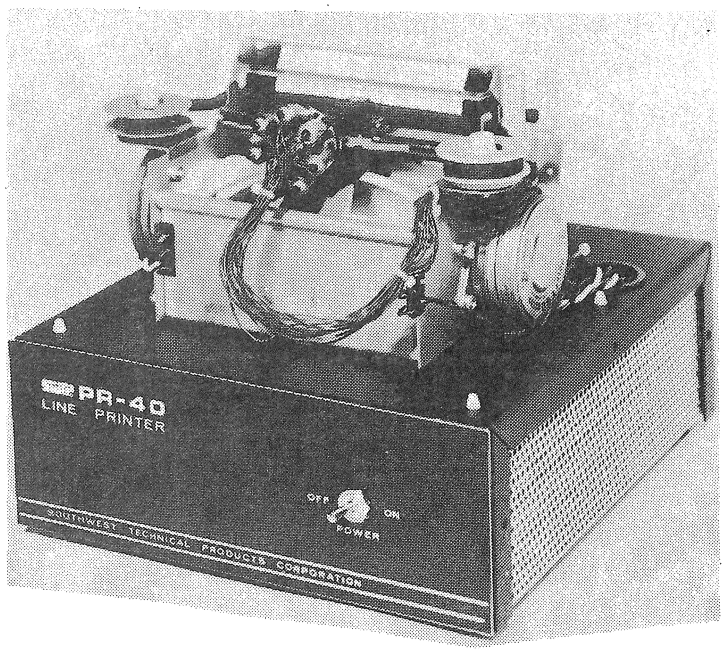
photos by Marj Kirk

40 CHARS/LINE PRINTER FOR $250

by Southwest Technical Products Corp. staff
219 W. Rhapsody, San Antonio TX 78216
(512) 344-0241

Southwest Technical's PR-40 Alphanumeric Printer Kit is a 5x7 dot matrix impact printer similar in operation to the well-known Centronics printers. It prints the 64 character upper case ASCII set with 40 characters/line at a print rate of 75 lines/minute on standard 3-7/8" wide rolls of adding machine paper. One complete line is printed at a time from an internal forty-character line buffer memory. Printing takes place either on the receipt of a carriage return or automatically whenever the line buffer memory is filled.

The printer is available in kit form only and includes the assembled print mechanism, chassis, circuit boards, components, 120/240 VAC-50/60 Hz power supply, assembly instructions, one ribbon and one roll of paper. It sells for $250 postpaid in the U.S., and delivery is 30 days.



DAY RECOMMENDS . . .

Dear Jim Warren Jr.

There are two excellent articles in May 7, 1976, *Science:*

"Microprocessors?—An End User's View", R.E. Dessy, pp 511-518.

"Microprocessor Aplications: A Less Sophisticated Approach," J.T. Arnold, pp 519-523.

Jim Day

Votrax makes the offer

speech synthesis kit for under $1K

by John McDaniel

Dear Jim: **April 26, 1976**

This letter is to confirm our conversation with regard to the VOTRAX Synthesizer Kit which we had discussed.

As I indicated in our conversation, VOTRAX was not aware of the computer hobbyist market, and therefore had not addressed itself to providing anything for that community.

As a result of preliminary investigation, I am pleased to relate to you that we could provide a VOTRAX Synthesizer in a kit form for a price not to exceed $1000. However, we require more information as to the specifications of the kit and the size of market potential, before we can commit to this price. Our quotation on a minimum order and delivery would be contingent on receiving this data.

Based on the text of your announcement in *Dr. Dobb's Journal* [Volume 1, Number 3, page 12], I believe that I have mislead you with regard to the size of the VOTRAX market. Your inference was that the VOTRAX market is not very large at present. This is not correct and, I am afraid, would tend to mislead your readers as to the impact the hobbyist market would have. This is not to say, however, that we're not interested in providing something for those users.

I hope that this information will be of benefit to you. I look forward to meeting with you again at the Home-Brew Computer Club Meeting. If I can be of further service, please don't hesitate to contact me.

Sincerely,

John H. McDaniel
Regional Sales Manager
Vocal Interface Division

4340 Campus Dr., No. 212
Newport Beach CA 92660
(714) 557-9181

BASIC SYNTHESIZER INTERFACE SPECIFICATIONS

GENERAL DESCRIPTION

The synthesizer requires 8 parallel data bits on its input pins in order to operate. Of these 8 bits, 6 are used for phoneme slection and 2 are used for inflection level selection.

The synthesizer provides a clock output which must be used to time the input data. Data should only be presented or changed on the positive transition of this clock.

Also provided is a status indicator (zero decode), which signals the presence of input data. This is useful when the synthesizer is operated from a buffer memory interface.

All signals are TTL signal elvels, except audio output.

Pin Description, Conn. 6, (Front Mother Board)

Connector Pin No.	Function	Description	See Signal Function
4	Data In, Inflection MSB	1 TTL load, Neg. True	A
5	Data In, Inflection LSB	1 TTL load, Neg. True	A
6	Data In, Phoneme MSB	1 TTL load, Neg. True	A
7	Data In, Phoneme	1 TTL load, Neg. True	A
8	Data In, Phoneme	1 TTL load, Neg. True	A
9	Data In, Phoneme	1 TTL load, Neg. True	A
10	Data In, Phoneme	1 TTL load, Neg. True	A
11	Data In, Phoneme LSB	1 TTL load, Neg. True	A
2	Output, Phoneme Clock	2 TTL loads, Neg. Pulse	B
3	Output, Zero Decode	2 TTL loads, Pos. Ture	C
13	Output, Audio Control	Ext. 50K Vol. Con. Wiper	D
15	Output, Audio Ground	Audio Return	D
1 & 10	Output, +5 VDC	Interface Supply	E
S	Output, +12 VDC	Interface Supply	E
P	Output, – 12 VDC	Interface Supply	E
A&N	Ground	Signal Return	E

SIGNAL FUNCTION DESCRIPTION

A. Data In (Pins 4-11) Six bit phoneme code and 2 bit inflection code inputs. All phoneme inputs high are Null code causing no output condition.

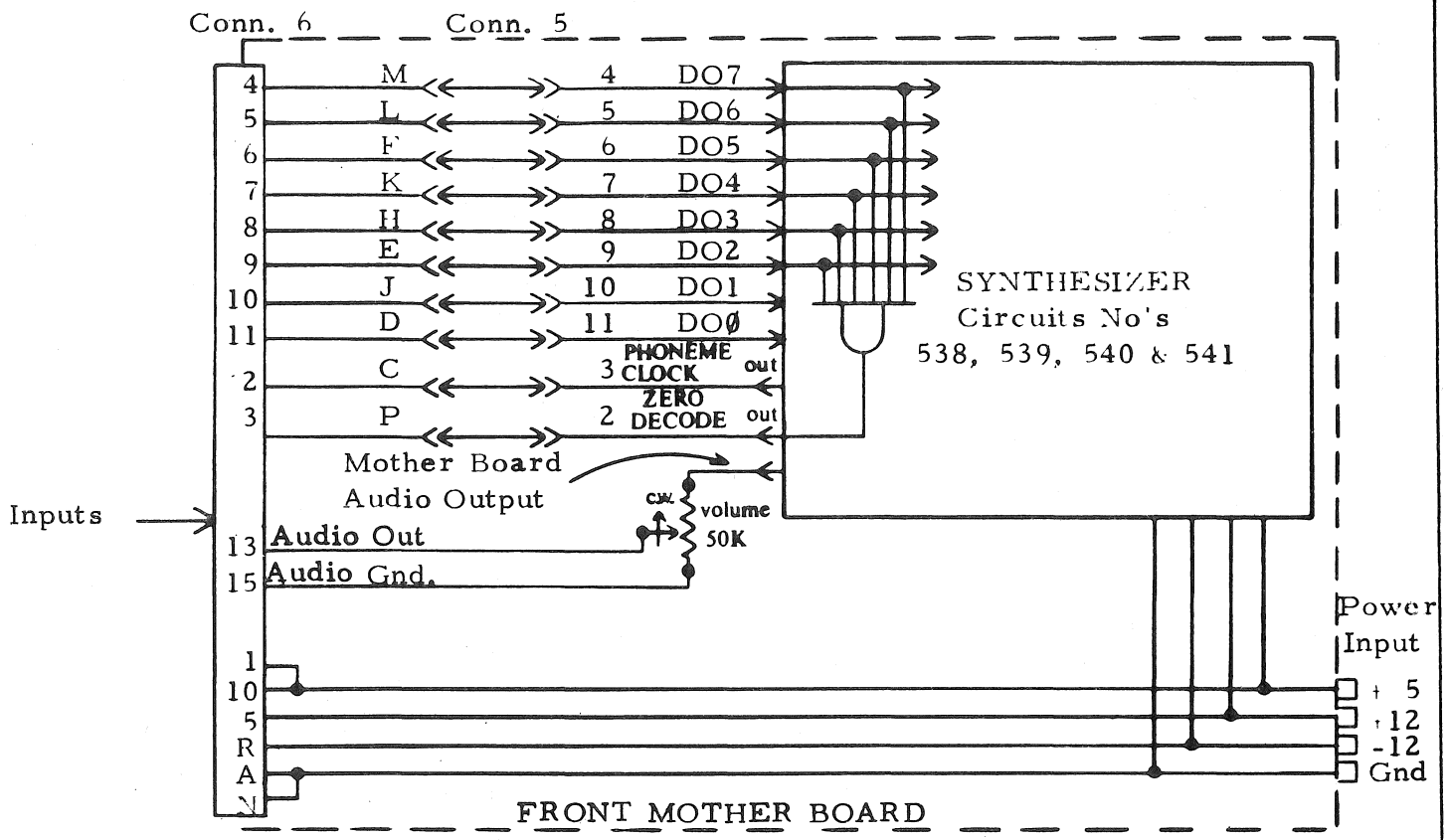
B. Phoneme Clock Output (Pin 2) Provides internal timing of synthesizer and must be used to control data input. Data should only be presented or changed on the positive edge of this signal. When all phoneme data inputs are high, this clock runs with a period of 6 milliseconds. When data input is present, the period lengthens, depending on speech rate and the particular phoneme.

C. Zero Decode Output (Pin 3) This signal indicates the status of data by a high signal in absence of data and a low signal when any phoneme data input is low.

D. Audio Control Output (Pins 13 & 15) 1-2 volt maximum audio signal from 50K external volume control. Additional external audio amplifier required to drive speaker.

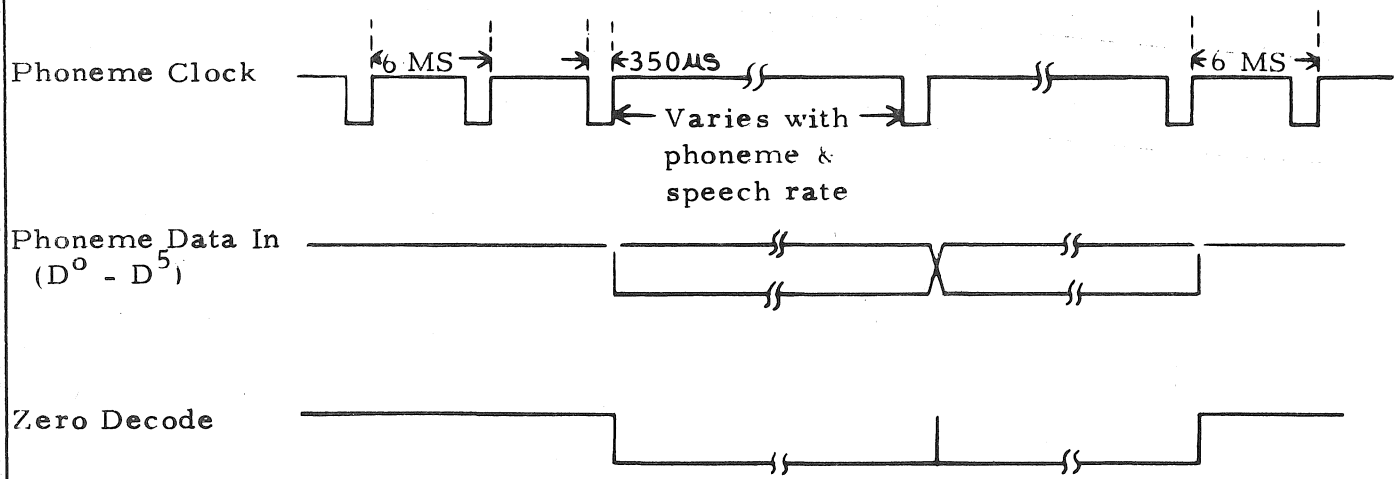
E. Power Supply Voltages. The basic synthesizer requires the following power:
+5 VDC regulated @ 120 MA
+12 VDC regulated @ 160 MA
–12 VDC regulated @ 200 MA
Pins A&N should be used for power supply returns, and Pin 15 for external audio amplifier ground return.

III. DIAGRAMS



NOTES: 1. Conn. 6 mounted on front mother board.
2. Conn. 5 requires jumpers as shown.
3. Power supplied to push on tabs on mother board.
4. Audio control connected to solder points on mother board.

INPUT/OUTPUT WAVEFORMS



NOTES:

1. Data in should be changed on positive edge of phoneme clock.
2. Zero decode output may contain spikes when data changes.
3. Data in is negative true (1 TTL load with 4.7K pullup resistor).

BAD BIT GETTERS:
Memory Test Programs

by Ray Boaz

(reprinted with permission from *Homebrew Computer Club Newsletter*)

Every computer system needs a memory test program or two to ensure a high level of confidence in the memory system hardware. One bad memory bit can send a program off to Never-Never Land. The memory test programs listed here are for use with 6800 systems operating with MIKBUG as a system monitor. MIKBUG is a simple monitor (as it was meant to be) which has many useful subroutines. Several of them are made use of in these memory test programs.

The terms used herein are consistent with the 6800 nomenclature. A and B are the two accumulators, X is the index register, PC is the program counter, SP is the stack pointer, CCR is the condition code register, and M is the memory location of interest.

Of the two programs, the shortest, MT1, is written to be used in the 6810 RAM used by MIKBUG as a scratch-pad. It is 27 bytes long and fits into address space A060 to A07A. This should work well for the SWTP 6800 Computer Systems. The second, MT2, is a more general test program but takes up 58 bytes total. The start and end addresses in MT1 are direct operands and therefore, shown blank in the listing. MT2 uses locations A002-3 for start, and A004-5 for end addresses. So these locations must be loaded before the program is started. Also in MT2 the I/O interrupt pointer locations (A000-1 are used so if applicable it must be set after running MT2. In general, both MT1 and MT2 operate the same until an error is found.

In both programs a store is made to an M start address, then the data is fetched back, compared to good data, and if true, it is incremented to test M again with A+1. This continues for all 256 bit combinations—00 to FF. Then X is incremented, and the next location is tested. This continues until the address is encountered.

In MT1, on finding an error, a branch to MIKBUG software interrupt is made which results in the status registers being printed out as follows: CCR-B-A-X-PC-SP. B is the good data, A is the error data, and X is the address with the error. If testing 1K bytes, 2000-23FF, and an error was found which resulted in B=00, A=00 and X=007, the chip containing data bit 7 (MSB) at M=2007 is bad. M+1 must be loaded as the new start address to continue the test to end address.

MT2 on finding an error goes to a routine to print A-B-X, then continues to the next M until end address is reached. Here again, A=error data, B=good data, and X=M error.

Both of these programs have worked well as memory testers and as chip testers for 2102 type memory chips.

MT1

```
A060 CE  ____   START  LDX   LOAD X WITH START
                              ADDR
A063 4F         OMT    CLRA  CLEAR A
A064 5F                CLRB  CLEAR B
A065 06                TAP   CLEAR CCR
A066 A7  00     NVAL   STAA  STORE A AT X ADDR
A068 A6  00            LDAA  LOAD A WITH DATA
                              JUST STORED
A06A 11                CBA   COMPARE B-A
A06B 26  OD            BNE   IF NOT SAME BRANCH
                              TO ERROR
A06D 4C                INCA  ELSE INCREMENT A
A06E 5C                INCB  INCREMENT B
A06F 26  F5            BNE   IF B NOT EQUAL TO 00
                              BRANCH TO NVAL
A071 08                INX   ELSE INCREMENT X
A072 8C  ____          CPX   COMPARE X TO END
                              ADDR
A075 26  EC            BNE   IF NOT SAME BRANCH
                              TO OMT
A077 7E  E0E3          JMP   ELSE JUMP TO MIKBUG
                              CONTROL
A07A 3F         ERROR  SWI   PRINT ERROR STATUS
```

MT2

```
0000 FE  A002   START  LDX   LOAD X WITH START
                              ADDR
0003 4F         OMT    CLRA  CLEAR A
0004 5F                CLRB  CLEAR B
0005 06                TAP   CLEAR CCR
0006 A7  00     NVAL   STAA  STORE A AT X ADDR
0008 A6  00            LDAA  LOAD A WITH DATA
                              JUST STORED
000A 11                CBA   COMPARE B-A
000B 26  OD            BNE   IF NOT SAME BRANCH
                              TO ERROR
000D 4C                INCA  ELSE INCREMENT A
000E 5C                INCB  INCREMENT B
000F 26  F5            BNE   IF B NOT EQUAL TO 00
                              BRANCH TO NVAL
0011 08                INX   ELSE INCREMENT X
0012 BC  A004          CPX   COMPARE X TO END
                              ADDR+1 AT A004
0015 26  EC            BNE   IF NOT SAME BRANCH
                              TO OMT
0017 7E  E0E3          JMP   ELSE JUMP TO MIKBUG
                              CONTROL
001A B7  A000   ERROR  STAA  STORE A AT A000
001D F7  A001          STAB  STORE B AT A001
0020 FF  A002          STX   STORE X AT A002
0023 CE  E19D          LDX   LOAD X WITH ADDR
                              OF CR/LF/* STRING
0026 BD  E07E          JSR   JUMP TO PRINT CR/LF/*
0029 CE  A000          LDX   LOAD X WITH ADDR OF
                              ERROR DATA
002C BD  E0CA          JSR   JUMP TO PRINT ERROR
                              (A)
002F BD  E0CA          JSR   JUMP TO PRINT DATA
                              (B)
0032 BD  E0C8          JSR   JUMP TO PRINT ERROR
                              ADDR (X)
0035 7C  A003          INC   INCREMENT M FOR
                              NEW START ADDR
0038 20  C6            BRA   BRANCH TO START
```

UNIZAP—A MODIFICATION OF 'SHOOTING STARS'

by John C. Shepard

EDITOR'S NOTE: A game called TEASER, written in BASIC, was published in the September, 1974, issue of *PCC*. The game was redone by Willard I. Nico under the title of "Shooting Stars", and was published as an eight-page article in the May, 1976, issue of *Byte* (pages 42-49). UNIZAP is a variation of Nico's "Shooting Stars" game.

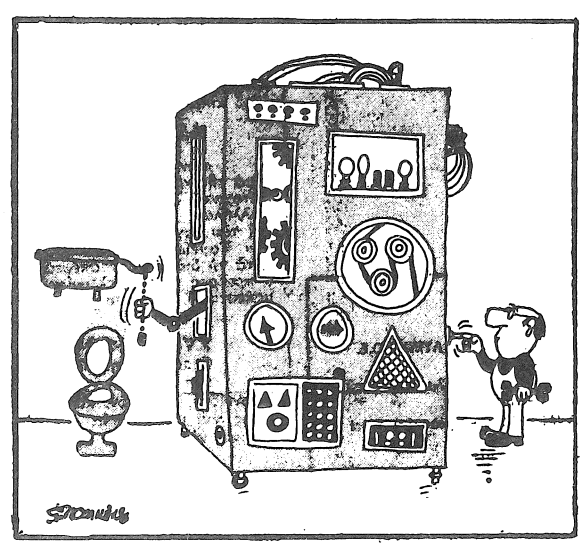
```
                        1   ; 'UNIZAP' - A MODIFICATION OF 'SHOOTING STARS', A P
                        2   ; DESCRIBED IN THE MAY ISSUE OF BYTE MAGAZINE (# 9).
                        3   ;
                        4   ; CONVERTED TO RUN ON INTEL SDK-80 8080 SYSTEMS BY
                        5   ;            JOHN C. SHEPARD
                        6   ;
                        7   ; THIS PROGRAM IS PUBLIC DOMAIN... SEE BYTE MAG. FOR
                        8   ; **************************************************
                        9   ;
                        10          ORG  1000H          ; A GOOD PLACE TO START!
                        11  GETCH   EQU  021BH          ; INPUT ROUTINE user-written
                        12  ECHO    EQU  01F4H          ; ECHO ROUTINE user-written
                        13  HOLE    EQU  2DH            ; A DOT
                        14  STAR    EQU  2AH            ; A STAR
                        15  WHY     EQU  59H            ; A "Y"
                        16  SPACE   EQU  20H            ; A SPACE
                        17  QUE     EQU  51H            ; A "Q"
                        18  COLON   EQU  3AH            ; A :
                        19  ZERO    EQU  30H            ; A "0"
                        20  CR      EQU  0DH            ; A ZERO
                        21  MESND   EQU  0FFH           ; END OF MESSAGE FLAG
1000 31 EE 12           22  START: LXI  SP,12EEH    ; SET UP STACK PTR.
1003 21 25 12           23         LXI  H,TITLE     ; LOAD POINTER TO TITLE M
1006 CD 32 11           24         CALL MESAGE      ; PRINT IT
1009 06 00              25         MVI  B,00H       ; INITIALIZE UNIVERSE
100B 0E 01              26         MVI  C,01H       ;    TO STARTING PATTERN
100D 50                 27         MOV  D,B         ; CLEAR SHOT COUNTER
100E 14                 28  UNIDIS: INR  D          ; INCREMENT SHOT COUNTER
100F 1E 0A              29  UNLOOP: MVI  E,0AH      ; LOOP COUNT 10 ITERATION
1011 1D                 30  ULOOP2: DCR  E          ; DECREMENT LOOP COUNTER
1012 CA 9F 10           31         JZ   WINTST      ; IF LOOP DONE, GO TO WIN
1015 78                 32         MOV  A,E         ; MOVE COUNTER FOR TESTIN
1016 FE 06              33         CPI  06H         ; IS LOOP IN 4TH CYCLE?
1018 CA 3E 10           34         JZ   CRLF        ; IF SO, DO A CR/LF
101B FE 03              35         CPI  03H         ; IS LOOP IN 7TH CYCLE?
101D CA 3E 10           36         JZ   CRLF        ; IF SO, DO A CR/LF
1020 FE 05              37         CPI  05H         ; IS IT STAR FIVE?
1022 CA 46 10           38         JZ   TEST5       ; IF SO, SO 'SPECIAL' FET
1025 AF                 39  NEDOT:  XRA  A           ; CLEAR CARRY (AND A)
1026 78                 40         MOV  A,B         ; MOVE UNIVERSE TO A
1027 0F                 41         RRC              ; ROTATE NEXT PLACE INTO
1028 47                 42         MOV  B,A         ; SAVE IT IN B FOR A WHIL
1029 D2 31 10           43  PSUDOT: JNC  LOADOT      ; IF BLACK HOLE, DO BLACK
102C 3E 2A              44         MVI  A,STAR      ; ELSE LOAD A STAR
102E C3 33 10           45         JMP  DOIT        ; AND PRINT IT
1031 3E 20              46  LOADOT: MVI  A,HOLE     ; LOAD A BLACK HOLE
1033 CD 28 11           47  DOIT:   CALL OUTPUT     ; PRINT THE CHARACTER
1036 3E 20              48         MVI  A,SPACE     ;  LOAD A SPACE
1038 CD 28 11           49         CALL OUTPUT      ; PRINT THAT, TOO
103B C3 11 10           50         JMP  ULOOP2      ; GO BACK INTO UNIVERSE P
103E 3E 0D              51  CRLF:   MVI  A,CR        ; LOAD A CARRIAGE RETURN
1040 CD 28 11           52         CALL OUTPUT      ; PRINT IT. OUTPUT SUPPLI
1043 C3 25 10           53         JMP  NEDOT       ; GO BACK AND DO MORE UNI
1046 AF                 54  TEST5:  XRA  A           ; CLEAR CARRY (AND A)
1047 79                 55         MOV  A,C         ; GET POSITION FIVE STATU
1048 0F                 56         RRC              ; PUT STATUS INTO CARRY
1049 C3 29 10           57         JMP  PSUDOT      ; REJOIN MAIN ROUTINE
104C 21 61 11           58  GOTSTR: LXI  H,URSHOT   ; POINT TO 'YOUR SHOT?' M
104F CD 32 11           59         CALL MESAGE      ; PRINT MESSAGE
1052 CD 1B 11           60         CALL INPUT       ; GET PLAYER'S INPUT
1055 1E 09              61         MVI  E,09H       ; LOOP COUNT FOR TABLE SE
1057 21 3D 11           62         LXI  H,MSKTBL    ; POINT TO THE MASK TABLE
105A BE                 63  NXTGRP: CMP  M           ; DOES INPUT EQUAL TABLE
105B CA 69 10           64         JZ   FOUND       ; IF SO, GO ALTER UNIVERS
105E 1D                 65         DCR  E           ; OTHERWISE, DECREMENT LO
105F CA 88 10           66         JZ   INVALD      ; IF LOOP HAS ENDED, CHAR
1062 23                 67         INX  H           ; OTHERWISE INCREMENT
1063 23                 68         INX  H           ;  TABLE POINTER FOUR
1064 23                 69         INX  H           ;   TIMES TO GET TO NEXT
1065 23                 70         INX  H           ;  ENTRY
1066 C3 5A 10           71         JMP  NXTGRP      ; THEN TEST THAT ENTRY
1069 23                 72  FOUND:  INX  H           ; POINT TO POSITION MASK
106A 7E                 73         MOV  A,M         ; AND LOAD MASK INTO A
106B A7                 74         ANA  A           ; SET THE FLAGS
106C C2 78 10           75         JNZ  UNI2A       ; IF MASK NOT 0 THEN FRIN
106F 79                 76         MOV  A,C         ; OTHERWISE THE CENTER PO
1070 FE 01              77         CPI  01H         ; IS STAR IN CENTER?
1072 C2 12 11           78         JNZ  BADMOV      ; IF NOT THEN IT'S A WRON
1075 C3 7D 10           79         JMP  NXTBYT      ; ELSE GO PROCESS STAR
1078 78                 80  UNI2A:  MOV  A,B         ; REST OF UNIVERSE TO A
1079 A6                 81         ANA  M           ; AND WITH MASK TO ISOLAT
107A CA 12 11           82         JZ   BADMOV      ; IF NO STAR THEN WRONG M
107D 23                 83  NXTBYT: INX  H           ; POINT TO GALAXY MASK
107E 78                 84         MOV  A,B         ; FETCH UNIVERSE AGAIN
107F AE                 85         XRA  M           ; AND COMPLEMENT IT (THAN
1080 47                 86         MOV  B,A         ; THEN REPLACE IT
1081 23                 87         INX  H           ; POINT TO CENTER MASK
1082 79                 88         MOV  A,C         ; FETCH CENTER OF UNIVERS
1083 AE                 89         XRA  M           ; COMPLEMENT CENTER
1084 4F                 90         MOV  C,A         ; AND SAVE THE CENTER OF
1085 C3 0E 10           91         JMP  UNIDIS      ; GO DISPLAY THE NEW UNIV
1088 FE 51              92  INVALD: CPI  QUE         ; WAS SHOT 'QUIT'?
108A C2 96 10           93         JNZ  NOTVAL      ; IF NOT DO BAD STAR ROUTI
108D 21 77 11           94         LXI  H,HEQUIT    ; POINT TO 'YOU QUIT' MES
1090 CD 32 11           95         CALL MESAGE      ; PRINT MESSAGE
1093 C3 EF 10           96         JMP  AGAIN       ; SEE IF HE WANTS TO TRY
1096 21 6F 12           97  NOTVAL: LXI  H,BADNUM    ;POINT TO BAD MESG.
1099 CD 32 11           98         CALL MESAGE      ; PRINT IT
109C C3 0F 10           99         JMP  UNLOOP      ; GO BACK AND REPRINT UNI
109F 78                 100 WINTST: MOV  A,B         ; MOVE UNIVERSE TO A
10A0 FE FF              101        CPI  0FFH        ;  ARE ALL FRINGE STARS P
10A2 C2 00 11           102        JNZ  LOSTST      ; IF NOT, SEE IF PLAYER H
10A5 79                 103        MOV  A,C         ; FETCH CENTER OF UNIVERS
10A6 A7                 104        ANA  A           ; SET FLAGS
10A7 C2 4C 10           105        JNZ  GOTSTR      ; IF CENTER NOT EMPTY, NO
10AA 21 92 11           106        LXI  H,WINMS1    ; POINT TO 1ST HALF OF WI
10AD CD 32 11           107        CALL MESAGE      ; AND PRINT WIN MESSAGE
10B0 1E 30              108        MVI  E,ZERO      ;  BEGIN BINARY TO DECIMA
10B2 43                 109        MOV  B,E         ; CONVERSION BY SETTING
10B3 48                 110        MOV  C,E         ; REGISTERS TO (ASCII) ZE
10B4 15                 111        DCR  D           ; GET RID OF LAST SHOT
10B5 7A                 112        MOV  A,D         ; MOVE SHOT COUNT TO A FO
10B6 A7                 113        ANA  A           ; SET FLAGS
10B7 CA E5 10           114        JZ   LSTSIG      ; IF ZERO, SKIP CONVERSIO
10BA 3E 3A              115        MVI  A,COLON     ;  SET A TO OVERFLOW COD
10BC 1C                 116 MORDEC: INR  E           ; COUNT UP 1 IN 1'S DIGIT
10BD B8                 117        CMP  E           ; IS IT = TO OVERFLOW?
10BE C2 C8 10           118        JNZ  TALLY       ; IF NOT THEN TALLY AND C
10C1 1E 30              119        MVI  E,ZERO      ;  ELSE RESET 1'S DIGIT T
10C3 0C                 120        INR  C           ; AND CARRY INTO NEXT DIG
10C4 B9                 121        CMP  C           ; IS IT = TO OVERFLOW TOO
10C5 C2 C8 10           122        JNZ  TALLY       ; IF NOT THEN TALLY & CON
10C8 0E 30              123        MVI  C,ZERO      ;  ELSE RESET MIDDLE DIGI
10CA 04                 124        INR  B           ; AND CARRY INTO 100'S DI
10CB 15                 125 TALLY:  DCR  D           ; DECREMENT SCORE FOR TAL
10CC C2 BC 10           126        JNZ  MORDEC      ; IF NOT ZERO THEN KEEP L
10CF 78                 127        MOV  A,B         ; FETCH LEADING DIGIT TO
10D0 FE 30              128        CPI  ZERO        ;  IS IT (ASCII) ZERO?
10D2 C2 DE 10           129        JNZ  THREED      ; IF NOT DISPLAY 3 DIGITS
10D5 79                 130        MOV  A,C         ; FETCH MIDDLE DIGIT TO A.
10D6 FE 30              131        CPI  ZERO        ;  IS IT (ASCII) ZERO TOO
10D8 C2 E2 10           132        JNZ  MIDPRT      ; IF NOT DISPLAY 2 DIGITS
10DB C3 E5 10           133        JMP  LSTSIG      ; IF SO DISPLAY ONLY ONE
10DE CD 28 11           134 THREED: CALL OUTPUT      ; DISPLAY 3 DIGITS, LEFT
10E1 79                 135        MOV  A,C         ; FETCH MIDDLE DIGIT TO A
10E2 CD 28 11           136 MIDPRT: CALL OUTPUT      ; DISPLAY 2 DIGITS, LEFT
10E5 78                 137 LSTSIG: MOV  A,E         ; FETCH 1'S DIGIT
10E6 CD 28 11           138        CALL OUTPUT      ; DISPLAY REMAINING DIGIT
10E9 21 AC 11           139        LXI  H,WINMS2    ; POINT TO 2ND HALF OF WI
10EC CD 32 11           140        CALL MESAGE      ; PRINT THE MESSAGE
10EF 21 CA 11           141 AGAIN:  LXI  H,AGNMES    ; POINT TO 'AGAIN?' MESSA
10F2 CD 32 11           142        CALL MESAGE      ; PRINT IT
10F5 CD 1B 11           143        CALL INPUT       ; GET PLAYER'S RESPONSE
10F8 FE 59              144        CPI  WHY         ; IS ANSWER 'YES'?
10FA CA 00 10           145        JZ   START       ; IF SO, DO GAME AGAIN
10FD C3 07 00           146        JMP  0007H       ; ELSE, BACK TO KIT EXIT
1100 A7                 147 LOSTST: ANA  A           ; SET FLAGS
1101 C2 4C 10           148        JNZ  GOTSTR      ; CONTINUE IF FRINGE NOT
1104 79                 149        MOV  A,C         ; TEST CENTER POSITION
1105 A7                 150        ANA  A           ; SET FLAGS
1106 C2 4C 10           151        JNZ  GOTSTR      ; IF CENTER NOT ZERO, GO
1109 21 E8 11           152        LXI  H,LOSTMS    ; POINT TO 'YOU LOST' MES
110C CD 32 11           153        CALL MESAGE      ; PRINT IT
110F C3 EF 10           154        JMP  AGAIN       ; SEE IF HE WANTS TO PLAY
1112 21 01 12           155 BADMOV: LXI  H,BADMES    ; POINT TO 'BAD MOVE' MES
1115 CD 32 11           156        CALL MESAGE      ; PRINT IT
1118 C3 0F 10           157        JMP  UNLOOP      ; AND BACK INTO GAME
111B C5                 158 INPUT:  PUSH B           ; SAVE B,C REGS. FROM MON
111C CD 1B 02           159        CALL GETCH       ; GET A CHARACTER FROM US
111F CD F4 01           160        CALL ECHO        ; ECHO IT!
1122 59                 161        MOV  C,C         ; SAVE IT!
1123 0E 0D              162        MVI  C,CR        ; DO A CR/LF
1125 CD F4 01           163        CALL ECHO        ;  USING THE ECHOER
1128 78                 164        MOV  A,E         ; GET THE INPUT BACK
1129 C1                 165        POP  B           ; RESTORE B,C REGISTERS
112A C9                 166        RET              ; THEN JUST RETURN
112B C5                 167 OUTPUT: PUSH B           ; SAVE B,C REGISTERS FROM
112C 4F                 168        MOV  C,A         ; LOAD CHARACTER TO PRINT
112D CD F4 01           169        CALL ECHO        ; PRINT IT
1130 C1                 170        POP  B           ; RESTORE B,C REGS.
1131 C9                 171        RET              ; THEN RETURN
1132 7E                 172 MESAGE: MOV  A,M         ; GET CHAR. FROM MEMORY
1133 FE FF              173        CPI  MESND       ; IS IT STOP CODE?
1135 C8                 174        RZ               ;  YES. RETURN TO CALLER
1136 CD 28 11           175        CALL OUTPUT      ;  NO.  PRINT CHAR.
1139 23                 176        INX  H           ; INC. MESSAGE POINTER
113A C3 32 11           177        JMP  MESAGE      ; GO BACK AND GET SOME MO
```

```
113D 31          178  MSKTBL: DB   31H,01H,08H,01H
113E 01
113F 09
1140 01
1141 32          179          DB   32H,02H,07H,00H,33H
1142 02
1143 07
1144 00
1145 33
1146 04          180          DB   04H,16H,01H,34H,08H
1147 16
1148 01
1149 34
114A 08
114B 29          181          DB   29H,00H,35H,00H,5AH
114C 00
114D 35
114E 00
114F 5A
1150 01          182          DB   01H,36H,10H,94H,00H
1151 36
1152 10
1153 94
1154 00
1155 37          183          DB   37H,20H,68H,01H,38H
1156 20
1157 68
1158 01
1159 38
115A 40          184          DB   40H,0E0H,00H,39H,80H
115B E0
115C 00
115D 39
115E 80
115F D0          185          DB   0D0H,01H
1160 01
1161 0D          186  URSHOT: DB  CR
1162             187          ASC '       YOUR SHOT > '
1176 FF          188          DB  MESND
1177 0D          189  HEQUIT: DB  CR,CR,CR
1178 0D
1179 0D
117A             190          ASC 'YOU GIVE UP TOO EASILY'
1190 0D          191          DB  CR,MESND
1191 FF
1192 0D          192  WINMS1: DB  CR,CR,CR
1193 0D
1194 0D
1195             193          ASC  'YOU WON !!!'
11A0 0D          194          DB  CR
11A1             195          ASC 'YOU FIRED '
11AB FF          196          DB  MESND
11AC             197  WINMS2: ASC ' SHOTS ... BEST SCORE IS 11 '
11C8 0D          198          DB CR,MESND
11C9 FF
11CA 0D          199  AGNMES: DB CR
11CB             200          ASC 'WOULD YOU LIKE TO TRY AGAIN?'
11E7 FF          201           DB  MESND
11E8 0D          202  LOSTMS: DB  CR
11E9             203          ASC 'HEY! YOU LOST THE GAME'
11FF 0D          204          DB  CR,MESND
1200 FF
1201 0D          205  BADMES: DB  CR
1202 0D          206          DB  CR
1203             207          ASC 'YOU MAY ONLY SHOOT STARS, TURKEY'
1223 0D          208          DB  CR,MESND
1224 FF
1225 0D          209  TITLE:  DB  CR
1226 0D          210         DB CR,CR
1227 0D
1228             211          ASC  '*.*.*.* UNIZAP *.*.*.*'
123E 0D          212          DB  CR,CR
123F 0D
1240             213          ASC  '        1 2 3'
124D 0D          214          DB  CR
124E             215          ASC  '         4 5 6'
125C 0D          216          DB  CR
125D             217          ASC  '         7 8 9'
1268 0D          218          DB  CR,CR,CR,MESND
126C 0D
126D 0D
126E FF
126F 0D          219  BADNUM: DB  CR
1270             220          ASC  'TYPE 1 THRU 9 ... OR "Q" TO QUIT'
1290 0D          221          DB  CR,MESND
1291 FF
```

```
NO ERRORS FOUND IN THIS ASSEMBLY
31EE12212512CD3211C6W0E0W15W141E1A1DCA9F1W7DFEW6CA3E1WFEW3CA3E1WFEW5CA46
10AF7BW47D2311W3E2AC3331W3E2DCC2B113E2WCD2B11C3111W3EWDCD2B11C3251WAF79
0FC3291,216111CD3211CD1B111E0213D119ECA691W1DCA81W232323C35A1W237EA7
C2761W79FE1C21211C37D1W7DA6CA1211237WAF472379AE4FC3FE1,FE51C2961W217711
CD3211C3CF1W216F12CD3211C3CF1W78FEFFC2WW1179A7C24C1W219211CD3211E3W434B
157AA7CAE51W3E3A1CBWC2CB1W1E3,WC99C2B1WWE3CW415C2BC1W7DFE3CC2DE1W79FE3W
C2E21WC3E51WCD2B1179CD2B117BCD2B1121AC11CD3211C1A11CD3211CD1B11FE5WCAWW
1WC3W7DWA7C24C1W79A7C24C1W21E811CD3211C3EF1W21W112CD3211C3CF1WC5CD1BCCD
F4W159WE5DCDF4W179C1C9C54FCDF4C1E1C97EFEFFC2CD2B1123C3321131W1WW132,W27
WW33W416W134W29,W35,W5AW1361W94,W372W68W13,A3EWWW39WDWW1WD2W2W2,W2W2W
2W2W594F55522C534A4F542W3E2WFFWDWDWD594F552,W4749564520555,.2W544F4F2,W4541
53494C59WDFFWDWDWD594F552W574F4E2W212121WD594F552W4649524544 WFF2W53484F
54552W2E2E2E2W424553542W53434F52452W49532W31312WWDFFWD574F554C442W594F55
2W4C494D452W544F2W5452592W414741494E3FFFWD4H4559212W594F552W4C4F53542W54
4F452W47414D45WDFFWDWD594F552W4D41592W4F4E4C592W534F4F542W53544152532C
2W5455524D455WDFFWDWDW2A2E2A2E2A2E2A2W554E495A415W2W2A2E2A2E2A2E2A2WDWD
2W2W2W2W2W2W2W312W322W33WD2W2W2W2W2W2W2W342W352W36WD2W2W2W2W2,2W2W
2W2W372W382W39WDWDWDFFWD545950452W312W544F52552W392W2E2E2E2W4F522W225122
2W544F2W51554954WDFF
```

SYMBOL NAME	TABLE VALUE		
GETCH	0218	NXTBYT	127D
ECHO	01F4	INVALD	1238
HOLE	022D	NOTVAL	1206
STAR	022A	WINTST	129F
WHY	0259	MORDEC	128C
SPACE	0220	TALLY	12CB
QUE	0251	THREED	12DE
COLON	023A	MIDPRT	12E2
ZERO	0230	LSTSIG	12E5
CR	020D	AGAIN	12EF
MESND	02FF	LOSTST	1320
START	1200	BADMOV	1112
UNIOIS	120E	INPUT	1118
UNLOOP	120F	OUTPUT	1128
ULOOP2	1211	MESAGE	1132
NEDOT	1225	MSKTBL	1130
PSUDOT	1229	URSHOT	1161
LOADOT	1231	HEQUIT	1177
DOIT	1233	WINMS1	1192
CRLF	123E	WINMS2	11AC
TEST5	1246	AGNMES	11CA
GOTSTR	124C	LOSTMS	11E8
NXTGRP	125A	BADMES	1201
FOUND	1269	TITLE	1225
UNI2A	1278	BADNUM	126F

Note that the assembler doesn't calculate the 'ASC' values in the listing--but leaves space for them. The hex dump has the ASCII in it. Prog. requires less than 700 bytes. The 11 step solution is 5,2,1,8,5,3,2,7,9,8,5



Kamenos/Eleftherotypia/Athens

Palo Alto Tiny BASIC

by Li-Chen Wang

Within the first five issues of *Dr. Dobb's Journal* we have published *complete* details and listings of four versions of Tiny BASIC:

Jan. & Feb., 1976:	Whipple's & Arnold's Tiny BASIC Extended (TBX)
March, 1976:	Greeb's Denver Tiny BASIC
April, 1976:	Mueller's MINOL
May, 1976 (now):	Wang's Palo Alto Tiny BASIC

All of these have been for 8080's. We think this is enough for the 8080, at least for the time being. Therefore, we will publish no more 8080 versions unless they have some very unusual or valuable features, for instance:

- floating point package (well documented)
- comprehensive string & substring operators
- well-designed graphics/display interface & commands
- complete interface & commands for mass storage
- significant program & variables editing features
- extensive package of interest built-in functions
- full BASIC capabilities in under two kilobytes

We are *very* interested, however, in publishing complete information on some versions of Tiny BASIC for the many *other* microprocessors, for example:

8008	PACE	6502
6800	SC/MP	ETC.

Also, as you read through the articles on the first four implementations, you should see better ways to do things, features that can easily be added, improvements that are possible in the documentation, etc. We hope you will implement those, as you note them; write them up, and forward them for publication in the *Journal* as enhancements to these four Tiny BASIC dialects. --Jim Warren, Jr., Editor

THE LANGUAGE

Numbers

All numbers are integers and must be less than 32767.

Variables

There are 26 variables denoted by letters A through Z. There is also a single array @(I). The dimension of this array is set automatically to make use of all the memory space that is left unused by the program. (i.e., 0 through SIZE/2, see SIZE function below.)

Functions

There are 3 functions:
ABS(X) gives the absolute value of X.
RND(X) gives a random number between 1 and X (inclusive).
SIZE gives the number of bytes left unused by the program.

Arithmetic and Compare Operators

```
/    divide.
*    multiply.
-    subtract.
+    add.
>    greater than (compare).
<    less than (compare).
=    equal to (compare).
#    not equal to (compare).
>=   greater than or equal to (compare).
<=   less than or equal to (compare).
```

+, -, *, and / operations result in a value between -32767 and 32767. (-32768 is also allowed in some cases.) All compare operators result in a 1 if true and a 0 if not true.

Expressions

Expressions are formed with numbers, variables, and functions with arithmetic and compare operators between them. + and - signs can also be used at the beginning of an expression. The value of an expression is evaluated from left to right, except that * and / are always done first, and then + and -, and then compare operators. Parentheses can also be used to alter the order of evaluation. Note that compare operators can be used in any expression. For example:

```
10 LET A=(X>Y)*123+(X=Y)*456+(X<Y)*789
20 IF (U=1)*(V<2)+(U>V)*(U<99)*(V>3) PRINT "YES"
30 LET R=RND(100), A=(R>3)+(R>15)+(R>56)+(R>98)
```

In statement 10, A will be set to 123 if X>Y, to 456 if X=Y, and to 789 if X<Y. In statement 20, the "*" operator acts like a logical AND, and the "+" operator acts like a logical OR. In statement 30, Y will be a random number between 0

and 4 with a prescribed probability distribution of: 3% of being 0, 15-3=12% of being 1, 56-15=41% of being 2, 98-56=42% of being 3, and 100-98=2% of being 4.

Direct Commands

All the commands described later can be used as direct commands except the following three, they can only be used as direct command and not as part of a statement:

RUN

will start to execute the program starting at the lowest statement number.

LIST

will print out all the statmentes in numerical order.

LIST 120

will print out all the statmentes in numerical order starting at statement 120.

NEW

will delete all statements.

Abbreviation and blanks

You may use blanks freely, except that numbers, command key words, and function names can not have embedded blanks.

You may truncate all command keywords and function names and follow them by a period. "P.", "PR.", "PRI.", and "PRIN." all stand for "PRINT". Also the word LET in LET command can be omitted. The "shortest" abbreviation for all keywords are as follows:

A.=ABS	F.=FOR	GOS.=GOSUB	G.=GOTO
IF=IF	IN.=INPUT	L.=LIST	N.=NEW
N.=NEXT	P.=PRINT	REM=REMARK	R.=RETURN
R.=RND	R.=RUN	S.=SIZE	S.=STEP
S.=STOP	TO=TO		

Implied = LET

Statements

A statement consists of a statement number of between 1 and 32767 followed by one or more commands. Commands in the same statement are separated by a semi-colon ";". "GOTO", "STOP", and "RETURN" commands must be the last command in any given statement.

Commands

Tiny Basic commands are listed below with examples. Remember that cammands can be concatenated with semi-colons. In order to store the statement, you must also have a statement number in front of the commands. The statement number and the concatenation are not shown in the examples.

REM or REMARK Command

```
REM anything goes
```

This line will be ignored by TBI.

LET Command

```
LET A=234-5*6, A=A/2, X=A-100, @(X+9)=A-1
```

will set the variable A to the value of the expression 234-5*6 (i.e., 204), set the variable A (again) to the value of the expression A/2 (i.e., 102), set the variable X to the value of the expression A-100 (i.e., 2), and then set the variable @(11) to 101 (where 11 is the value of the expression X+9 and 101 is the value of the expression A-1).

```
LET U=A#B, V=(A>B)*X+(A<B)*Y
```

will set the variable U to either 1 or 0 depending on whether A is not equal to or is equal to B; and set the variable V to either X, Y or 0 depending on whether A is greater than, less than, or equal to B.

PRINT Command

```
PRINT
```

will cause a carriage-return (CR) and a line-feed (LF) on the output device.

```
PRINT A*3+1, "ABC 123 !@#", ' CBA '
```

will print the value of the expression A*3+1 (i.e., 307), the string of characters "ABC 123 !@#", and the string " CBA ", and then a CR-LF. Note that either single or double quotes can be used to quote strings, but pairs must be matched.

```
PRINT A*3+1, "ABC 123 !@#", ' CBA ',
```

will produce the same output as before, except that there is no CR-LF after the last item is printed. This enables the program to continue printing on the same line with another "PRINT".

```
PRINT A, B, #3, C, D, E, #10, F, G
```

will print the values of A and B in 6 spaces, the values of C, D, and E in 3 spaces, and the values of F and G in 10 spaces. If there are not enough spaces specified for a given value to be printed, the value will be printed with enough spaces anyway.

```
PRINT 'ABC',-,'XXX'
```

will print the string "ABC", a CR without a LF, and then the

string "XXX" (over the ABC) followed by a CR-LF.

INPUT Command

INPUT A, B

When this command is executed, Tiny Basic will print "A:" and wait to read in an expression from the input device. The variable A will be set to the value of this expression. Then "B:" is printed and varable B is set to the value of the next exprssion read from the input device. Note that not only numbers, but also exprssions can be read as input.

INPUT 'WHAT IS THE WEIGHT'A, "AND SIZE"B

This is the same as the command above, except the prompt "A:" is replaced by "WHAT IS THE WEIGHT:" and the prompt "B:" is replaced by "AND SIZE:". Again, both single and double quotes can be used as long as they are matched.

INPUT A, 'STRING', -, "ANOTHER STRING", B

The strings and the "-" have the same effect as in "PRINT".

IF Command

IF A<B LET X=3; PRINT 'THIS STRING'

will test the value of the expression A<B. If it is not zero (i.e., if it is true), the commands in the rest of this statement will be executed. If the value of the expression is zero (i.e., if it is not true), the rest of this statement will be skipped over and execution continues at next statement. Note that the word "THEN" is not used.

GOTO Command

GOTO 120

will cause the execution to jump to statement 120. Note that GOTO command cannot be followed by a semi-colon and other commands. It must be ended with a CR.

GOTO A*10+B

will cause the execution to jump to a different statement number as computed from the value of the expression.

GOSUB and RETURN Commands

GOSUB command is similar to GOTO command except that: a) the current statement number and position whthin the statement is remembered; and b) a semi-colon and other commands can follow it in the same statement.

GOSUB 120

will cause the execution to jump to statement 120.

GOSUB A*10+B

will cause the execution to jump to different statements as computed from the value of the expression A*10+B.

RETURN

A RETURN command must be the last command in a statement and followed by a CR. When a RETURN command is encountered, it will cause the execution to jump back to the command following the most recent GOSUB command.

GOSUB can be nested. The depth of nesting is limited only by the stack space.

FOR and NEXT Commands

FOR X=A+1 TO 3*B STEP C-1

The variable X is set to the value of the expression A+1. The values of the expressions (not the expressions themselves) 3*B and C-1 are remembered. The name of the variable X, the statement number and the position of this command within the statement are also remembered. Execution then continues the normal way until a NEXT command is encountered.

The STEP can be positive, negative or even zero. The word STEP and the expression following it can be omitted if the desired STEP is +1.

NEXT X

The name of the variable (X) is checked with that of the most recent FOR command. If they do not agree, that FOR is terminated and the next recent FOR is checked, etc. When a match is found, this variable will be set to its current value plus the value of the STEP expression saved by the FOR command. The updated value is then compared with the value of the TO expression also saved by the FOR command. If this is within the limit, execution will jump back to the command following the FOR command. If this is outside the limit, execution continues following the NEXT command itself.

FOR can be nested. The depth of nesting is limited only by the stack space. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR will be terminated automatically.

STOP Command

STOP

This command stops the execution of the program and returns control to direct commands from the input device. It can appear many times in a program but must be the last command in any given statement. i.e., it cannot be followed by a semi-colon and other commands.

Stopping the Execution

The execution of program or listing of program can be stopped

by the Control-C key on the input device.

Control of Output Device

The Control-O key on the input device can be used to turn the output device ON and OFF. This is useful when you want to read in a program punched on paper tape.

To produce such a paper tape, type "LIST" without CR. Turn on the paper tape punch and type a few Control-Shift-P's and then a CR. When listing is finished, type more Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW", CR, and Control-O, then turn on the paper tape reader. When the paper tape is read, turn the reader off and type a Control-O again.

Error Report

There are only three error conditions in TINY BASIC. The statement with the error is printed out with a question mark inserted at the point where the error is detected.

(1) WHAT? means it does not understand you. Example:

```
WHAT?
210 P?TINT "THIS"          where PRINT is misstyped

WHAT?
260 LET A=B+3, C=(3+4?, X=4
```

(2) HOW? means it understands you but does not know how to do it.

```
HOW?
310 LET A=B*C?+2        where B*C is greater than 32767

HOW?
380 GOTO 412?           where 412 does not exist
```

(3) SORRY means it understands you and knows how to do it but there is not enough memory to do it.

Error Corrections

If you notice an error in typing before you hit the CR, you can delete the last character by the Rub-Out key or delete the entire line by the Alt-Mode key. Tiny basic will echo a back-slash for each Rub-Out. Echo for Alt-Mode consists of a LF, a CR, and an up-arrow.

To correct a statement, you can retype the statement number and the correct commands. Tiny Basic will replace the old statement with the new one.

To delete a statement, type the statement number and a CR only.

Verify the corrections by "LIST nnnn" and hit the Control-C key while the line is being printed.

```
*************************************************************
*
*                  TINY BASIC FOR INTEL 8080
*                        VERSION 1.0
*                    BY LI-CHEN WANG
*                     10 JUNE, 1976
*                       @COPYLEFT
*                   ALL WRONGS RESERVED
*
*************************************************************
*
*   *** ZERO PAGE SUBROUTINES ***
*
*   THE 8080 INSTRUCTION SET LETS YOU HAVE 8 ROUTINES IN LOW
*   MEMORY THAT MAY BE CALLED BY RST N. N BEING 0 THROUGH 7.
*   THIS IS A ONE BYTE INSTRUCTION AND HAS THE SAME POWER AS
*   THE THREE BYTE INSTRUCTION CALL LLHH.  TINY BASIC WILL
*   USE RST 0 AS START OR RESTART AND RST 1 THROUGH RST 7 FOR
*   THE SEVEN MOST FREQUENTLY USED SUBROUTINES.
*   TWO OTHER SUBROUTINES (CRLF AND TSTNUM) ARE ALSO IN THIS
*   SECTION.  THEY CAN BE REACHED ONLY BY 3-BYTE CALLS.
*
                        ORG   X'0000'
0000 F3          START  DI    .          *** START/RESTART ***
0001 310020 @@@@        LODI  SP,STACK   INITIALIZE THE STACK
0004 C3BA00             JMP   ST1        GO TO THE MAIN SECTION
0007 4C                 CHAR  'L'
*
0008 E3                 XCH   HL,(SP)    *** TSTC OR RST 1 ***
0009 EF                 IGNBLK           IGNORE BLANKS AND
000A BE                 CMP   M          TEST CHARACTER
000B C36800             JMP   TC1        REST OF THIS IS AT TC1
*
000E 3E0D        CRLF   LODI  A,@CR      *** CRLF ***
*
0010 F5                 PUSH  AF         *** OUTC OR RST 2 ***
0011 3A0008             LD    A,OCSW     PRINT CHARACTER ONLY
0014 B7                 IOR   A          IF OCSW SWITCH IS ON
0015 C31A07             JMP   OC2        REST OF THIS IS AT OC2
*
0018 CD5504             CALL  EXPR2      *** EXPR OR RST 3 ***
0018 E5                 PUSH  HL         EVALUATE AN EXPRESION
001C C31104             JMP   EXPR1      REST OF IT IS AT EXPR1
001F 57                 CHAR  'W'
*
0020 7C                 LOD   A,H        *** COMP OR RST 4 ***
0021 BA                 CMP   D          COMPARE HL WITH DE
0022 C0                 RET   NZ         RETURN CORRECT C AND
0023 7D                 LOD   A,L        Z FLAGS
0024 BB                 CMP   E          BUT OLD A IS LOST
0025 C9                 RET   U
0026 414E               CHAR  'AN'
*
0028 1A          SS1    LD    A,(DE)     *** IGNBLK/RST 5 ***
0029 FE20               CMPI  ' '        IGNORE BLANKS
002B C0                 RET   NZ         IN TEXT (WHERE DE->)
002C 13                 INC   DE         AND RETURN THE FIRST
002D C328C0             JMP   SS1        NON-BLANK CHAR. IN A
*
0030 F1                 POP   AF         *** FINISH/RST 6 ***
0031 CD9105             CALL  FIN        CHECK END OF COMMAND
0034 C3A405             JMP   QWHAT      PRINT "WHAT?" IF WRONG
0037 47                 CHAR  'G'
*
0038 EF                 IGNBLK           *** TSTV OR RST 7 ***
0039 D640               SUBI  '@'        TEST VARIABLES
003B D8                 RET   C          C:NOT A VARIABLE
003C C2580C             JMP   NZ,TV1     NOT "@" ARRAY
003F 13                 INC   DE         IT IS THE "@" ARRAY
0040 CDFB04             CALL  PARN       @ SHOULD BE FOLLOWED
0043 29                 ADD   HL,HL      BY (EXPR) AS ITS INDEX
0044 DA9F00             JMP   C,QHOW     IS INDEX TOO BIG?
0047 D5                 PUSH  DE         WILL IT OVERWRITE
0048 EB                 XCH   HL,DE      TEXT?
0049 CD3D05             CALL  SIZE       FIND SIZE OF FREE
004C E7                 COMP             AND CHECK THAT
004D DAD005             JMP   C,ASORRY   IF SO, SAY "SORRY"
0050 21001F @@@@        LODI  HL,VARBGN  IF NOT, GET ADDRESS
```

```
0053 CD6005            CALL  SUBDE      OF @(EXPR) AND PUT IT
0056 D1                POP   DE         IN HL
0057 C9                RET   U          C FLAG IS CLEARED
0058 FE1B        TV1   CMPI  27         NOT @, IS IT A TO Z?
005A 3F                CMC   .          IF NOT RETURN C FLAG
005B D8                RET   C
005C 13                INC   DE         IF A THROUGH Z
005D 21001F 0000       LODI  HL,VARBGN  COMPUTE ADDRESS OF
0060 07                ROT   L          THAT VARIABLE
0061 85                ADD   L          AND RETURN IT IN HL
0062 6F                LOD   L,A        WITH C FLAG CLEARED
0063 3E00              LODI  A,0
0065 8C                ADC   H
0066 67                LOD   H,A
0067 C9                RET   U
*
*                TSTC  XCH   HL,(SP)    *** TSTC OR RST 1 ***
                       IGNBLK           THIS IS AT LOC. 8
                       CMP   M          AND THEN JMP HERE
0068 23          TC1   INC   HL         COMPARE THE BYTE THAT
0069 CA7300            JMP   Z,TC2      FOLLOWS THE RST INST.
006C C5                PUSH  BC         WITH THE TEXT (DE->)
006D 4E                LOD   C,M        IF NOT =, ADD THE 2ND
006E 0600              LODI  B,0        BYTE THAT FOLLOWS THE
0070 09                ADD   HL,BC      RST TO THE OLD PC
0071 C1                POP   BC         I.E., DO A RELATIVE
0072 1B                DEC   DE         JUMP IF NOT =
0073 13          TC2   INC   DE         IF =, SKIP THOSE BYTES
0074 23                INC   HL         AND CONTINUE
0075 E3                XCH   HL,(SP)
0076 C9                RET   U
*
0077 210000      TSTNUM LODI  HL,0      *** TSTNUM ***
007A 44                LOD   B,H        TEST IF THE TEXT IS
007B EF                IGNBLK           A NUMBER
007C FE30        TN1   CMPI  '0'        IF NOT, RETURN 0 IN
007E D8                RET   C          B AND HL
007F FE3A              CMPI  X'3A'      IF NUMBERS, CONVERT
0081 D0                RET   NC         TO BINARY IN HL AND
0082 3EF0              LODI  A,X'F0'    SET B TO # OF DIGITS
0084 A4                AND   H          IF H>255, THERE IS NO
0085 C29F00            JMP   NZ,QHOW    ROOM FOR NEXT DIGIT
0088 04                INC   B          B COUNTS # OF DIGITS
0089 C5                PUSH  BC
008A 444D              LOD   BC,HL      HL=10*HL+(NEW DIGIT)
008C 29                ADD   HL,HL      WHERE 10* IS DONE BY
008D 29                ADD   HL,HL      SHIFT AND ADD
008E 09                ADD   HL,BC
008F 29                ADD   HL,HL
0090 1A                LD    A,(DE)     AND (DIGIT) IS FROM
0091 13                INC   DE         STRIPPING THE ASCII
0092 E60F              ANDI  X'0F'      CODE
0094 85                ADD   L
0095 6F                LOD   L,A
0096 3E00              LODI  A,0
0098 8C                ADC   H
0099 67                LOD   H,A
009A C1                POP   BC
009B 1A                LD    A,(DE)     DO THIS DIGIT AFTER
009C F27C00            JMP   NS,TN1     DIGIT. S SAYS OVERFLOW
009F D5          QHOW  PUSH  DE         *** ERROR: "HOW?" ***
00A0 11A600      AHOW  LODI  DE,HOW
00A3 C3A805            JMP   ERROR
00A6 484F573F    HOW   CHAR  'HOW?',@CR
00AA 0D
00AB 4F4B        OK    CHAR  'OK',@CR
00AD 0D
00AE 574841543F  WHAT  CHAR  'WHAT?',@CR
00B3 0D
00B4 534F525259  SORRY CHAR  'SORRY',@CR
00B9 0D
*
*****************************************************************
*
* *** MAIN ***
*
* THIS IS THE MAIN LOOP THAT COLLECTS THE TINY BASIC PROGRAM
* AND STORES IT IN THE MEMORY.
*
* AT START, IT PRINTS OUT "(CR)OK(CR)", AND INITIALIZES THE
* STACK AND SOME OTHER INTERNAL VARIABLES.  THEN IT PROMPTS
* ">" AND READS A LINE.  IF THE LINE STARTS WITH A NON-ZERO
* NUMBER, THIS NUMBER IS THE LINE NUMBER.  THE LINE NUMBER
* (IN 16 BIT BINARY) AND THE REST OF THE LINE (INCLUDING CR)
* IS STORED IN THE MEMORY.  IF A LINE WITH THE SAME LINE
* NUMBER IS ALREDY THERE, IT IS REPLACED BY THE NEW ONE.  IF
* THE REST OF THE LINE CONSISTS OF A CR ONLY, IT IS NOT STORED
* AND ANY EXISTING LINE WITH THE SAME LINE NUMBER IS DELETED.
*
* AFTER A LINE IS INSERTED, REPLACED, OR DELETED, THE PROGRAM
* LOOPS BACK AND ASK FOR ANOTHER LINE.  THIS LOOP WILL BE
* TERMINATED WHEN IT READS A LINE WITH ZERO OR NO LINE
* NUMBER; AND CONTROL IS TRANSFERED TO "DIRECT".
*
* TINY BASIC PROGRAM SAVE AREA STARTS AT THE MEMORY LOCATION
* LABELED "TXTBGN" AND ENDED AT "TXTEND".  WE ALWAYS FILL THIS
* AREA STARTING AT "TXTBGN", THE UNFILLED PORTION IS POINTED
* BY THE CONTENT OF A MEMORY LOCATION LABELED "TXTUNF".
*
* THE MEMORY LOCATION "CURRNT" POINTS TO THE LINE NUMBER
* THAT IS CURRENTLY BEING INTERPRETED.  WHILE WE ARE IN
* THIS LOOP OR WHILE WE ARE INTERPRETING A DIRECT COMMAND
* (SEE NEXT SECTION), "CURRNT" SHOULD POINT TO A 0.
*
*                START LODI  SP,STACK   THIS IS AT LOC. 0
00BA CD0E00      ST1   CALL  CRLF       AND JUMP TO HERE
00BD 11AB00            LODI  DE,OK      DE->STRING
00C0 97                SUB   A          A=0
00C1 CD3C06            CALL  PRTSTG     PRINT STRING UNTIL CR
00C4 21C800            LODI  HL,ST2+1   LITERAL 0
00C7 220108            ST    HL,CURRNT  CURRNT->LINE # = 0
00CA 210000      ST2   LODI  HL,0
00CD 220708            ST    HL,LOPVAR
00D0 220308            ST    HL,STKGOS
00D3 3E3E        ST3   LODI  A,'>'      PROMPT '>' AND
00D5 CDD605            CALL  GETLN      READ A LINE
00D8 D5                PUSH  DE         DE->END OF LINE
00D9 11371F 0000       LODI  DE,BUFFER  DE->BEGINNING OF LINE
00DC CD7700            CALL  TSTNUM     TEST IF IT IS A NUMBER
00DF EF                IGNBLK
00E0 7C                LOD   A,H        HL=VALUE OF THE # OR
00E1 B5                IOR   L          0 IF NO # WAS FOUND
00E2 C1                POP   BC         BC->END OF LINE
00E3 CAF501            JMP   Z,DIRECT
00E6 1B                DEC   DE         BACKUP DE AND SAVE
00E7 7C                LOD   A,H        VALUE OF LINE # THERE
00E8 12                ST    A,(DE)
00E9 1B                DEC   DE
00EA 7D                LOD   A,L
00EB 12                ST    A,(DE)
00EC C5D5              PUSH  BC,DE      BC,DE->BEGIN, END
00EE 79                LOD   A,C
00EF 93                SUB   E
00F0 F5                PUSH  AF         A=# OF BYTES IN LINE
00F1 CD1406            CALL  FNDLN      FIND THIS LINE IN SAVE
00F4 D5                PUSH  DE         AREA, DE->SAVE AREA
00F5 C20801            JMP   NZ,ST4     NZ:NOT FOUND, INSERT
00F8 D5                PUSH  DE         Z:FOUND, DELETE IT
00F9 CD3006            CALL  FNDNXT     FIND NEXT LINE
*                                       DE->NEXT LINE
00FC C1                POP   BC         BC->LINE TO BE DELETED
00FD 2A1308            LD    HL,TXTUNF  HL->UNFILLED SAVE AREA
0100 CD8D06            CALL  MVUP       MOVE UP TO DELETE
0103 6069              LOD   HL,BC      TXTUNF->UNFILLED AREA
0105 221308            ST    HL,TXTUNF  UPDATE
0108 C1          ST4   POP   BC         GET READY TO INSERT
0109 2A1308            LD    HL,TXTUNF  BUT FIRST CHECK IF
010C F1                POP   AF         THE LENGTH OF NEW LINE
010D E5                PUSH  HL         IS 3 (LINE # AND CR)
010E FE03              CMPI  3          THEN DO NOT INSERT
0110 CA0000            JMP   Z,START    MUST CLEAR THE STACK
0113 85                ADD   L          COMPUTE NEW TXTUNF
0114 6F                LOD   L,A
0115 3E00              LODI  A,0
0117 8C                ADC   H
```

```
0118 67                 LOD  H,A        HL->NEW UNFILLED AREA
0119 11001F 0000        LODI DE,TXTEND  CHECK TO SEE IF THERE
011C E7                 COMP            IS ENOUGH SPACE
011D D2CF05             JMP  NC,QSORRY  SORRY, NO ROOM FOR IT
0120 221308             ST   HL,TXTUNF  OK, UPDATE TXTUNF
0123 D1                 POP  DE         DE->OLD UNFILLED AREA
0124 CDC606             CALL MVDOWN
0127 D1E1               POP  DE,HL      DE->BEGIN, HL->END
0129 CDBD06             CALL MVUP       MOVE NEW LINE TO SAVE
012C C3D300             JMP  ST3        AREA
*
*****************************************************************
*
* *** TABLES *** DIRECT *** & EXEC ***
*
* THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
* WHEN A MATCH IS FOUND, CONTROL IS TRANSFERED TO THE SECTION
* OF CODE ACCORDING TO THE TABLE.
*
* AT 'EXEC', DE SHOULD POINT TO THE STRING AND HL SHOULD POINT
* TO THE TABLE-1.  AT 'DIRECT', DE SHOULD POINT TO THE STRING,
* HL WILL BE SET UP TO POINT TO TAB1-1, WHICH IS THE TABLE OF
* ALL DIRECT AND STATEMENT COMMANDS.
*
* A '.' IN THE STRING WILL TERMINATE THE TEST AND THE PARTIAL
* MATCH WILL BE CONSIDERED AS A MATCH.  E.G., 'P.', 'PR.',
* 'PRI.', 'PRIN.', OR 'PRINT' WILL ALL MATCH 'PRINT'.
*
* THE TABLE CONSISTS OF ANY NUMBER OF ITEMS.  EACH ITEM
* IS A STRING OF CHARACTERS WITH BIT 7 SET TO 0 AND
* A JUMP ADDRESS STORED HI-LOW WITH BIT 7 OF THE HIGH
* BYTE SET TO 1.
*
* END OF TABLE IS AN ITEM WITH A JUMP ADDRESS ONLY.  IF THE
* STRING DOES NOT MATCH ANY OF THE OTHER ITEMS, IT WILL
* MATCH THIS NULL ITEM AS DEFAULT.
*
012F                TAB1  EQU  *          DIRECT COMMANDS
012F 4C495354               ITEM 'LIST',LIST
0133 8261
0135 52554E                 ITEM 'RUN',RUN
0138 8233
013A 4E4557                 ITEM 'NEW',NEW
013D 8226
013F                TAB2  EQU  *          DIRECT/STATEMENT
013F 4E455854               ITEM 'NEXT',NEXT
0143 8349
0145 4C4554                 ITEM 'LET',LET
0148 8407
014A 4946                   ITEM 'IF',IF
014C 839A
014E 474F544F               ITEM 'GOTO',GOTO
0152 8252
0154 474F535542             ITEM 'GOSUB',GOSUB
0159 8281
0158 52455455524E           ITEM 'RETURN',RETURN
0161 82D1
0163 52454D                 ITEM 'REM',REM
0166 8396
0168 464F52                 ITEM 'FOR',FOR
016B 82EA
016D 494E505554             ITEM 'INPUT',INPUT
0172 8381
0174 5052494E54             ITEM 'PRINT',PRINT
0179 8279
017B 53544F50               ITEM 'STOP',STOP
017F 822F
0181 8401                   ITEM ,DEFLT
0183 594F55204D415920       CHAR 'YOU MAY INSERT  MORE COMMANDS.'
0188 494E534552542020
0193 4D4F524520434F4D
019B 4D414E44532E
01A1                TAB4  EQU  *          FUNCTIONS
01A1 524E44                 ITEM 'RND',RND
01A4 8506
01A6 414253                 ITEM 'ABS',ABS
01A9 8531
01AB 53495A45               ITEM 'SIZE',SIZE
01AF 853D
01B1 84EC                   ITEM ,XP40
01B3 594F55204D415920       CHAR 'YOU MAY INSERT  MORE FUNCTIONS'
01BB 494E534552542020
01C3 4D4F524520466554E
01CB 4354494F4E53
01D1                TAB5  EQU  *          "TO" IN "FOR"
01D1 544F                   ITEM 'TO',FR1
01D3 82FA
01D5 85A4                   ITEM ,QWHAT
01D7                TAB6  EQU  *          "STEP" IN "FOR"
01D7 53544550               ITEM 'STEP',FR2
01DB 8304
01DD 8308                   ITEM ,FR3
01DF                TAB8  EQU  *          RELATION OPERATORS
01DF 3E3D                   ITEM '>=',XP11
01E1 8417
01E3 23                     ITEM '#',XP12
01E4 841D
01E6 3E                     ITEM '>',XP13
01E7 8423
01E9 3D                     ITEM '=',XP15
01EA 8432
01EC 3C3D                   ITEM '<=',XP14
01EE 842A
01F0 3C                     ITEM '<',XP16
01F1 8438
01F3 843E                   ITEM ,XP17
*
01F5 212E01         DIRECT  LODI HL,TAB1-1 *** DIRECT ***
*
01F8                EXEC    EQU  *          *** EXEC ***
01F8 EF             EX0     IGNBLK          IGNORE LEADING BLANKS
01F9 D5                     PUSH DE         SAVE POINTER
01FA 1A             EX1     LD   A,(DE)     IF FOUND '.' IN STRING
01FB 13                     INC  DE         BEFORE ANY MISMATCH
01FC FE2E                   CMPI '.'        WE DECLARE A MATCH
01FE CA1702                 JMP  Z,EX3
0201 23                     INC  HL         HL->TABLE
0202 BE                     CMP  M          IF MATCH, TEST NEXT
0203 CAFA01                 JMP  Z,EX1
0206 3E7F                   LODI A,X'7F'    ELSE, SEE IF BIT 7
0208 1B                     DEC  DE         OF TABLE IS SET, WHICH
0209 BE                     CMP  M          IS THE JUMP ADDR. (HI)
020A DA1E02                 JMP  C,EX5      C:YES, MATCHED
020D 23             EX2     INC  HL         NC:NO, FIND JUMP ADDR.
020E BE                     CMP  M
020F D20D02                 JMP  NC,EX2
0212 23                     INC  HL         BUMP TO NEXT TAB, ITEM
0213 D1                     POP  DE         RESTORE STRING POINTER
0214 C3F801                 JMP  EX0        TEST AGAINST NEXT ITEM
0217 3E7F           EX3     LODI A,X'7F'    PARTIAL MATCH, FIND
0219 23             EX4     INC  HL         JUMP ADDR., WHICH IS
021A BE                     CMP  M          FLAGGED BY BIT 7
021B D21902                 JMP  NC,EX4
021E 7E             EX5     LOD  A,M        LOAD HL WITH THE JUMP
021F 23                     INC  HL         ADDRESS FROM THE TABLE
0220 6E                     LOD  L,M
0221 E67F                   ANDI X'7F'      MASK OFF BIT 7
0223 67                     LOD  H,A
0224 F1                     POP  AF         CLEAN UP THE GABAGE
0225 E9                     JMP  (HL)       AND WE GO DO IT
*
*****************************************************************
*
* WHAT FOLLOWS IS THE CODE TO EXECUTE DIRECT AND STATEMENT
* COMMANDS.  CONTROL IS TRANSFERED TO THESE POINTS VIA THE
* COMMAND TABLE LOOKUP CODE OF 'DIRECT' AND 'EXEC' IN LAST
* SECTION.  AFTER THE COMMAND IS EXECUTED, CONTROL IS
* TRANSFERED TO OTHER SECTIONS AS FOLLOWS:
*
* FOR 'LIST', 'NEW', AND 'STOP': GO BACK TO 'START'
* FOR 'RUN': GO EXECUTE THE FIRST STORED LINE IF ANY; ELSE
* GO BACK TO 'START'.
* FOR 'GOTO' AND 'GOSUB': GO EXECUTE THE TARGET LINE.
* FOR 'RETURN' AND 'NEXT': GO BACK TO SAVED RETURN LINE.
* FOR ALL OTHERS: IF 'CURRNT' -> 0, GO TO 'START', ELSE
```

```
* GO EXECUTE NEXT COMMAND.  (THIS IS DONE IN 'FINISH'.)
*
*****************************************************************
*
* *** NEW *** STOP *** RUN (& FRIENDS) *** & GOTO ***
*
* 'NEW(CR)' SETS 'TXTUNF' TO POINT TO 'TXTBGN'
*
* 'STOP(CR)' GOES BACK TO 'START'
*
* 'RUN(CR)' FINDS THE FIRST STORED LINE, STORE ITS ADDRESS (IN
* 'CURRNT'), AND START EXECUTE IT.  NOTE THAT ONLY THOSE
* COMMANDS IN TAB2 ARE LEGAL FOR STORED PROGRAM.
*
* THERE ARE 3 MORE ENTRIES IN 'RUN':
* 'RUNNXL' FINDS NEXT LINE, STORES ITS ADDR. AND EXECUTES IT.
* 'RUNTSL' STORES THE ADDRESS OF THIS LINE AND EXECUTES IT.
* 'RUNSML' CONTINUES THE EXECUTION ON SAME LINE.
*
* 'GOTO EXPR(CR)' EVALUATES THE EXPRESSION, FIND THE TARGET
* LINE, AND JUMP TO 'RUNTSL' TO DO IT.
*
0226 CDA005       NEW     CALL  ENDCHK     *** NEW(CR) ***
0229 211508               LODI  HL,TXTBGN
022C 221308               ST    HL,TXTUNF
*
022F CDA005       STOP    CALL  ENDCHK     *** STOP(CR) ***
0232 C7                   RSTART
*
0233 CDA005       RUN     CALL  ENDCHK     *** RUN(CR) ***
0236 111508               LODI  DE,TXTBGN  FIRST SAVED LINE
*
0239 210000       RUNNXL  LODI  HL,0       *** RUNNXL ***
023C CD1C06               CALL  FNDLNP     FIND WHATEVER LINE #
023F DA0000               JMP   C,START    C:PASSED TXTUNF, QUIT
*
0242 EB           RUNTSL  XCH   HL,DE      *** RUNTSL ***
0243 220108               ST    HL,CURRNT  SET 'CURRNT'->LINE #
0246 EB                   XCH   HL,DE
0247 1313                 INC   DE,2       BUMP PASS LINE #
*
0249 CD3207       RUNSML  CALL  CHKIO      *** RUNSML ***
024C 213E01               LODI  HL,TAB2-1  FIND COMMAND IN TAB2
024F C3F801               JMP       EXEC   AND EXECUTE IT
*
0252 DF           GOTO    EXPR             *** GOTO EXPR ***
0253 D5                   PUSH  DE         SAVE FOR ERROR ROUTINE
0254 CDA005               CALL  ENDCHK     MUST FIND A CR
0257 CD1406               CALL  FNDLN      FIND THE TARGET LINE
025A C2A000               JMP   NZ,AHOW    NO SUCH LINE #
025D F1                   POP   AF         CLEAR THE PUSH DE
025E C34202               JMP   RUNTSL     GO DO IT
*
*****************************************************************
*
* *** LIST *** & PRINT ***
*
* LIST HAS TWO FORMS:
* 'LIST(CR)' LISTS ALL SAVED LINES
* 'LIST #(CR)' START LIST AT THIS LINE #
* YOU CAN STOP THE LISTING BY CONTROL C KEY
*
* PRINT COMMAND IS 'PRINT ....;' OR 'PRINT ....(CR)'
* WHERE '....' IS A LIST OF EXPRESIONS, FORMATS, BACK-
* ARROWS, AND STRINGS.  THESE ITEMS ARE SEPERATED BY COMMAS.
*
* A FORMAT IS A POUND SIGN FOLLOWED BY A NUMBER.  IT CONTROLS
* THE NUMBER OF SPACES THE VALUE OF A EXPRESION IS GOING TO
* BE PRINTED.  IT STAYS EFFECTIVE FOR THE REST OF THE PRINT
* COMMAND UNLESS CHANGED BY ANOTHER FORMAT.  IF NO FORMAT IS
* SPECIFIED, 6 POSITIONS WILL BE USED.
*
* A STRING IS QUOTED IN A PAIR OF SINGLE QUOTES OR A PAIR OF
* DOUBLE QUOTES.
*
* A BACK-ARROW MEANS GENERATE A (CR) WITHOUT (LF)
*
* A (CRLF) IS GENERATED AFTER THE ENTIRE LIST HAS BEEN
* PRINTED OR IF THE LIST IS A NULL LIST.  HOWEVER IF THE LIST
* ENDED WITH A COMMA, NO (CRLF) IS GENERATED.
*
0261 CD7700       LIST    CALL  TSTNUM     TEST IF THERE IS A #
0264 CDAC05               CALL  ENDCHK     IF NO # WE GET A 0
0267 CD1406               CALL  FNDLN      FIND THIS OR NEXT LINE
026A DA0000       LS1     JMP   C,START    C:PASSED TXTUNF
026D CDAA06               CALL  PRTLN      PRINT THE LINE
0270 CD3207               CALL  CHKIO      STOP IF HIT CONTROL-C
0273 CD1C06               CALL  FNDLNP     FIND NEXT LINE
0276 C36A02               JMP   LS1        AND LOOP BACK
*
0279 0E06         PRINT   LODI  C,6        C = # OF SPACES
027B CF                   TSTC  ';',PR2    IF NULL LIST & ";"
027C 3B
027D 06
027E CD0E00               CALL  CRLF       GIVE CR-LF AND
0281 C34902               JMP   RUNSML     CONTINUE SAME LINE
0284 CF           PR2     TSTC  @CR,PR0    IF NULL LIST (CR)
0285 0D
0286 06
0287 CD0E00               CALL  CRLF       ALSO GIVE CR-LF AND
028A C33902               JMP   RUNNXL     GO TO NEXT LINE
028D CF           PR0     TSTC  '#',PR1    ELSE IS IT FORMAT?
028E 23
028F 05
0290 DF                   EXPR             YES, EVALUATE EXPR.
0291 4D                   LOD   C,L        AND SAVE IT IN C
0292 C39B02               JMP   PR3        LOOK FOR MORE TO PRINT
0295 CD4806       PR1     CALL  QTSTG      OR IS IT A STRING?
0298 C3A802               JMP   PR8        IF NOT, MUST BE EXPR.
029B CF           PR3     TSTC  ',',PR6    IF ",", GO FIND NEXT
029C 2C
029D 06
029E CD9105               CALL  FIN        IN THE LIST.
02A1 C38D02               JMP   PR0        LIST CONTINUES
02A4 CD0E00       PR6     CALL  CRLF       LIST ENDS
02A7 F7                   FINISH
02A8 DF           PR8     EXPR             EVALUATE THE EXPR
02A9 C5                   PUSH  BC
02AA CD6E06               CALL  PRTNUM     PRINT THE VALUE
02AD C1                   POP   BC
02AE C39B02               JMP   PR3        MORE TO PRINT?
*
*****************************************************************
*
* *** GOSUB *** & RETURN ***
*
* 'GOSUB EXPR;' OR 'GOSUB EXPR (CR)' IS LIKE THE 'GOTO'
* COMMAND, EXCEPT THAT THE CURRENT TEXT POINTER, STACK POINTER
* ETC. ARE SAVE SO THAT EXECUTION CAN BE CONTINUED AFTER THE
* SUBROUTINE 'RETURN'.  IN ORDER THAT 'GOSUB' CAN BE NESTED
* (AND EVEN RECURSIVE), THE SAVE AREA MUST BE STACKED.
* THE STACK POINTER IS SAVED IN 'STKGOS'. THE OLD 'STKGOS' IS
* SAVED IN THE STACK.  IF WE ARE IN THE MAIN ROUTINE, 'STKGOS'
* IS ZERO (THIS WAS DONE BY THE "MAIN" SECTION OF THE CODE),
* BUT WE STILL SAVE IT AS A FLAG FOR NO FURTHER 'RETURN'S.
*
* 'RETURN(CR)' UNDOS EVERYTHING THAT 'GOSUB' DID, AND THUS
* RETURN THE EXCUTION TO THE COMMAND AFTER THE MOST RECENT
* 'GOSUB'.  IF 'STKGOS' IS ZERO, IT INDICATES THAT WE
* NEVER HAD A 'GOSUB' AND IS THUS AN ERROR.
*
02B1 CDF106       GOSUB   CALL  PUSHA      SAVE THE CURRENT "FOR"
02B4 DF                   EXPR             PARAMETERS
02B5 D5                   PUSH  DE         AND TEXT POINTER
02B6 CD1406               CALL  FNDLN      FIND THE TARGET LINE
02B9 C2A000               JMP   NZ,AHOW    NOT THERE. SAY "HOW?"
02BC 2A0108               LD    HL,CURRNT  FOUND IT, SAVE OLD
02BF E5                   PUSH  HL         'CURRNT' OLD 'STKGOS'
02C0 2A0308               LD    HL,STKGOS
02C3 E5                   PUSH  HL
02C4 210000               LODI  HL,0       AND LOAD NEW ONES
02C7 220708               ST    HL,LOPVAR
02CA 39                   ADD   HL,SP
02CB 220308               ST    HL,STKGOS
02CE C34202               JMP   RUNTSL     THEN RUN THAT LINE
02D1 CDA005       RETURN  CALL  ENDCHK     THERE MUST BE A CR
```

```
02D4 2A0308             LD   HL,STKGOS OLD STACK POINTER
02D7 7C                 LOD  A,H       0 MEANS NOT EXIST
02D8 B5                 IOR  L
02D9 CAA405             JMP  Z,QWHAT   SO, WE SAY: "WHAT?"
02DC F9                 LOD  SP,HL     ELSE, RESTORE IT
02DD E1                 POP  HL
02DE 220308             ST   HL,STKGOS AND THE OLD 'STKGOS'
02E1 E1                 POP  HL
02E2 220108             ST   HL,CURRNT AND THE OLD 'CURRNT'
02E5 D1                 POP  DE        OLD TEXT POINTER
02E6 CDD506             CALL POPA      OLD "FOR" PARAMETERS
02E9 F7                 FINISH         AND WE ARE BACK HOME
*
*****************************************************************
*
* *** FOR *** & NEXT ***
*
* 'FOR' HAS TWO FORMS:
* 'FOR VAR=EXP1 TO EXP2 STEP EXP1' AND 'FOR VAR=EXP1 TO EXP2'
* THE SECOND FORM MEANS THE SAME THING AS THE FIRST FORM WITH
* EXP1=1.  (I.E., WITH A STEP OF +1.)
* TBI WILL FIND THE VARIABLE VAR. AND SET ITS VALUE TO THE
* CURRENT VALUE OF EXP1.  IT ALSO EVALUATES EXPR2 AND EXP1
* AND SAVE ALL THESE TOGETHER WITH THE TEXT POINTER ETC. IN
* THE 'FOR' SAVE AREA, WHICH CONSISTS OF 'LOPVAR', 'LOPINC',
* 'LOPLMT', 'LOPLN', AND 'LOPPT'.  IF THERE IS ALREADY SOME-
* THING IN THE SAVE AREA (THIS IS INDICATED BY A NON-ZERO
* 'LOPVAR'), THEN THE OLD SAVE AREA IS SAVED IN THE STACK
* BEFORE THE NEW ONE OVERWRITES IT.
* TBI WILL THEN DIG IN THE STACK AND FIND OUT IF THIS SAME
* VARIABLE WAS USED IN ANOTHER CURRENTLY ACTIVE 'FOR' LOOP.
* IF THAT IS THE CASE, THEN THE OLD 'FOR' LOOP IS DEACTIVATED.
* (PURGED FROM THE STACK..)
*
* 'NEXT VAR' SERVES AS THE LOGICAL (NOT NECESSARILLY PHYSICAL)
* END OF THE 'FOR' LOOP.  THE CONTROL VARIABLE VAR. IS CHECKED
* WITH THE 'LOPVAR'.  IF THEY ARE NOT THE SAME, TBI DIGS IN
* THE STACK TO FIND THE RIGHT ONE AND PURGES ALL THOSE THAT
* DID NOT MATCH.  EITHER WAY, TBI THEN ADDS THE 'STEP' TO
* THAT VARIABLE AND CHECK THE RESULT WITH THE LIMIT.  IF IT
* IS WITHIN THE LIMIT, CONTROL LOOPS BACK TO THE COMMAND
* FOLLOWING THE 'FOR'.  IF OUTSIDE THE LIMIT, THE SAVE ARER
* IS PURGED AND EXECUTION CONTINUES.
*
02EA CDF106     FOR     CALL PUSHA     SAVE THE OLD SAVE AREA
02ED CD7E05             CALL SETVAL    SET THE CONTROL VAR.
02F0 28                 DEC  HL        HL IS ITS ADDRESS
02F1 220708             ST   HL,LOPVAR SAVE THAT
02F4 21D001             LODI HL,TAB5-1 USE 'EXEC' TO LOOK
02F7 C3F801             JMP  EXEC      FOR THE WORD 'TO'
02FA DF         FR1     EXPR           EVALUATE THE LIMIT
02FB 220B08             ST   HL,LOPLMT SAVE THAT
02FE 21D601             LODI HL,TAB6-1 USE 'EXEC' TO LOOK
0301 C3F801             JMP  EXEC      FOR THE WORD 'STEP'
0304 DF         FR2     EXPR           FOUND IT, GET STEP
0305 C30803             JMP  FR4
0308 210100     FR3     LODI HL,1      NOT FOUND, SET TO 1
030B 220908     FR4     ST   HL,LOPINC SAVE THAT TOO
030E 2A0108     FR5     LD   HL,CURRNT SAVE CURRENT LINE #
0311 220D08             ST   HL,LOPLN
0314 EB                 XCH  HL,DE     AND TEXT POINTER
0315 220F08             ST   HL,LOPPT
0318 010A00             LODI BC,10     DIG INTO STACK TO
0318 2A0708             LD   HL,LOPVAR FIND 'LOPVAR'
031E EB                 XCH  HL,DE
031F 60                 LOD  H,B
0320 68                 LOD  L,B       HL=0 NOW
0321 39                 ADD  HL,SP     HERE IS THE STACK
0322 3E                 SKIP
0323 09         FR7     ADD  HL,BC     EACH LEVEL IS 10 DEEP
0324 7E                 LOD  A,M       GET THAT OLD 'LOPVAR'
0325 23                 INC  HL
0326 86                 IOR  M
0327 CA4403             JMP  Z,FR8     0 SAYS NO MORE IN IT
032A 7E                 LOD  A,M
0328 28                 DEC  HL
032C BA                 CMP  D         SAME AS THIS ONE?
032D C22303             JMP  NZ,FR7
0330 7E                 LOD  A,M       THE OTHER HALF?
0331 BB                 CMP  E
0332 C22303             JMP  NZ,FR7
0335 EB                 XCH  HL,DE     YES, FOUND ONE
0336 210000             LODI HL,0
0339 39                 ADD  HL,SP     TRY TO MOVE SP
033A 444D               LOD  BC,HL
033C 210A00             LODI HL,10
033F 19                 ADD  HL,DE
0340 CDC606             CALL MVDOWN    AND PURGE 10 WORDS
0343 F9                 LOD  SP,HL     IN THE STACK
0344 2A0F08     FR8     LD   HL,LOPPT  JOB DONE, RESTORE DE
0347 EB                 XCH  HL,DE
0348 F7                 FINISH         AND CONTINUE
*
0349 FF         NEXT    TSTV           GET ADDRESS OF VAR.
034A DAA405             JMP  C,QWHAT   NO VARIABLE, "WHAT?"
034D 220508             ST   HL,VARNXT YES, SAVE IT
0350 D5         NX0     PUSH DE        SAVE TEXT POINTER
0351 EB                 XCH  HL,DE
0352 2A0708             LD   HL,LOPVAR GET VAR. IN 'FOR'
0355 7C                 LOD  A,H
0356 B5                 IOR  L         0 SAYS NEVER HAD ONE
0357 CAA505             JMP  Z,AWHAT   SO WE ASK: "WHAT?"
035A E7                 COMP           ELSE WE CHECK THEM
035B CA6803             JMP  Z,NX3     OK, THEY AGREE
035E D1                 POP  DE        NO, LET'S SEE
035F CDD506             CALL POPA      PURGE CURRENT LOOP
0362 2A0508             LD   HL,VARNXT AND POP ONE LEVEL
0365 C35003             JMP  NX0       GO CHECK AGAIN
0368 5E         NX3     LOD  E,M       COME HERE WHEN AGREED
0369 23                 INC  HL
036A 56                 LOD  D,M       DE=VALUE OF VAR.
036B 2A0908             LD   HL,LOPINC
036E E5                 PUSH HL
036F 19                 ADD  HL,DE     ADD ONE STEP
0370 EB                 XCH  HL,DE
0371 2A0708             LD   HL,LOPVAR PUT IT BACK
0374 73                 LOD  M,E
0375 23                 INC  HL
0376 72                 LOD  M,D
0377 2A0B08             LD   HL,LOPLMT HL->LIMIT
037A F1                 POP  AF        OLD HL
037B B7                 IOR  A
037C F28003             JMP  NS,NX1    STEP > 0
037F EB                 XCH  HL,DE     STEP < 0
0380 CD7605     NX1     CALL CKHLDE    COMPARE WITH LIMIT
0383 D1                 POP  DE        RESTORE TEXT POINTER
0384 DA9203             JMP  C,NX2     OUTSIDE LIMIT
0387 2A0D08             LD   HL,LOPLN  WITHIN LIMIT, GO
038A 220108             ST   HL,CURRNT BACK TO THE SAVED
038D 2A0F08             LD   HL,LOPPT  'CURRNT' AND TEXT
0390 EB                 XCH  HL,DE     POINTER
0391 F7                 FINISH
0392 CDD506     NX2     CALL POPA      PURGE THIS LOOP
0395 F7                 FINISH
*
*****************************************************************
*
* *** REM *** IF *** INPUT *** & LET (& DEFLT) ***
*
* 'REM' CAN BE FOLLOWED BY ANYTHING AND IS IGNORED BY TBI.
* TBI TREATS IT LIKE AN 'IF' WITH A FALSE CONDITION.
*
* 'IF' IS FOLLOWED BY AN EXPR. AS A CONDITION AND ONE OR MORE
* COMMANDS (INCLUDING OUTHER 'IF'S) SEPERATED BY SEMI-COLONS.
* NOTE THAT THE WORD 'THEN' IS NOT USED.  TBI EVALUATES THE
* EXPR. IF IT IS NON-ZERO, EXECUTION CONTINUES.  IF THE
* EXPR. IS ZERO, THE COMMANDS THAT FOLLOWS ARE IGNORED AND
* EXECUTION CONTINUES AT THE NEXT LINE.
*
* 'INPUT' COMMAND IS LIKE THE 'PRINT' COMMAND, AND IS FOLLOWED
* BY A LIST OF ITEMS.  IF THE ITEM IS A STRING IN SINGLE OR
* DOUBLE QUOTES, OR IS A BACK-ARROW, IT HAS THE SAME EFFECT AS
* IN 'PRINT'.  IF AN ITEM IS A VARIABLE, THIS VARIABLE NAME IS
* PRINTED OUT FOLLOWED BY A COLON.  THEN TBI WAITS FOR AN
* EXPR. TO BE TYPED IN.  THE VARIABLE IS THEN SET TO THE
* VALUE OF THIS EXPR.  IF THE VARIABLE IS PROCEDED BY A STRING
```

```
* (AGAIN IN SINGLE OR DOUBLE QUOTES). THE STRING WILL BE
* PRINTED FOLLOWED BY A COLON.  TBI THEN WAITS FOR INPUT EXPR.
* AND SET THE VARIABLE TO THE VALUE OF THE EXPR.
*
* IF THE INPUT EXPR. IS INVALID, TBI WILL PRINT "WHAT?",
* "HOW?" OR "SORRY" AND REPRINT THE PROMPT AND REDO THE INPUT.
* THE EXECUTION WILL NOT TERMINATE UNLESS YOU TYPE CONTROL-C.
* THIS IS HANDLED IN 'INPERR'.
*
* 'LET' IS FOLLOWED BY A LIST OF ITEMS SEPERATED BY COMMAS.
* EACH ITEM CONSISTS OF A VARIABLE, AN EQUAL SIGN, AND AN EXPR.
* TBI EVALUATES THE EXPR. AND SET THE VARIBLE TO THAT VALUE.
* TBI WILL ALSO HANDLE 'LET' COMMAND WITHOUT THE WORD 'LET'.
* THIS IS DONE BY 'DEFLT'.
*
0396 210000        REM    LODI HL,0        *** REM ***
0399 3E                   SKIP             THIS IS LIKE 'IF 0'
*
039A DF            IF     EXPR             *** IF ***
039B 7C                   LOD  A,H         IS THE EXPR.=0?
039C B5                   IOR  L
039D C24902               JMP  NZ,RUNSML   NO, CONTINUE
03A0 CD3206               CALL FNDSKP      YES, SKIP REST OF LINE
03A3 D24202               JMP  NC,RUNTSL   AND RUN THE NEXT LINE
03A6 C7                   RSTART           IF NO NEXT, RE-START
*
03A7 2A0508        INPERR LD   HL,STKINP   *** INPERR ***
03AA F9                   LOD  SP,HL       RESTORE OLD SP
03AB E1                   POP  HL          AND OLD 'CURRNT'
03AC 220108               ST   HL,CURRNT
03AF D1                   POP  DE          AND OLD TEXT POINTER
0380 D1                   POP  DE          REDO INPUT
*
03B1               INPUT  EQU  *           *** INPUT ***
03B1 D5            IP1    PUSH DE          SAVE IN CASE OF ERROR
03B2 CD4806               CALL QTSTG       IS NEXT ITEM A STRING?
03B5 C3BF03               JMP  IP2         NO
0388 FF                   TSTV             YES. BUT FOLLOWED BY A
0389 DAF903               JMP  C,IP4       VARIABLE?  NO.
03BC C3CF03               JMP  IP3         YES.  INPUT VARIABLE
03BF D5            IP2    PUSH DE          SAVE FOR 'PRTSTG'
03C0 FF                   TSTV             MUST BE VARIABLE NOW
03C1 DAA405               JMP  C,QWHAT     "WHAT?" IT IS NOT?
03C4 1A                   LD   A,(DE)      GET READY FOR 'PRTSTG'
03C5 4F                   LOD  C,A
03C6 97                   SUB  A
03C7 12                   ST   A,(DE)
03C8 D1                   POP  DE
03C9 CD3C06               CALL PRTSTG      PRINT STRING AS PROMPT
03CC 79                   LOD  A,C         RESTORE TEXT
03CD 1B                   DEC  DE
03CE 12                   ST   A,(DE)
03CF D5            IP3    PUSH DE          SAVE IN CASE OF ERROR
03D0 EB                   XCH  HL,DE
03D1 2A0108               LD   HL,CURRNT   ALSO SAVE 'CURRNT'
03D4 E5                   PUSH HL
03D5 218103               LODI HL,IP1      A NEGATIVE NUMBER
03D8 220108               ST   HL,CURRNT   AS A FLAG
03DB 210000               LODI HL,0        SAVE SP TOO
03DE 39                   ADD  HL,SP
03DF 220508               ST   HL,STKINP
03E2 D5                   PUSH DE          OLD HL
03E3 3E3A                 LODI A,':'       PRINT THIS TOO
03E5 CDD605               CALL GETLN       AND GET A LINE
03E8 11371F 0000          LODI DE,BUFFER   POINTS TO BUFFER
03EB DF                   EXPR             EVALUATE INPUT
03EC 000000               NOP  3           CAN BE 'CALL ENDCHK'
03EF D1                   POP  DE          OK, GET OLD HL
03F0 EB                   XCH  HL,DE
03F1 73                   LOD  M,E         SAVE VALUE IN VAR.
03F2 23                   INC  HL
03F3 72                   LOD  M,D
03F4 E1                   POP  HL          GET OLD 'CURRNT'
03F5 220108               ST   HL,CURRNT
03F8 D1                   POP  DE          AND OLD TEXT POINTER
03F9 F1            IP4    POP  AF          PURGE JUNK IN STACK
03FA CF                   TSTC ',',IP5     IS NEXT CH. ','?
03FB 2C
03FC 03
03FD C3B103               JMP  IP1         YES, MORE ITEMS.
0400 F7            IP5    FINISH
*
0401 1A            DEFLT  LD   A,(DE)      *** DEFLT ***
0402 FE0D                 CMPI @CR         EMPTY LINE IS OK
0404 CA1004               JMP  Z,LT1       ELSE IT IS 'LET'
*
0407 CD7E05        LET    CALL SETVAL      *** LET ***
040A CF                   TSTC ',',LT1     SET VALUE TO VAR.
040B 2C
040C 03
040D C30704               JMP  LET         ITEM BY ITEM
0410 F7            LT1    FINISH           UNTIL FINISH
*
*****************************************************************
*
* *** EXPR ***
*
* 'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.
* <EXPR>::=<EXPR2>
*          (<EXPR2><REL.OP.><EXPR2>
* WHERE <REL.OP.> IS ONE OF THE OPERATORS IN TAB8 AND THE
* RESULT OF THESE OPERATIONS IS 1 IF TRUE AND 0 IF FALSE.
* <EXPR2>::=(+ OR -)<EXPR3>(+ OR -<EXPR3>)(....)
* WHERE () ARE OPTIONAL AND (....) ARE OPTIONAL REPEATS.
* <EXPR3>::=<EXPR4>(<* OR /><EXPR4>)(....)
* <EXPR4>::=<VARIABLE>
*           <FUNCTION>
*           (<EXPR>)
* <EXPR> IS RECURSIVE SO THAT VARIABLE '@' CAN HAVE AN <EXPR>
* AS INDEX, FUNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS, AND
* <EXPR4> CAN BE AN <EXPR> IN PARANTHESE.
*
*                   EXPR   CALL EXPR2       THIS IS AT LOC. 18
*                          PUSH HL          SAVE <EXPR2> VALUE
0411 21DE01        EXPR1  LODI HL,TAB8-1   LOOKUP REL.OP.
0414 C3F801               JMP  EXEC        GO DO IT
0417 CD4004        XP11   CALL XP18        REL.OP.">="
041A D8                   RET  C           NO. RETURN HL=0
041B 6F                   LOD  L,A         YES. RETURN HL=1
041C C9                   RET  U
041D CD4004        XP12   CALL XP18        REL.OP."#"
0420 C8                   RET  Z           FALSE, RETURN HL=0
0421 6F                   LOD  L,A         TRUE, RETURN HL=1
0422 C9                   RET  U
0423 CD4004        XP13   CALL XP18        REL.OP.">"
0426 C8                   RET  Z           FALSE
0427 D8                   RET  C           ALSO FALSE, HL=0
0428 6F                   LOD  L,A         TRUE, HL=1
0429 C9                   RET  U
042A CD4004        XP14   CALL XP18        REL.OP."<="
042D 6F                   LOD  L,A         SET HL=1
042E C8                   RET  Z           REL. TRUE, RETURN
042F D8                   RET  C
0430 6C                   LOD  L,H         ELSE SET HL=0
0431 C9                   RET  U
0432 CD4004        XP15   CALL XP18        REL.OP."="
0435 C0                   RET  NZ          FALSE, RETRUN HL=0
0436 6F                   LOD  L,A         ELSE SET HL=1
0437 C9                   RET  U
0438 CD4004        XP16   CALL XP18        REL.OP."<"
043B D0                   RET  NC          FALSE, RETURN HL=0
043C 6F                   LOD  L,A         ELSE SET HL=1
043D C9                   RET  U
043E E1            XP17   POP  HL          NOT REL.OP.
043F C9                   RET  U           RETURN HL=<EXPR2>
0440 79            XP18   LOD  A,C         SUBROUTINE FOR ALL
0441 E1C1                 POP  HL,BC       REL.OP.'S
0443 E5C5                 PUSH HL,BC       REVERSE TOP OF STACK
0445 4F                   LOD  C,A
0446 CD5504               CALL EXPR2       GET 2ND <EXPR2>
0449 EB                   XCH  HL,DE       VALUE IN DE NOW
044A E3                   XCH  HL,(SP)     1ST <EXPR2> IN HL
044B CD7605               CALL CKHLDE      COMPARE 1ST WITH 2ND
044E D1                   POP  DE          RESTORE TEXT POINTER
044F 210000               LODI HL,0        SET HL=0, A=1
0452 3E01                 LODI A,1
```

```
0454 C9                      RET  U
*
0455 CF            EXPR2     TSTC '-',XP21     NAGATIVE SIGN?
0456 2D
0457 06
0458 210000                  LODI HL,0         YES, FAKE '0-'
045B C37F04                  JMP  XP26         TREAT LIKE SUBTRACT
045E CF            XP21      TSTC '+',XP22     POSITIVE SIGN? IGNORE
045F 2B
0460 00
0461 CD8904        XP22      CALL EXPR3        1ST <EXPR3>
0464 CF            XP23      TSTC '+',XP25     ADD?
0465 2B
0466 15
0467 E5                      PUSH HL           YES, SAVE VALUE
0468 CD8904                  CALL EXPR3        GET 2ND <EXPR3>
046B EB            XP24      XCH  HL,DE        2ND IN DE
046C E3                      XCH  HL,(SP)      1ST IN HL
046D 7C                      LOD  A,H          COMPARE SIGN
046E AA                      XOR  D
046F 7A                      LOD  A,D
0470 19                      ADD  HL,DE
0471 D1                      POP  DE           RESTORE TEXT POINTER
0472 FA6404                  JMP  S,XP23       1ST 2ND SIGN DIFFER
0475 AC                      XOR  H            1ST 2ND SIGN EQUAL
0476 F26404                  JMP  NS,XP23      SO IS RESULT
0479 C39F00                  JMP  QHOW         ELSE WE HAVE OVERFLOW
047C CF            XP25      TSTC '-',XP42     SUBTRACT?
047D 2D
047E 83
047F E5            XP26      PUSH HL           YES, SAVE 1ST <EXPR3>
0480 CD8904                  CALL EXPR3        GET 2ND <EXPR3>
0483 CD6A05                  CALL CHGSGN       NEGATE
0486 C36B04                  JMP  XP24         AND ADD THEM
*
0489 CDE604        EXPR3     CALL EXPR4        GET 1ST <EXPR4>
048C CF            XP31      TSTC '*',XP34     MULTIPLY?
048D 2A
048E 2C
048F E5                      PUSH HL           YES, SAVE 1ST
0490 CDE604                  CALL EXPR4        AND GET 2ND <EXPR4>
0493 0600                    LODI B,0          CLEAR B FOR SIGN
0495 CD6705                  CALL CHKSGN       CHECK SIGN
0498 EB                      XCH  HL,DE        2ND IN DE NOW
0499 E3                      XCH  HL,(SP)      1ST IN HL
049A CD6705                  CALL CHKSGN       CHECK SIGN OF 1ST
049D 7C                      LOD  A,H          IS HL > 255 ?
049E B7                      IOR  A
049F CAA804                  JMP  Z,XP32       NO
04A2 7A                      LOD  A,D          YES, HOW ABOUT DE
04A3 B2                      IOR  D
04A4 EB                      XCH  HL,DE        PUT SMALLER IN HL
04A5 C2A000                  JMP  NZ,AHOW      ALSO >, WILL OVERFLOW
04A8 7D            XP32      LOD  A,L          THIS IS DUMB
04A9 210000                  LODI HL,0         CLEAR RESULT
04AC B7                      IOR  A            ADD AND COUNT
04AD CAD804                  JMP  Z,XP35
04B0 19            XP33      ADD  HL,DE
04B1 DAA000                  JMP  C,AHOW       OVERFLOW
04B4 3D                      DEC  A
04B5 C2B004                  JMP  NZ,XP33
04B8 C3D804                  JMP  XP35         FINISHED
04BB CF            XP34      TSTC '/',XP42     DIVIDE?
04BC 2F
04BD 44
04BE E5                      PUSH HL           YES, SAVE 1ST <EXPR4>
04BF CDE604                  CALL EXPR4        AND GET 2ND ONE
04C2 0600                    LODI B,0          CLEAR B FOR SIGN
04C4 CD6705                  CALL CHKSGN       CHECK SIGN OF 2ND
04C7 EB                      XCH  HL,DE        PUT 2ND IN DE
04C8 E3                      XCH  HL,(SP)      GET 1ST IN HL
04C9 CD6705                  CALL CHKSGN       CHECK SIGN OF 1ST
04CC 7A                      LOD  A,D          DIVIDE BY 0?
04CD B3                      IOR  E
04CE CAA000                  JMP  Z,AHOW       SAY "HOW?"
04D1 C5                      PUSH BC           ELSE SAVE SIGN
04D2 CD4A05                  CALL DIVIDE       USE SUBROUTINE
04D5 6069                    LOD  HL,BC        RESULT IN HL NOW
04D7 C1                      POP  BC           GET SIGN BACK
04D8 D1            XP35      POP  DE           AND TEXT POINTER
04D9 7C                      LOD  A,H          HL MUST BE +
04DA B7                      IOR  A
04DB FA9F00                  JMP  S,QHOW       ELSE IT IS OVERFLOW
04DE 78                      LOD  A,B
04DF B7                      IOR  A
04E0 FC6A05                  CALL S,CHGSGN     CHANGE SIGN IF NEEDED
04E3 C38C04                  JMP  XP31         LOOK FOR MORE TERMS
*
04E6 21A001        EXPR4     LODI HL,TAB4-1    FIND FUNCTION IN TAB4
04E9 C3F801                  JMP  EXEC         AND GO DO IT
04EC FF            XP40      TSTV              NO, NOT A FUNCTION
04ED DAF504                  JMP  C,XP41       NOR A VARIABLE
04F0 7E                      LOD  A,M          VARIABLE
04F1 23                      INC  HL
04F2 66                      LOD  H,M          VALUE IN HL
04F3 6F                      LOD  L,A
04F4 C9                      RET  U
04F5 CD7700        XP41      CALL TSTNUM       OR IS IT A NUMBER
04F8 78                      LOD  A,B          # OF DIGIT
04F9 B7                      IOR  A
04FA C0                      RET  NZ           OK
04FB CF            PARN      TSTC '(',XP43     NO DIGIT, MUST BE
04FC 28
04FD 05
04FE DF                      EXPR              "(EXPR)"
04FF CF                      TSTC ')',XP43
0500 29
0501 01
0502 C9            XP42      RET  U
0503 C3A405        XP43      JMP  QWHAT        ELSE SAY: "WHAT?"
*
0506 CDFB04        RND       CALL PARN         *** RND(EXPR) ***
0509 7C                      LOD  A,H          EXPR MUST BE +
050A B7                      IOR  A
050B FA9F00                  JMP  S,QHOW
050E B5                      IOR  L            AND NON-ZERO
050F CA9F00                  JMP  Z,QHOW
0512 D5E5                    PUSH DE,HL        SAVE BOTH
0514 2A1108                  LD   HL,RANPNT    GET MEMORY AS RANDOM
0517 11FF07                  LODI DE,LSTROM    NUMBER
051A E7                      COMP
051B DA2105                  JMP  C,RA1        WRAP AROUND IF LAST
051E 210000                  LODI HL,START
0521 5E            RA1       LOD  E,M
0522 23                      INC  HL
0523 56                      LOD  D,M
0524 221108                  ST   HL,RANPNT
0527 E1                      POP  HL
0528 EB                      XCH  HL,DE
0529 C5                      PUSH BC
052A CD4A05                  CALL DIVIDE       RND(N)=MOD(M,N)+1
052D C1D1                    POP  BC,DE
052F 23                      INC  HL
0530 C9                      RET  U
*
0531 CDFB04        ABS       CALL PARN         *** ABS(EXPR) ***
0534 CD6705                  CALL CHKSGN       CHECK SIGN
0537 7C                      LOD  A,H          NOTE THAT -32768
0538 B4                      IOR  H            CANNOT CHANGE SIGN
0539 FA9F00                  JMP  S,QHOW       SO SAY: "HOW?"
053C C9                      RET  U
053D 2A1308        SIZE      LD   HL,TXTUNF    *** SIZE ***
0540 D5                      PUSH DE           GET THE NUMBER OF FREE
0541 EB                      XCH  HL,DE        BYTES BETWEEN 'TXTUNF'
0542 21001F 0000             LODI HL,VARBGN    AND 'VARBGN'
0545 CD6005                  CALL SUBDE
0548 D1                      POP  DE
0549 C9                      RET  U
*
*****************************************************************
*
* *** DIVIDE *** SUBDE *** CHKSGN *** CHGSGN *** & CKHLDE ***
*
* 'DIVIDE' DIVIDES HL BY DE, RESULT IN BC, REMAINDER IN HL
*
* 'SUBDE' SUBTRACTS DE FROM HL
```

```
*
* 'CHKSGN' CHECKS SIGN OF HL.  IF +, NO CHANGE.  IF -, CHANGE
* SIGN AND FLIP SIGN OF B.
*
* 'CHGSGN' CHNGES SIGN OF HL AND B UNCONDITIONALLY.
*
* 'CKHLDE' CHECKS SIGN OF HL AND DE.  IF DIFFERENT, HL AND DE
* ARE INTERCHANGED.  IF SAME SIGN, NOT INTERCHANGED.  EITHER
* CASE, HL DE ARE THEN COMPARED TO SET THE FLAGS.
*
054A E5          DIVIDE PUSH HL          *** DIVIDE ***
054B 6C                 LOD  L,H         DIVIDE H BY DE
054C 2600               LODI H,0
054E CD5505             CALL DV1
0551 41                 LOD  B,C         SAVE RESULT IN B
0552 7D                 LOD  A,L         (REMAINDER+L)/DE
0553 E1                 POP  HL
0554 67                 LOD  H,A
0555 0EFF        DV1    LODI C,-1        RESULT IN C
0557 0C          DV2    INC  C           DUMB ROUTINE
0558 CD6005             CALL SUBDE       DIVIDE BY SUBTRACT
055B D25705             JMP  NC,DV2      AND COUNT
055E 19                 ADD  HL,DE
055F C9                 RET  U
*
0560 7D          SUBDE  LOD  A,L         *** SUBDE ***
0561 93                 SUB  E           SUBTRACT DE FROM
0562 6F                 LOD  L,A         HL
0563 7C                 LOD  A,H
0564 9A                 SBB  D
0565 67                 LOD  H,A
0566 C9                 RET  U
*
0567 7C          CHKSGN LOD  A,H         *** CHKSGN ***
0568 B7                 IOR  A           CHECK SIGN OF HL
0569 F0                 RET  NS          IF -, CHANGE SIGN
*
056A 7C          CHGSGN LOD  A,H         *** CHGSGN ***
056B 2F                 CMA  .           CHANGE SIGN OF HL
056C 67                 LOD  H,A
056D 7D                 LOD  A,L
056E 2F                 CMA  .
056F 6F                 LOD  L,A
0570 23                 INC  HL
0571 78                 LOD  A,B         AND ALSO FLIP B
0572 EE80               XORI X'80'
0574 47                 LOD  B,A
0575 C9                 RET  U
*
0576 7C          CKHLDE LOD  A,H
0577 AA                 XOR  D           SAME SIGN?
0578 F27C05             JMP  NS,CK1      YES, COMPARE
057B EB                 XCH  HL,DE       NO, XCH AND COMP
057C E7          CK1    COMP
057D C9                 RET  U
*
*****************************************************************
*
* *** SETVAL *** FIN *** ENDCHK *** & ERROR (& FRIENDS) ***
*
* "SETVAL" EXPECTS A VARIABLE, FOLLOWED BY AN EQUAL SIGN AND
* THEN AN EXPR.  IT EVALUATES THE EXPR. AND SET THE VARIABLE
* TO THAT VALUE.
*
* "FIN" CHECKS THE END OF A COMMAND.  IF IT ENDED WITH ";",
* EXECUTION CONTINUES.  IF IT ENDED WITH A CR, IT FINDS THE
* NEXT LINE AND CONTINUE FROM THERE.
*
* "ENDCHK" CHECKS IF A COMMAND IS ENDED WITH CR.  THIS IS
* REQUIRED IN CERTAIN COMMANDS. (GOTO, RETURN, AND STOP ETC.)
*
* "ERROR" PRINTS THE STRING POINTED BY DE (AND ENDS WITH CR).
* IT THEN PRINTS THE LINE POINTED BY 'CURRNT' WITH A "?"
* INSERTED AT WHERE THE OLD TEXT POINTER (SHOULD BE ON TOP
* OF THE STACK) POINTS TO.  EXECUTION OF TB IS STOPPED
* AND TBI IS RESTARTED.  HOWEVER, IF 'CURRNT' -> ZERO
* (INDICATING A DIRECT COMMAND), THE DIRECT COMMAND IS NOT
* PRINTED.  AND IF 'CURRNT' -> NEGATIVE # (INDICATING 'INPUT'
* COMMAND, THE INPUT LINE IS NOT PRINTED AND EXECUTION IS
* NOT TERMINATED BUT CONTINUED AT 'INPERR'.
*
* RELATED TO 'ERROR' ARE THE FOLLOWING:
* 'QWHAT' SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
* 'AWHAT' JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
* 'QSORRY' AND 'ASORRY' DO SAME KIND OF THING.
* 'QHOW' AND 'AHOW' IN THE ZERO PAGE SECTION ALSO DO THIS
*
057E FF          SETVAL TSTV             *** SETVAL ***
057F DAA405             JMP  C,QWHAT     "WHAT?" NO VARIABLE
0582 E5                 PUSH HL          SAVE ADDRESS OF VAR.
0583 CF                 TSTC '=',SV1     PASS "=" SIGN
0584 3D
0585 08
0586 DF                 EXPR             EVALUATE EXPR.
0587 444D               LOD  BC,HL       VALUE IN BC NOW
0589 E1                 POP  HL          GET ADDRESS
058A 71                 LOD  M,C         SAVE VALUE
058B 23                 INC  HL
058C 70                 LOD  M,B
058D C9                 RET  U
058E C3A405      SV1    JMP  QWHAT       NO "=" SIGN
*
0591 CF          FIN    TSTC ';',FI1     *** FIN ***
0592 3B
0593 04
0594 F1                 POP  AF          ";", PURGE RET. ADDR.
0595 C34902             JMP  RUNSML      CONTINUE SAME LINE
0598 CF          FI1    TSTC @CR,FI2     NOT ";", IS IT CR?
0599 0D
059A 04
059B F1                 POP  AF          YES, PURGE RET. ADDR.
059C C33902             JMP  RUNNXL      RUN NEXT LINE
059F C9          FI2    RET  U           ELSE RETURN TO CALLER
*
05A0 EF          ENDCHK IGNBLK           *** ENDCHK ***
05A1 FE0D               CMPI @CR         END WITH CR?
05A3 C8                 RET  Z           OK, ELSE SAY: "WHAT?"
*
05A4 D5          QWHAT  PUSH DE          *** QWHAT ***
05A5 11AE00      AWHAT  LODI DE,WHAT     *** AWHAT ***
05A8 97          ERROR  SUB  A           *** ERROR ***
05A9 CD3C06             CALL PRTSTG      PRINT 'WHAT?', 'HOW?'
05AC D1                 POP  DE          OR 'SORRY'
05AD 1A                 LD   A,(DE)      SAVE THE CHARACTER
05AE F5                 PUSH AF          AT WHERE OLD DE ->
05AF 97                 SUB  A           AND PUT A 0 THERE
05B0 12                 ST   A,(DE)
05B1 2A0108             LD   HL,CURRNT   GET CURRENT LINE #
05B4 E5                 PUSH HL
05B5 7E                 LOD  A,M         CHECK THE VALUE
05B6 23                 INC  HL
05B7 B6                 IOR  M
05B8 D1                 POP  DE
05B9 CA0000             JMP  Z,START     IF ZERO, JUST RESTART
05BC 7E                 LOD  A,M         IF NEGATIVE,
05BD B7                 IOR  A
05BE FAA703             JMP  S,INPERR    REDO INPUT
05C1 CDAA06             CALL PRTLN       ELSE PRINT THE LINE
05C4 1B                 DEC  DE          UPTO WHERE THE 0 IS
05C5 F1                 POP  AF          RESTORE THE CHARACTER
05C6 12                 ST   A,(DE)
05C7 3E3F               LODI A,'?'       PRINT A "?"
05C9 D7                 OUTC
05CA 97                 SUB  A           AND THE REST OF THE
05CB CD3C06             CALL PRTSTG      LINE
05CE C7                 RSTART           THEN RESTART
05CF D5          QSORRY PUSH DE          *** QSORRY ***
05D0 11B400      ASORRY LODI DE,SORRY     *** ASORRY ***
05D3 C3A805             JMP  ERROR
*
*****************************************************************
*
* *** GETLN *** FNDLN (& FRIENDS) ***
*
* 'GETLN' READS A INPUT LINE INTO 'BUFFER'.  IT FIRST PROMPT
* THE CHARACTER IN A (GIVEN BY THE CALLER), THEN IT FILLS THE
```

```
* THE BUFFER AND ECHOS.  IT IGNORES LF'S AND NULLS, BUT STILL
* ECHOS THEM BACK.  RUB-OUT IS USED TO CAUSE IT TO DELETE
* THE LAST CHARATER (IF THERE IS ONE), AND ALT-MOD IS USED TO
* CAUSE IT TO DELETE THE WHOLE LINE AND START IT ALL OVER.
* CR SIGNALS THE END OF A LINE, AND CAUSE 'GETLN' TO RETURN.
*
* 'FNDLN' FINDS A LINE WITH A GIVEN LINE # (IN HL) IN THE
* TEXT SAVE AREA.  DE IS USED AS THE TEXT POINTER.  IF THE
* LINE IS FOUND, DE WILL POINT TO THE BEGINNING OF THAT LINE
* (I.E., THE LOW BYTE OF THE LINE #), AND FLAGS ARE NC & Z.
* IF THAT LINE IS NOT THERE AND A LINE WITH A HIGHER LINE #
* IS FOUND, DE POINTS TO THERE AND FLAGS ARE NC & NZ.  IF
* WE REACHED THE END OF TEXT SAVE ARE AND CANNOT FIND THE
* LINE, FLAGS ARE C & NZ.
* 'FNDLN' WILL INITIALIZE DE TO THE BEGINNING OF THE TEXT SAVE
* AREA TO START THE SEARCH.  SOME OTHER ENTRIES OF THIS
* ROUTINE WILL NOT INITIALIZE DE AND DO THE SEARCH.
* 'FNDLNP' WILL START WITH DE AND SEARCH FOR THE LINE #.
* 'FNDNXT' WILL BUMP DE BY 2, FIND A CR AND THEN START SEARCH.
* 'FNDSKP' USE DE TO FIND A CR, AND THEN STRART SEARCH.
*
05D6 D7          GETLN  OUTC             *** GETLN ***
05D7 11371F 0000        LODI  DE,BUFFER  PROMPT AND INIT.
05DA CD3207      GL1    CALL  CHKIO      CHECK KEYBOARD
05DD CADA05              JMP   Z,GL1      NO INPUT, WAIT
05E0 D7                  OUTC             INPUT, ECHO BACK
05E1 FE0A                CMPI  @LF        IGNORE LF
05E3 CADA05              JMP   Z,GL1
05E6 B7                  IOR   A          IGNORE NULL
05E7 CADA05              JMP   Z,GL1
05EA FE7F                CMPI  @DLCH      DELETE LAST CHARACTER?
05EC CAFF05              JMP   Z,GL3      YES
05EF FE7D                CMPI  @DLLN      DELETE THE WHOLE LINE?
05F1 CA0C06              JMP   Z,GL4      YES
05F4 12                  ST    A,(DE)     ELSE, SAVE INPUT
05F5 13                  INC   DE         AND BUMP POINTER
05F6 FE0D                CMPI  @CR        WAS IT CR?
05F8 C8                  RET   Z          YES, END OF LINE
05F9 78                  LOD   A,E        ELSE, MORE FREE ROOM?
05FA FE7F                CMPI  BUFEND,>
05FC C2DA05              JMP   NZ,GL1     YES, GET NEXT INPUT
05FF 78          GL3    LOD   A,E        DELETE LAST CHARACTER
0600 FE37                CMPI  BUFFER,>   BUT DO WE HAVE ANY?
0602 CA0C06              JMP   Z,GL4      NO, REDO WHOLE LINE
0605 1B                  DEC   DE         YES, BACKUP POINTER
0606 3E5C                LODI  A,@BKS     AND ECHO A BACK-SLASH
0608 D7                  OUTC
0609 C3DA05              JMP   GL1        GO GET NEXT INPUT
060C CD0E00      GL4    CALL  CRLF       REDO ENTIRE LINE
060F 3E5E                LODI  A,@UPA     CR, LF AND UP-ARROW
0611 C30605              JMP   GETLN
*
0614 7C          FNDLN  LOD   A,H        *** FNDLN ***
0615 B7                  IOR   A          CHECK SIGN OF HL
0616 FA9F00              JMP   S,QHOW     IT CANNOT BE -
0619 111508              LODI  DE,TXTBGN  INIT. TEXT POINTER
*
061C             FNDLNP EQU   *           *** FNDLNP ***
061C E5          FL1    PUSH  HL         SAVE LINE #
061D 2A1308              LD    HL,TXTUNF  CHECK IF WE PASSED END
0620 2B                  DEC   HL
0621 E7                  COMP
0622 E1                  POP   HL         GET LINE # BACK
0623 D8                  RET   C          C,NZ PASSED END
0624 1A                  LD    A,(DE)     WE DID NOT, GET BYTE 1
0625 95                  SUB   L          IS THIS THE LINE?
0626 47                  LOD   B,A        COMPARE LOW ORDER
0627 13                  INC   DE
0628 1A                  LD    A,(DE)     GET BYTE 2
0629 9C                  SBB   H          COMPARE HIGH ORDER
062A DA3106              JMP   C,FL2      NO, NOT THERE YET
062D 1B                  DEC   DE         ELSE WE EITHER FOUND
062E B0                  IOR   B          IT, OR IT IS NOT THERE
062F C9                  RET   U          NC,Z:FOUND; NC,NZ:NO
*
0630             FNDNXT EQU   *           *** FNDNXT ***
0630 13                  INC   DE         FIND NEXT LINE
0631 13          FL2    INC   DE         JUST PASSED BYTE 1 & 2
*
0632 1A          FNDSKP LD    A,(DE)     *** FNDSKP ***
0633 FE0D                CMPI  @CR        TRY TO FIND CR
0635 C23106              JMP   NZ,FL2     KEEP LOOKING
0638 13                  INC   DE         FOUND CR, SKIP OVER
0639 C31C06              JMP   FL1        CHECK IF END OF TEXT
*
*****************************************************************
*
* *** PRTSTG *** QTSTG *** PRTNUM *** & PRTLN ***
*
* 'PRTSTG' PRINTS A STRING POINTED BY DE.  IT STOPS PRINTING
* AND RETURNS TO CALLER WHEN EITHER A CR IS PRINTED OR WHEN
* THE NEXT BYTE IS THE SAME AS WHAT WAS IN A (GIVEN BY THE
* CALLER).  OLD A IS STORED IN B, OLD B IS LOST.
*
* 'QTSTG' LOOKS FOR A BACK-ARROW, SINGLE QUOTE, OR DOUBLE
* QUOTE.  IF NONE OF THESE, RETURN TO CALLER.  IF BACK-ARROW,
* OUTPUT A CR WITHOUT A LF.  IF SINGLE OR DOUBLE QUOTE, PRINT
* THE STRING IN THE QUOTE AND DEMANDS A MATCHING UNQUOTE.
* AFTER THE PRINTING THE NEXT 3 BYTES OF THE CALLER IS SKIPPED
* OVER (USUALLY A JUMP INSTRUCTION).
*
* 'PRTNUM' PRINTS THE NUMBER IN HL.  LEADING BLANKS ARE ADDED
* IF NEEDED TO PAD THE NUMBER OF SPACES TO THE NUMBER IN C.
* HOWEVER, IF THE NUMBER OF DIGITS IS LARGER THAN THE # IN
* C, ALL DIGITS ARE PRINTED ANYWAY.  NEGATIVE SIGN IS ALSO
* PRINTED AND COUNTED IN.  POSITIVE SIGN IS NOT.
*
* 'PRTLN' PRINTS A SAVED TEXT LINE WITH LINE # AND ALL.
*
063C 47          PRTSTG LOD   B,A        *** PRTSTG ***
063D 1A          PS1    LD    A,(DE)     GET A CHARACTER
063E 13                 INC   DE         BUMP POINTER
063F B8                 CMP   B          SAME AS OLD A?
0640 C8                 RET   Z          YES, RETURN
0641 D7                 OUTC             ELSE PRINT IT
0642 FE0D               CMPI  @CR        WAS IT A CR?
0644 C23D06             JMP   NZ,PS1     NO, NEXT
0647 C9                 RET   U          YES, RETURN
*
0648 CF          QTSTG  TSTC  '"',QT3    *** QTSTG ***
0649 22
064A 0F
0648 3E22               LODI  A,'"'      IT IS A "
064D CD3C06      QT1    CALL  PRTSTG     PRINT UNTIL ANOTHER
0650 FE0D               CMPI  @CR        WAS LAST ONE A CR?
0652 E1                 POP   HL         RETURN ADDRESS
0653 CA3902             JMP   Z,RUNNXL   WAS CR, RUN NEXT LINE
0656 232323      QT2    INC   HL,3       SKIP 3 BYTES ON RETURN
0659 E9                 JMP   (HL)       RETURN
065A CF          QT3    TSTC  @QT,QT4    IS IT A ' ?
065B 27
065C 05
065D 3E27               LODI  A,@QT      YES, DO SAME
065F C34D06             JMP   QT1        AS IN "
0662 CF          QT4    TSTC  @BKA,QT5   IS IT BACK-ARROW?
0663 5F
0664 08
0665 3E8D               LODI  A,X'8D'    YES, CR WITHOUT LF
0667 D7                 OUTC             DO IT TWICE TO GIVE
0668 D7                 OUTC             TTY ENOUGH TIME
0669 E1                 POP   HL         RETURN ADDRESS
066A C35606             JMP   QT2
066D C9          QT5    RET   U          NONE OF ABOVE
*
066E D5          PRTNUM PUSH  DE         *** PRTNUM ***
066F 110A00             LODI  DE,10      DECIMAL
0672 D5                 PUSH  DE         SAVE AS A FLAG
0673 42                 LOD   B,D        B=SIGN
0674 0D                 DEC   C          C=SPACES
0675 CD6705             CALL  CHKSGN     CHECK SIGN
0678 F27E06             JMP   NS,PN1     NO SIGN
067B 062D               LODI  B,'-'      B=SIGN
067D 0D                 DEC   C          '-' TAKES SPACE
067E C5          PN1    PUSH  BC         SAVE SIGN & SPACE
067F CD4A05      PN2    CALL  DIVIDE     DEVIDE HL BY 10
0682 78                 LOD   A,B        RESULT 0?
```

```
0683 B1                 IOR   C
0684 CA8F06             JMP   Z,PN3       YES, WE GOT ALL
0687 E3                 XCH   HL,(SP)     NO, SAVE REMAINDER
0688 2D                 DEC   L           AND COUNT SPACE
0689 E5                 PUSH  HL          HL IS OLD BC
068A 6069               LOD   HL,BC       MOVE RESULT TO BC
068C C37F06             JMP   PN2         AND DIVIDE BY 10
068F C1           PN3   POP   BC          WE GOT ALL DIGITS IN
0690 0D           PN4   DEC   C           THE STACK
0691 79                 LOD   A,C         LOOK AT SPACE COUNT
0692 B7                 IOR   A
0693 FA9C06             JMP   S,PN5       NO LEADING BLANKS
0696 3E20               LODI  A,' '       LEADING BLANKS
0698 D7                 OUTC
0699 C39006             JMP   PN4         MORE?
069C 78           PN5   LOD   A,B         PRINT SIGN
069D D7                 OUTC              MAYBE - OR NULL
069E 5D                 LOD   E,L         LAST REMAINDER IN E
069F 78           PN6   LOD   A,E         CHECK DIGIT IN E
06A0 FE0A               CMPI  10          10 IS FLAG FOR NO MORE
06A2 D1                 POP   DE
06A3 C8                 RET   Z           IF SO, RETURN
06A4 C630               ADDI  '0'         ELSE COVERT TO ASCII
06A6 D7                 OUTC              AND PRINT THE DIGIT
06A7 C39F06             JMP   PN6         GO BACK FOR MORE
*
06AA 1A           PRTLN LD    A,(DE)      *** PRTLN ***
06AB 6F                 LOD   L,A         LOW ORDER LINE #
06AC 13                 INC   DE
06AD 1A                 LD    A,(DE)      HIGH ORDER
06AE 67                 LOD   H,A
06AF 13                 INC   DE
06B0 0E04               LODI  C,4         PRINT 4 DIGIT LINE #
06B2 CD6E06             CALL  PRTNUM
06B5 3E20               LODI  A,' '       FOLLOWED BY A BLANK
06B7 D7                 OUTC
06B8 97                 SUB   A           AND THEN THE TEXT
06B9 CD3C06             CALL  PRTSTG
06BC C9                 RET   U
*
*****************************************************************
*
* *** MVUP *** MVDOWN *** POPA *** & PUSHA ***
*
* 'MVUP' MOVES A BLOCK UP FROM WHERE DE-> TO WHERE BC-> UNTIL
* DE = HL
*
* 'MVDOWN' MOVES A BLOCK DOWN FROM WHERE DE-> TO WHERE HL->
* UNTIL DE = BC
*
* 'POPA' RESTORES THE 'FOR' LOOP VARIABLE SAVE AREA FROM THE
* STACK
*
* 'PUSHA' STACKS THE 'FOR' LOOP VARIABLE SAVE AREA INTO THE
* STACK
*
06BD E7           MVUP  COMP              *** MVUP ***
06BE C8                 RET   Z           DE = HL, RETURN
06BF 1A                 LD    A,(DE)      GET ONE BYTE
06C0 02                 ST    A,(BC)      MOVE IT
06C1 13                 INC   DE          INCREASE BOTH POINTERS
06C2 03                 INC   BC
06C3 C3BD06             JMP   MVUP        UNTIL DONE
*
06C6 78           MVDOWN LOD  A,B         *** MVDOWN ***
06C7 92                 SUB   D           TEST IF DE = BC
06C8 C2CE06             JMP   NZ,MD1      NO, GO MOVE
06CB 79                 LOD   A,C         MAYBE, OTHER BYTE?
06CC 93                 SUB   E
06CD C8                 RET   Z           YES, RETURN
06CE 1B           MD1   DEC   DE          ELSE MOVE A BYTE
06CF 2B                 DEC   HL          BUT FIRST DECREASE
06D0 1A                 LD    A,(DE)      BOTH POINTERS AND
06D1 77                 LOD   M,A         THEN DO IT
06D2 C3C606             JMP   MVDOWN      LOOP BACK
*
06D5 C1           POPA  POP   BC          BC = RETURN ADDR.
06D6 E1                 POP   HL          RESTORE LOPVAR, BUT
06D7 220708             ST    HL,LOPVAR   =0 MEANS NO MORE
06DA 7C                 LOD   A,H
06DB B5                 IOR   L
06DC CAEF06             JMP   Z,PP1       YEP, GO RETURN
06DF E1                 POP   HL          NOP, RESTORE OTHERS
06E0 220908             ST    HL,LOPINC
06E3 E1                 POP   HL
06E4 220B08             ST    HL,LOPLMT
06E7 E1                 POP   HL
06E8 220D08             ST    HL,LOPLN
06EB E1                 POP   HL
06EC 220F08             ST    HL,LOPPT
06EF C5           PP1   PUSH  BC          BC = RETURN ADDR.
06F0 C9                 RET   U
*
06F1 21A71F 0000  PUSHA LODI  HL,STKLMT   *** PUSHA ***
06F4 CD6A05             CALL  CHGSGN
06F7 C1                 POP   BC          BC=RETURN ADDRESS
06F8 39                 ADD   HL,SP       IS STACK NEAR THE TOP?
06F9 D2CF05             JMP   NC,QSORRY   YES, SORRY FOR THAT.
06FC 2A0708             LD    HL,LOPVAR   ELSE SAVE LOOP VAR.S
06FF 7C                 LOD   A,H         BUT IF LOPVAR IS 0
0700 B5                 IOR   L           THAT WILL BE ALL
0701 CA1707             JMP   Z,PU1
0704 2A0F08             LD    HL,LOPPT    ELSE, MORE TO SAVE
0707 E5                 PUSH  HL
0708 2A0D08             LD    HL,LOPLN
070B E5                 PUSH  HL
070C 2A0B08             LD    HL,LOPLMT
070F E5                 PUSH  HL
0710 2A0908             LD    HL,LOPINC
0713 E5                 PUSH  HL
0714 2A0708             LD    HL,LOPVAR
0717 E5           PU1   PUSH  HL
0718 C5                 PUSH  BC          BC = RETURN ADDR.
0719 C9                 RET   U
*
*****************************************************************
*
* *** OUTC *** & CHKIO ***
*
* THESE ARE THE ONLY I/O ROUTINES IN TBI.
* 'OUTC' IS CONTROLLED BY A SOFTWARE SWITCH 'OCSW'.  IF OCSW=0
* 'OUTC' WILL JUST RETURN TO THE CALLER.  IF OCSW IS NOT 0,
* IT WILL OUTPUT THE BYTE IN A.  IF THAT IS A CR, A LF IS ALSO
* SEND OUT.  ONLY THE FLAGS MAY BE CHANGED AT RETURN. ALL REG.
* ARE RESTORED.
*
* 'CHKIO' CHECKS THE INPUT.  IF NO INPUT, IT WILL RETURN TO
* THE CALLER WITH THE Z FLAG SET.  IF THERE IS INPUT, Z FLAG
* IS CLEARED AND THE INPUT BYTE IS IN A.  HOWERER, IF THE
* INPUT IS A CONTROL-O, THE 'OCSW' SWITCH IS COMPLIMENTED, AND
* Z FLAG IS RETURNED.  IF A CONTROL-C IS READ, 'CHKIO' WILL
* RESTART TBI AND DO NOT RETURN TO THE CALLER.
*
*                 OUTC  PUSH  AF          THIS IS AT LOC. 10
*                       LD    A,OCSW      CHECK SOFTWARE SWITCH
*                       IOR   A
071A C21F07       OC2   JMP   NZ,OC3      IT IS ON
071D F1                 POP   AF          IT IS OFF
071E C9                 RET   U           RESTORE AF AND RETURN
071F DB00         OC3   INP   0           COME HERE TO DO OUTPUT
0721 E602               ANDI  X'02'       STATUS BIT
0723 CA1F07             JMP   Z,OC3       NOT READY, WAIT
0726 F1                 POP   AF          READY, GET OLD A BACK
0727 D301               OUT   1           AND SEND IT OUT
0729 FE0D               CMPI  @CR         WAS IT CR?
072B C0                 RET   NZ          NO, FINISHED
072C 3E0A               LODI  A,@LF       YES, WE SEND LF TOO
072E D7                 OUTC              THIS IS RECURSIVE
072F 3E0D               LODI  A,@CR       GET CR BACK IN A
0731 C9                 RET   U
0732 DB00         CHKIO INP   0           *** CHKIO ***
0734 00                 NOP   .           STATUS BIT FLIPPED?
0735 E620               ANDI  X'20'       MASK STATUS BIT
0737 C8                 RET   Z           NOT READY, RETURN "Z"
0738 DB01               INP   1           READY, READ DATA
073A E67F               ANDI  X'7F'       MASK BIT 7 OFF
```

A NOTE TO MEMBERS OF THE SOUTHERN CALIFORNIA COMPUTER SOCIETY

by Jim C. Warren, Jr., Editor, *Dr. Dobb's Journal*

I am writing to object to several of the actions of Louis G. Fields, vice president of SCCS, that have recently come to our attention.

1. On April 12th, we sent Mr. Fields 500 copies of the March-April issue of *People's Computer Company* newspaper, for distribution at the April 24th SCCS meeting, compliments of PCC.

I happened to attend the meeting and found that, first of all, Mr. Fields had failed to bring most of that supply to the meeting, and secondly, those he *had* distributed were stamped in oversized red letters, "Compliments of Louis G. Fields."

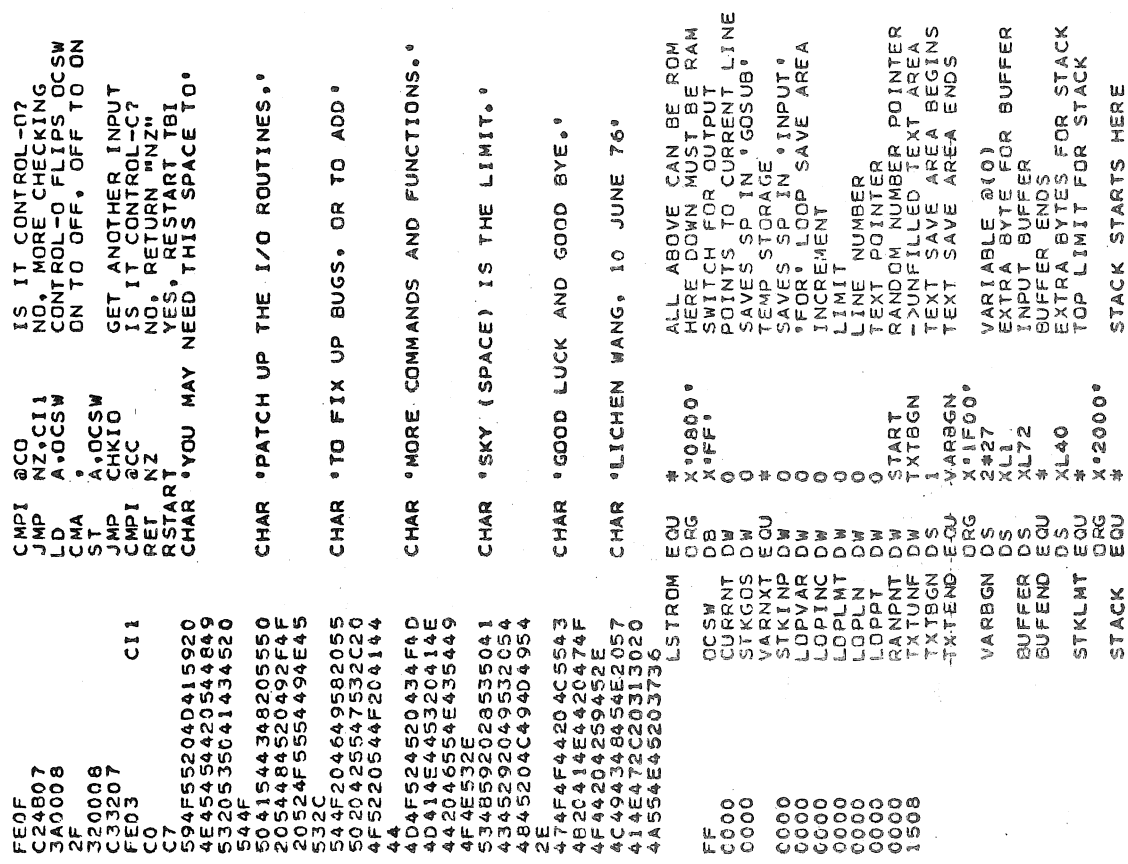
This may have given the **completely false** impression that either Mr. Fields was the donor of these complimentary copies, or that PCC was somehow supporting Mr. Fields. Mr. **Fields did NOT donate these copies**, and, after having the opportunity to see him in action, we wish to **adamantly state that PCC does NOT support Mr. Fields in any way!**

2. Along with these 500 **free** copies of *PCC*, we also forwarded a number of other PCC publications, as per Mr. Fields' telephone order of April 12th. His order totaled $359.75 (unrelated to the free *PCC* copies). We forwarded the telephone-ordered publications to Mr. Fields with the understanding that he was representing SCCS.

In spite of repeated billings, to date he has paid only $59.75, received June 10th. Our contacts with the *other* SCCS officers have consistently shown them to be responsible, trustworthy, and responsive.

3. As a result of the above-noted incidents, PCC will make no further donations or sales through Mr. Fields to SCCS. We will be delighted, however, to work with any of the many other SCCS officers.

Note: We do not mean this to in any way reflect negatively on the SCCS, nor on the many Society officers who are doing an excellent job of administering a worthwhile organization that provides many useful services to the computer hobbyist community.



```
073C FE0F          CMPI @CO       IS IT CONTROL-O?
073E C24807        JMP  NZ.CI1    NO, MORE CHECKING
0741 3A0008        LD   A.OCSW    CONTROL-O FLIPS OCSW
0744 2F            CMA            ON TO OFF, OFF TO ON
0745 320008        ST   A.OCSW
0748 C33207        JMP  CHKIO     GET ANOTHER INPUT
074B FE03      CI1 CMPI @CC       IS IT CONTROL-C?
074D C0            RET  NZ        NO, RETURN "NZ"
074E C7            RSTART         YES, RESTART TBI
074F 594F55204D415920  CHAR 'YOU MAY NEED THIS SPACE TO'
0757 4E45544420544849
075F 5320535041434520
0767 544F
0769 5041544348205550  CHAR 'PATCH UP THE I/O ROUTINES.'
0771 20544845204492F4F
0779 20524F5554494E45
0781 532C
0783 544F20464958205S
078B 5020425547532C20
0793 4F5220544F204144
079B 44
079C 4D4F5245204344F4D  CHAR 'TO FIX UP BUGS, OR TO ADD'
07A4 4D414E44532041 4E
07AC 4420465554E435449
07B4 4F4E532E
07B8 5348592028535041  CHAR 'MORE COMMANDS AND FUNCTIONS.'
07C0 4345292049532054
07C8 484520C494D4954
07D0 2E
07D1 474F4F4420C5543   CHAR 'SKY (SPACE) IS THE LIMIT.'
07D9 4B2C414E4420474F
07E1 4F442042594S2E
07E8 4C494348454E2057  CHAR 'GOOD LUCK AND GOOD BYE.'
07F0 414E472C20313020
07F8 4A554E4520373G
07FF               LSTROM EQU  #'0800'  ALL ABOVE CAN BE ROM
                   ORG  X'0800'  HERE DOWN MUST BE RAM
0800 FF        OCSW DB   X'FF'    SWITCH FOR OUTPUT
0801 0000      CURRNT DW 0        POINTS TO CURRENT LINE
0803 0000      STKGOS DW 0        SAVES SP IN 'GOSUB'
0805           VARNXT EQU #       TEMP STORAGE
0805 0000      STKINP DW 0        SAVES SP IN 'INPUT'
0807 0000      LOPVAR DW 0        'FOR' LOOP SAVE AREA
0809 0000      LOPINC DW 0        INCREMENT
080B 0000      LOPLMT DW 0        LIMIT
080D 0000      LOPLN  DW 0        LINE NUMBER
080F 0000      LOPPT  DW 0        TEXT POINTER
0811 0000      RANPNT DW START    RANDOM NUMBER POINTER
0813 1508      TXTUNF DW TXTBGN   ->UNFILLED TEXT AREA
0815           TXTBGN DS 1        TEXT SAVE AREA BEGINS
#F00           TXTEND EQU VARBGN  TEXT SAVE AREA ENDS
                   ORG  X'1F00'
1F00           VARBGN DS 2#27     VARIABLE @(0)
1F36           BUFFER DS XL1      EXTRA BYTE FOR BUFFER
1F37           BUFEND DS XL72     INPUT BUFFER
1F7F                  EQU #       BUFFER ENDS
1F7F           STKLMT DS XL40     EXTRA BYTES FOR STACK
1FA7                  EQU X'2000' TOP LIMIT FOR STACK
                   ORG  X'2000'
2000           STACK  EQU #       STACK STARTS HERE
                   END
```

INDEX TO *THE COMPUTER HOBBYIST*
VOLUME 1, ISSUES 1 THROUGH 9

This is an *excellent* newsletter. We strongly recommend it to every serious computer hobbyist. Not flashy; no side-show . . . just excellent, comprehensive, well-written articles.
—Jim Warren

The Computer Hobbyist is a newsletter/magazine totally dedicated to the use of computers and associated devices as a hobby. Both software and hardware are discussed in feature articles. Circuit diagrams and program listings frequently supplement these articles. In addition to the features, each issue contains an editorial and regular columns on surplus, letters from subscribers, and want ads free to subscribers. Occasionally, as the material warrants, book and product reviews are printed. Frequency of publication varies from 4 to 12 weeks, so subscriptions are for a particular number of issues rather than a time period. *The Computer Hobbyist* is offset printed on looseleaf 8½"x11" paper suitable for ring binding (except for the first three issues which were folded to half size) and is mailed third class to subscribers in the continental United States.

Occasionally, printed circuit boards and other hard to find components are made available to aid our readers in constructing the projects written up.

For further information write to: *The Computer Hobbyist*, Box 295, Cary NC 27511.

THE COMPUTER HOBBYIST

NUMBER 1
1. A GRAPHICS DISPLAY FOR THE 8008 PART 1 - Fundamental concepts and programming technique for inexpensive vector graphics display.
2. SURPLUS SUMMARY - THE TELETYPE STORY - Survey of teleprinter equipment suitable for computer output.
3. NOTES ON THE 8008 INSTRUCTION SET - Some simple tips for more effective 8008 programming not mentioned in the Intel manuals.

NUMBER 2
1. THE 8080 IS HERE - MITS ALTAIR 8800 product line preview and comments on the future of hobby computers.
2. LOGIC SYMBOL CONVENTIONS or HOW TO READ TCH LOGIC DIAGRAMS - Discussion of MIL-STD-806C logic symbols and logic design using the "dot convention".
3. A GRAPHICS DISPLAY FOR THE 8008 PART 2 - Description and diagram of digital and analog circuitry for vector graphics display generator.
4. INTERFACING A 5-LEVEL TELEPRINTER PART 1 - Description of simple current loop interface with common teleprinters.

NUMBER 3
1. A GRAPHICS DISPLAY FOR THE 8008 PART 3 - Description and diagram for large screen XYZ scope using common magnetic deflection CRT's and photos of display.
2. BOOK REVIEW - Critical review of MICROCOMPUTER DESIGN from Martin Research.
3. CHESS BOARD DISPLAY - Description and listing of chessboard and chesspiece display program, fits in about 500 bytes.
4. INTERFACING A 5-LEVEL TELEPRINTER PART 2 - Description and listing of print software that accepts ASCII input.
5. A CHEAP MARK SENSE CARD READER - Description and diagram of circuit for reading pencilled data from tab cards with surplus Western Union card reader.

NUMBER 4
1. HUMAN INTERFACE YOUR GRAPHICS DISPLAY - Description, diagram, and driver software for a 4 variable proportional control input for interactive graphics.
2. 8008 VS 8080 VS IMP-16 WHICH MICROPROCESSOR FOR YOU?. - Detailed comparison of strengths and weaknesses of microprocessors available to hobbyists.

NUMBER 5
1. TCH AUDIO CASSETTE DATA RECORDING STANDARD PART 1 - Rationale and description of proposed data recording standard for recording on audio cassettes.
2. RANDOM NUMBER GENERATOR - Description and listing of random number generator, sequence length over 2 billion, program length 40 bytes
3. TCH STANDARD AUDIO CASSETTE INTERFACE - Description and diagram of machine independent audio cassette interface conforming to TCH standard.
4. DRAWING WITH YOUR POT CONTROLS - Description and listing of program for interactive drawing on graphics display.

NUMBER 6
1. TCH AUDIO CASSETTE DATA RECORDING STANDARD PART 2 - Discussion of logical data format, cyclic redundancy error detection technique, and summary of standard.
2. ADD A DATA STACK TO YOUR 8008 - Description, circuit, and timing diagram for a 16 byte push-pop stack enhancement for 8008; machine status save/restore routine.
3. THE TCH CASSETTE INTERFACE PRINTED CIRCUIT BOARD - description, foil patterns,

parts list and assembly diagram for TCH audio cassette interface.

NUMBER 7
1. TCH AUDIO CASSETTE STANDARD ROM - Description and listing of audio cassette software for both 8008 and 8080. Also order form for 1702/5203 programming.
2. BOOK REVIEW - Critical review of MACHINE LANGUAGE PROGRAMMING FOR THE 8008 by Nat Wadsworth of Scelbi Computer Consulting Inc.
3. NEW PRODUCTS - Brief review of Micro 400 computer kit and the Cramerkit series of microcomputer kits.
4. COMPUTER PING-PONG - Discussion, description, and listing of ping-pong program using the TCH graphic display and pot controls.

NUMBER 8
1. INTERFACING THE ALTAIR 8800 PART 1 - Detailed discussion of system loading considerations, timing diagrams, and example TCH cassette and ROM interface.
2. NEW PRODUCTS - Brief review of Altair 680, JOLT, and Sphere computer kits. Brief discussion of Mini-Software's FORTRAN system for the 8080.
3. AN IMP-16 MICROCOMPUTER SYSTEM PART 1 - Overall system description of high performance 16 bit microcomputer construction project.
4. WHAT THIS COUNTRY NEEDS IS A GOOD $20 MICROPROCESSOR - Discussion of features and programming considerations of the MOS Technology 650X series of microprocessors.

NUMBER 9
1. TCH SUPER SIMPLE FLOPPY DISK INTERFACE PART 1 - General description of floppy disk drives and discussion of controller requirements.
2. INTERFACING THE ALTAIR 8800 PART 2 - Discussion of polled interrupt scheme that doesn't require a vectored interrupt card, diagram for interrupting keyboard.
3. AN IMP-16 MICROCOMPUTER SYSTEM PART 2 - Detailed discussion of system bus controller. Also schematic and timing diagrams for bus controller.

JIPDEC VISITS PCC

We didn't know that People's Computer Company fame had spread so far. On May 19th, we were visited by the Microprocessor Application Study Team from the Japan Information Processing Development Center. The team included Professor Hidetoshi from Keio University; Koichiro Ishihara, a Research Fellow with Hitachi Systems Development Lab; Tatsui Miyakawa, Manager of Fujitsu's Computer Science Lab; and Kazuya Watanabe, from NEC (Nippon Electric Company). We were surprised, honored, and delighted by their visit, and pleased with the opportunity to exchange ideas and information.

What may be even more interesting to our readers is that, in the face of all of the massive microprocessor research and manufacturing that is centered in the San Francisco Bay area, this JIPDEC group chose to visit only PCC and IMS Associates, Inc. (IMSAI).

CHICAGO STORES:
CHICAGO COMPUTER STORE
ITTY BITTY MACHINE CO.

There are two new computer stores in the Chicago area:

Chicago Computer Store (handles MITS stuff)
517 Talcott Rd at Hwy 62
Park Ridge IL 60068

Itty Bitty Machine Co. (handles IMSAI, etc.)
1316 W. Chicago Ave.
Evanston Ill.

TedNelson of *Computer Lib* has a hand in the second one.

WHAT'S A BAMUG?

by Richard Lindberg

BAMUG stands for [San Francisco] Bay Area Microprocessor's Users Group. It is one of the many computer hobby clubs which have sprung up like weeds in the past year. We meet in San Leandro, Cal., on the first Thursday of each month. Meetings are held at the Great Western Savings building at E. 14th and Davis, at 7 p.m.

BAMUG is a relatively small group. Anyone is welcome to join us, but we are mostly interested in the Intel 8080. We are starting a software library which, though still small, contains some very useful programs.

We try to keep the meetings interesting. We have some demonstration at every meeting, if possible. At the last meeting we had one of our members, Ken Jackman, tell us some of the concepts involved in Nim-like games. He is writing a book on the subject and presents it well with the aid of his vintage Nova.

At the next meeting we will have George Morrow. He will be hauling odds and ends out of his basement to show us. Of particular interest is his inexpensive tape cassette interface. We also hope to get a preview of his 16K ALTAIR-IMSAI compatible RAM board.

We have voluntary dues of 50 cents per meeting, but you are welcome to attend without paying. If you have a program to contribute to our library, please bring us a copy.

(Reprinted with permission from *Homebrew Computer Club Newsletter*)

Texas Tiny BASIC (TBX) Marries TV-Cassette Operating System (TVCOS)

by Digital Group Software Systems, Inc. staff
Box 6528, Denver CO 80206; (303) 861-1686

QUESTION: What could almost be better than having your micro programs run correctly?

ANSWER: Having unique micro software to utilize to your heart's content!! Whether to create some fantastic program(s) or to have the facility to entertain yourself and/or friends with programs and games requiring a bit of "think power" . . . especially for those days when your "think power" could stand a bit of bolstering!

All this is possible thanks to the software packages now being provided by Digital Group Software Systems, Inc., (DGSS), which was recently established and is headed by Chuck and Dianne Howerton. Software which is now available on cassette tapes, complete (for the most part) with documentation for each offering, is as follows:

1. TINY BASIC EXTENDED (TBS-TVCOS) 10K, VERSION 1

Created by Dick Whipple and John Arnold of Tyler, Texas, based upon design criteria published in *Dr. Dobb's Journal* Volume 1, Number 1, page 14. Dr. Robert Suding of The Digital Group designed and developed the software interfaces between TXB (TINY BASIC EXTENDED) and the TV-Cassette Operating System (TVCOS) for the Digital Group 8080 microcomputer. This is a superset of TINY BASIC as originally proposed; a limited and modified subset of the full BASIC language. It has 26 possible simple or dimensioned variables, 4 immediate commands (LST, NEW, RUN, and SZE), and 12 commands (DIM, DTA, END, FOR, GOTO, GOSUB, IF, IN, LET, NXT, PR, and RET) which may be entered either as program statements or immediate execution commands. Also included is a RN or Random Number generator. 14 pages of documentation, which lets you do everything with TINY BASIC EXTENDED that is presently possible, is also included.

PRICE: $5.00 @ cassette with documentation.

2. TINY BASIC GAMES, DGSS SET 1

Contains 5 games written in TBX language—3 games are TBX versions of games listed in either *101 Computer Games* or *What to do After You Hit Return* [Both are available through PCC bookstore. Please see inside of back cover.]. Other two games are original products of author Howerton. Games on Set 1 tape are:

a. CHOMP: 2 or more players take turns biting into the poisioned cookie - Each trying to force one of the others to take the poisioned "byte"!!!

b. CHECKERS: You against the computer in this classic board game - WARNING: The computer is as crafty as a 6-year old playing his first game!!

c. TIC-TAC-TOE: The computer plays a fairly reasonable game with just enough randomness in its selection to make the game interesting - Can be beaten.

d. DIGIGUESS: Try to guess a 4 digit number "thunk" up by the computer based upon clues it supplies.

e. BRAINTEASER: A strategy game - You against your own worst enemy...YOU! Try to create a pattern working with a set of rules to drive you nuts!!!

PRICE: $5.00 @ cassette with documentation

NOTE: TINY BASIC GAMES, DGSS Set 1, and all TINY BASIC GAMES are read in AFTER "bootstrapping" TBX-TVCOS program into micro-computer.... and away (:RUNcr) you go!!

3. TINY BASIC GAMES, DGSS Set 2

All the games in this set were adapted to TBX-TVCOS by Dr. Robert Suding from 101 COMPUTER GAMES, PEOPLE'S COMPUTER COMPANY and WHAT TO DO AFTER YOU HIT RETURN.

a. WAR-3: An artillery duel between 2 or 3 armies - OBJECT: To blast the opponent(s).

b. THERAPY: Have a relaxing "conversation" with DR. THERAPY and investigate your inner problem(s).

c. GOLF: 9 holes of championship play on the "micro-links" - Hooks and slices are rare and 1-putts from 40 feet are fairly common.

d. REVERSE: Test your skills at arranging things in sequential order from a random beginning according to a fixed set of rules.

e. BIORYTHM: Predict your "highs" and

"lows" from a computer-produced chart based upon your personal statistics.

PRICE: $5.00 @ cassette with documentation

4. TINY BASIC GAMES, DGSS Set 3

All the games in this set were adapted to TBX-TVCOS by Ted Holdahl from 101 COMPUTER GAMES.

a. TAXMAN: Test your knowledge of factoring by trying to beat the TAXMAN!! It's tough but he can be beaten!!
b. SNARK: Learn symbolic logic and the use of Ven diagrams while trying to find the SNARK.
c. TRAP: Find the computers' number by trapping it between two guesses of your own.
d. NUMBER: Learn binary-searching techniques while trying to pinpoint a randomly-generated number between 1 and 100.
e. SQUARE-ROOTS: Computes the square root of any number from -32K to +32K.
f. CLOCK: This bonus program keeps accurate time using looping techniques and displays on the screen to the nearest second.

PRICE: $5.00 @ cassette with documentation.

5. TINY BASIC GAMES, DGSS Set 4

All the games in this set were adapted to TBX-TVCOS by Ted Holdahl from 101 COMPUTER GAMES.

a. HAMURABI: Try governing ancient Sumaria successfully. The author says that the object of the game is to discover the rules.
b. STARS: Guess the computers' number based on clues which tell you whether your warm, hot, or cold.
c. 23-MATCHES: Outfox the computer by making it take the last match in the pile - You can beat it if you try hard enough.
d. 20-QUESTIONS: Take-off on an old radio-TV game where the player guesses the computers' number by asking questions.
e. BLACKJACK: From 1-9 players - Play against the house in this Casino game.
f. FACTOR: A bonus program which will compute the factors of any number up to 32,767 or tell you if it is prime.
g. BATUM: A super bonus game which is a varient of 23-MATCHES. Good selections will really make the computer think!!

PRICE: $5.00 @ Cassette with documentation.

6. KINGDOM; LIFE 1 & LIFE 2
(all on one tape)

a. KINGDOM: Lets you be the "KING" of land/bushels/population of YOUR "KINGDOM" - Object of game is to accumulate 1 million acres and/or bushels which can only be accomplished by buying and selling land. Tests your ability to buy and sell these items and outwit the complications all KINGS have!!
b. LIFE 1: Standard LIFE which will take a pattern and, with simple rules of LIFE, iterate until a stable situation is reached.
c. LIFE 2: Bi-Symetrical pattern generator based upon modified rules of LIFE but a stable situation is NEVER achieved - Will continue to run and never repeat pattern for approximately 10^{31} years!!

PRICE: $5.00 @ cassette with documentation (sort of)!!

7. EDUCATOR 8080

Designed to assist the micro-computer user in understanding the effect that the execution of various instructions has on status and operation of micro-processor unit - Provides continuous real-time display of status flags, Accumulator, B and C registers which reflect their contents as changed and/or modified by instructions

issued by user - Complete with EDUCATOR instruction set.

PRICE: $10.00 @ cassette with documentation.

8. AMATEUR RADIO HAM CASSETTE FOR 8080

Amateur Radio (CW) Send and Receive
RTTY (Baudot) Send and Receive

CW Receive is automatically self-adjusting to any CW speed sent
CW Send has 8 100-character memories that may be individually called up
CW Send also features a 256-character software FIFO buffer

RTTY Receive can select 60, 66 and 100 Words per Minute (WPM); upper or lower case output to TV.
RTTY Send program can send at 60, 66, or 100 WPM.

All 4 programs are designed to reside simultaneously in a 10K Digital Group 8080 system.

PRICE: Documentation and cassette HAM-1 $5.00

9. OPERATING GUIDE - TVT MONITOR 8 FOR 8008 BASED SYSTEMS

The TVT Monitor 8 is a modification and extention of the Monitor 8 published by MIL before their untimely demise. The original Monitor 8 was written to be used with teletype/paper tape input and output. The TVT Monitor 8 is designed to be used with a Digital Group TVT, ASCII keyboard and Digital Group Cassette tape interface as input and output.

Several features were added to the Monitor 8 to increase its usefulness. One of these is software scrolling of the TVT. This causes the TVT to simulate the format of a teletype by displaying the previous seven lines. Two other additions were the "Insert" (INS) and "Delete" (DEL) routines which allow one to insert or delete octal code at any point in your program. These routines move the following instructions in memory to make room for the new instructions or close up a space. In addition, the addresser of JMP and CAL instructions are modified as appropriate to maintain proper loop registration. A program to zero memory was added (ZRO). And a program to load ASCII directly into memory (helpful for TVT output) was also developed (TXT).

The use of this operating system in program development would typically involve the following steps:

1) ZRO - zero program storage area
2) Symbolic input of rough program draft
3) TXT - input of any ASCII characters needed
4) DPS - symbolic dump to verify program
   or DPO - octal dump to verify code
5) XQT - execute program as subroutine of monitor
6) SBP - set break point to print out register and flag status at any program step
7) EDT, INS, DEL - these functions can be used to change, insert, or delete, instructions at any location. A particular advantage of this monitor is that on deletion or insertion of instructions the code following is moved up or down in memory and all JMP and CAL instructions are modified to maintain registration.

ONCE THE PROGRAM IS RUNNING TO YOUR SATISFACTION;

8) CPY - moves blocks of data around im memory
9) TRN - changes JMP and CAL arguments to reflect

new page numbers

10) TAS - can be used to store completed program on cassette tape

11) PRG - can be used (if one has the proper hardware) to program EPROM's.

Price: MONITOR-8 $6.00

PLEASE NOTE: This package runs only on 8008-based systems - will not run on 8080.

All the above described programs and games can be obtained either by ordering through the Digital Group or, if your order is exclusively for software, you may order direct from:

Digital Group Software Systems
PO Box 1086
Arvada, CO 80001

Should you have any immediate questions, Chuck or Dianne may be reached at (303) 422-6197. Please try to restrict calls to 9 - 5pm Mon - Saturday.

Watch for the next Digital Group Flyer to find out what's new in the way of available programs and/or games and other interesting items for your microcomputer.

NOTE: All TINY BASIC EXTENDED (TBX-TVCOS) users are encouraged to develop additional programs and games which will run under the system. Users who submit programs and/or games to DGSS with appropriate documentation and who consent to permit DGSS to distribute copies will be paid a small royalty for each copy sold. Submissions must include a cassette tape containing the program or game and typewritten documentation as required. All submissions will be acknowledged but cannot be returned unless accompanied by a self-addressed stamped mailer. Obviously, if many users send in submissions, duplications will occur. In this event, date of receipt AND quality of work will determine which contributor's submission will be utilized. Royalties will be paid quarterly in cash or may be applied to the purchase of products at a discounted rate.

THE ALPHA-NUMERIC MUSIC SYSTEM

by Malcolm Wright

In the January, 1976, issue of *PCC Newspaper*, an article was published on a music program for the 8080 CPU, called "Alpha-Numeric Music with Amplitude Control." Since January, the PCC bookstore has made this 22-page article available for $2. The interest at the Home Brew Computer Club, computer stores, and mail orders has been relatively high. There have been many other music routines written for the 8008 and 8080 CPU's, but Alpha-Numeric Music seems to be getting a great deal of interest from music majors, as well as the general hobbyist. Why?

Looking into the Alpha-Numeric Music (ANM) program one can see many features that don't exist in most of the other routines.

1. An easy-to-follow encoding scheme for writing music. Not just *number look-up tables* for coding, but letters like C, D# (D-sharp), A! (A-flat), for the actual notes to be played.

2. A range of 6 octaves can be played, from about 31 Hz up to 2092 Hz.

3. Different voicing (different sounds) can be specified any time in the musical piece by typing CTRL-E and a number 0 thru 9 to pick the sound.

4. Three different volume levels can be called upon for loud and soft passages of music.

5. The duration of the note can be set from a whole note (W) down to a thirty-second note (T), to satisfy most music fans, any time in the musical measure.

6. The tempo of the music can be changed to three different rates including a nominal 100 beats a minute.

If you have a convenient way of loading programs into your 8080 computer, I am sure you will enjoy this program.

This is a NOTEworthy routine that will MEASURE up to any other music program and help you SCALE new heights in micro-computer enjoyment by BEATing the dull and average software.

Editor's Note: Malcolm Wright is the designer of this system, and the author of *Alpha-Numeric Music with Amplitude Control*. **Please see page 35 for ordering information.**

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
Hunkle	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
Hanurbi	3
King	5
Civil2	7
Market	5
Stock	5
Policy	4
Polut	4
Science Fiction Games	$12
Trader	10
Strtrl	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

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.

SELL: *MIL MOD-8 with MONITOR, TVT-1 with UART and keyboard, Creed TTY, Altair 8800 new kit, unassembled. Send SASE to R. Schultz, 611 N. Dexter, Lansing MI 48910. –5/14/76*

"BASIC JUST WON'T CUT IT.
GET STARTED ON A NEW LANGUAGE."

Dear Jim Warren, May 15, 1976

I think you are wasting good space with still another version of BASIC. BASIC just won't cut it.

Suggest you get started on a new language for hobbyists with the best features of many languages. Should come in various sizes, always upward compatible. Also, it should be interpretive for fast programming, and compilable for fast execution.

Bill Roch

24365 Clipstone
Woodland Hills CA 91364

HOW 'BOUT SMALL PASCAL?

Bill,

I fully agree. You're asking for a lot in requesting various sizes + upwards compatibility + interpretable + compilable, but I hope to offer at least part of that in our next HLL (High Level Language) for micros. Unless someone else volunteers to initiate a HLL project through the *Journal* in the next month or three, I hope to start a build-your-own-compiler project called SMALL PASCAL. My plan is to detail the design of each module of a compiler in a series of articles. The compiler will be for a block-structured PASCAL-like language, trimmed for microcomputer size, and modified as appropriate for a presumed interactive, stand-alone environment (as opposed to a batch system). At this point, expandability and interpretability are low-priority goals for this project, but I'll keep them in mind.

I definitely wish to "push" a "good" HLL, instead of continuing emphasis or reinforcement of BASIC. BASIC is better than nothing (and, currently, nothing else is available for hobbyists). However, I see no reason to continue to eat 19¢ hamburgers when classy filet mignons are available at the cost of a little interesting effort. –JCW, Jr.

DR. DOBB'S JOURNAL OF COMPUTER CALISTHENTICS AND ORTHODONTIA is published ten time per year, monthly except in July and December.

U.S. Subscriptions:

☐ $1.50 for a single copy: Vol. ____, No.____
☐ $3.00 for the first three issues
☐ $10.00 per year (10 issues/year): Begin with Vol. ____ No.____

For foreign subscriptions:

☐ *add* $4.00 per year for surface mail, or
☐ *add* $12.00 per year for air mail

Payment must accompany the subscription. We do not invoice for subscriptions or single orders.

Send to: PCC
P.O. Box 310
Menlo Park, Ca. 94025

Necessary Information:

Name (last name first) _____

Mailing Address _____

_____

City _____ State _____ Zip Code _____

☐ yes ☐ no: This information may be published in directories and lists of individuals interested in computers in non-commercial environments.

Optional Information:

Equipment that you have or are planning on purchasing, immediately:

Make & model _____ Manufacturer _____

CPU model _____ CPU Manufacturer _____

I/O Devices _____

Mass storage peripherals _____

_____

_____

_____

Primary areas of interest concerning non-commercial and home computers:

_____

_____

_____

_____

_____

Questions: What would you like to see published in **DR. DOBB'S JOURNAL**? It will help guide us if you will rate these, 1 to 10 (1 – minimally desire; 10 – super-eager to see) or 0 (would prefer we not waste space publishing it).

____ Schematics and acticles from all of the computer club newsletters
____ Short news articles directly related to home computers
____ Short news articles concerning computers in general, particularly their social implications
____ Indices to all articles in all other computer hobby publications
____ Indices to selected articles from other computer, electronic, and trade publications
____ Letters having technical, critical, or entertaining content
____ Classified ads (as opposed to display advertising)
____ Suggestions and "blue skying" about what can be done with home computers in the foreseeable future.

OVER

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 Wright. 1975. 23 pp. $2. Xeroxed.

BIOFEEDBACK: Turning on the Power of your Mind
Marvin Karlins & Lewis Anderson. 1973. 190 pp. $1.25.

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

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

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

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

THE BEST OF CREATIVE COMPUTING, Vol. 1
David Ahl, editor. 1976. 328 pp. $8.95.

BODY TIME
Gay Gaer Luce. 1973. 411 pp. $1.25.

THE BUGBOOK I & II with INSTRUCTOR'S WORKBOOK
Rony, Larsen, & Braden. 1974. $16.95. 2 volumes + workbook.

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 Sage. 1975. 360 pp. $5.95.

GAMES, TRICKS, & PUZZLES FOR A HAND CALCULATOR
Wallace 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 Hunter. 1974. 204 pp. $4.95.

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

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

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

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

MICROPROCESSOR/MICROPROGRAMMING HANDBOOK
Brice Ward. 1976. 294 pp. $6.95.

NEW MIND, NEW BODY; BIO-FEEDBACK: New Directions for the Mind
Barbara Brown, Ph.D. 1974. 523 pp. $2.50.

101 BASIC COMPUTER GAMES
David Ahl, editor. 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.

PROF E. McSQUARED'S (ORIGINAL FANTASTIC & SATISFYING CALCULUS PRIMER)
Swann & Johnson. 1975. 111 pp. $2.95.

PROGRAMMING PROVERBS
Henry 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.

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

TV TYPEWRITER COOKBOOK
Don Lancaster. 1976. 256 pp. $9.95.

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

THE UNIVERSAL TRAVELER
Don Koberg & Jim Bagnall. 1974. 128 pp. $4.95.

WHOLE EARTH EPILOG
Stewart Brand, editor. 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.

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.

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