dr. dobb's journal of COMPUTER Calisthenics & Orthodontia

$1.50

Running Light Without Overbyte

June/July, 1976 Box 310, Menlo Park CA 94025 Volume 1, Number 6

A REFERENCE JOURNAL FOR USERS OF HOME COMPUTERS

CONSUMER COMMENTS

SOFTWARE

CORRECTIONS & IMPROVEMENTS

VIDEO OUTPUT

MISCELLANEOUS HARDWARE

RANDOM DATA POINTS

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 6; June/July, 1976
Box 310, Menlo Park CA 94025
Copyright © 1976 by People's Computer Company

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 centerfold.)
$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.

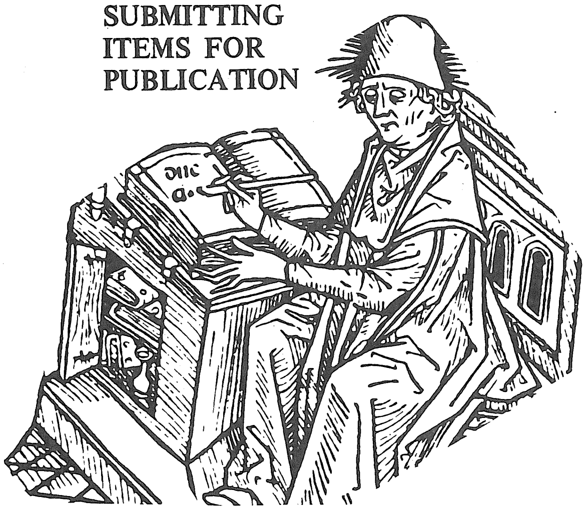
Disclaimer

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

Reprint privileges

Articles herein that are copyrighted by individual authors or otherwise explicitly marked as having restricted reproduction rights may not be reprinted or copied without permission from People's Computer Company, or the authors. All other articles may be reprinted for any non-commercial purpose, provided a credit-line is included. The credit-line should indicate that the material was reprinted from Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Box 310, Menlo Park CA 94025.

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.

DO YOU . . .

. . . LIKE WHAT WE ARE DOING?

* Publishing significant systems software, *every* month
* Reprinting materials from club newsletters
* Proposing & detailing "realizable fantasies" . . . exciting projects, feasible for home computers
* Actively pursuing a role of consumer advocate
* Publishing useful references . . . indices to periodicals, bibliographies, lists, etc.
* "All Meat" pages; we are not accepting commercial advertising
* And more–

. . . KNOW THAT *MUCH* MORE MATERIAL IS BEING SUBMITTED
THAN WE ARE FINANCIALLY ABLE TO PRINT?

* Many more programs than we have room to print
* Much more very useful material from *many* club newsletters
* A *number* of projects that are practical & appropriate for home computer users
* More consumer evaulations of products & services
* Many, many more reference lists, indices, tables, etc.
* Much more

. . . KNOW THAT YOU CAN HELP US
TO BE ABLE TO PUBLISH ALL THE GOOD THINGS WE ARE RECEIVING?

* Since we . . .
    - are supported entirely by subscriptions & sales through stores
    - want to keep it that way ("keeps us honest" when we indulge in consumer advocacy)
    - are serving *you*; not serving commercial advertisers
* Then . . .
    - the only way we can get more income to pay our printers to print more pages, is to have more people and companies subscribe to and purchase the *Journal*
    - *you* have already helped by purchasing this issue

. . . WISH TO HELP US HELP YOU?

* Tear out the center-fold (not very sexy, but we hope it's attractive)
    - pass it along to a friend or professional associate
    - post it on the bulletin board at school or at work
    - give it to a manager of microprocessor software or design
    - reprint it in your club newsletter
* Stand up and tell your next computer club meeting about the *Journal*

[and . . . if you *really* like what we are doing:]

* Send *tax-deductible* contributions to People's Computer Company
    - do so as a company or an individual

[Oh . . . didn't you know? PCC, the publisher of *Dr. Dobb's Journal* is a legitimate, state and federally chartered, non-profit, educational organization. Contributions to it are tax-deductible.]

PRAISE FOR PITTMAN'S 6800 TINY BASIC and A Minor Complaint . . . With Tom's Response

Dear Bob, May 17, 1976

I bought Tom Pittman's 6800 TINY BASIC and think it's the best $5 I've spent in a long time. I haven't tested it exhaustively, but it seems to work admirably, though slowly. The user's manual that came with it was simple and comprehensive, and gave enough info to make the program run on anyone's system with a minimum of fuss. Mine worked almost as soon as I got the paper tape read in. Tom Pittman is to be applauded not only for producing a good TINY BASIC that uses less than 2K, but for doing such a good job of explaining how to use it. If all hobbyist vendors conformed to Tom's standards, there would be far fewer complaints.

My only complaint about Tom is that he staunchly refuses to release the source listing of his program. I need to make some modifications to the program, for use with my cassette O/S, and I would like to be able to expand it. It is very frustrating to be kept so ignorant about his program, particularly since it seems to work so well. He seems concerned about his ability to retain control over the integrity of the program, and perhaps about the investment in time and potential money he expects to receive from it. I can't see how he'll ever make enough money (at $5 a copy) to keep himself in business. But the price may serve to discourage people from circulating clandestine copies of the program. Anyone who uses Tom's program without paying Tom for the privilege, should be tarred and feathered.

Sincerely,

David M. Allen
1317 Central Ave.
Kansas City KS 66102

Dear Jim: 11 June '76

I have to agree with David's complaint—I would be very unhappy to find the tv I just bought did not have a schematic, but then a $5 transistor radio is something else. Though he does not seem to realize it, David has actually touched on the reason why I have not made source listings available.

When I first started this venture, I too was not sure I could make enough money to stay in business; it was in fact a sort of experiment in economics. Therefore, as a hedge against possible losses, I sold copies of the source and maintenance documentation to a company for a lot more money than any hobbyist would be willing to pay, though it was still considerably less than I usually sell custom programs of comparable size for. While the sale was non-exclusive, I do not think it fair to devalue this company's investment when to some extent they helped make Tiny possible.

Aside from that one large sale, Tiny BASIC 6800 has not yet paid for itself, but the promise is there, so I expect to go ahead with other software for the hobbyist in the same price range, and I hope to make source listings available for the new packages. As for Tiny BASIC, I am presently preparing a comprehensive description of the IL (which is substantially the same as that originally published in *PCC*) including instructions for modifying it to add functional capability or change syntax, to be published in *DDJ* (if you will have it). I had hoped to include an assembler written in Tiny (yes, I know it's slow), but already I am time-sharing my efforts between this, new software, and those expensive custom programs that keep the rent paid; the projects with non-zero financial return seem to get higher throughput.

**Tom Pittman
Box 23189
San Jose CA 95153**

P.S. Your readers may appreciate being made aware of the fact that Tiny BASIC does run on a Sphere configuration, but they should mention which computer they have, since the code is slightly different for the different operating systems (e.g., Sphere vs. SWTP, etc.).

4K STATIC RAM BOARD (UNPOPULATED) FOR $18.75

Dear Friend, May 10, 1976

I would appreciate your disseminating the spec sheet enclosed to your friends and club members. A discount of 5% will be given to clubs with an order of 50, and 10% on 100.

Several months ago I received several inputs on making an unpopulated 4K RAM board, hence I am producing the board for the hobbyist that does not want to get ripped off.

The board has been fully tested and is in use by many people here in Dallas. I might add that it is in use on 8080, 6502, and 6800 CPU's.

Sincerely yours,

Jim Garrett
Micro Applications
Box 2161
Garland TX 75041

4K STATIC MEMORY BOARD (unpopulated)

FEATURES

- 2102 and 91L02 compatible
- User selectable options
  - Protect/Unprotect switch
  - Battery backup
  - Selection of address by dip switch
- Fully compatible with MITS/ALTAIR and IMSAI 8080
- Can be used with other micro/homebrew computers
- Full buffering of address and data lines
- Bipass capacitors on all ICs for improved noise immunity

SPECIFICATIONS

- Double-sided MIL-spec board
- 100-pin (50x50) on 0.125-inch centers
- Standard dimensions
- Plated through holes
- Gold plated edge contacts

GENERAL DESCRIPTION

This is an unpopulated 4,096 word (byte) Random Access Memory. *The cost* to populated is *less* than any kit available (based on advertised prices). Full instructions, schematics and parts list are included.

PLUSES

- 100% tested
- Instruction package
- Plated through holes and gold-plated edge contacts
- Uses 2102's or 91L02's

PRICES

- 1-3 @ $18.75 each
- 4 or more @ $16.25 each
- $1 for instruction package (*one* is included with each order)
- Texas residents add 5% tax

DELIVERY

- 3-4 weeks

Coming Soon: "The Extender."

We are interested in receiving consumers' compliments and complaints concerning Micro Applications, and all other large and small marketeers to the hobbyist community. --JCW

DENVER'S DIGITAL GROUP KIT DRAWS PRAISE

Dear Bob: April 26, 1976

I finally broke down and bought myself a system. I took out a bank loan, added some cash of my own and mailed my cashier's check to the Digital Group for their Three-board system.

Three days later I read in *PCC* that caution was needed in dealing with DG. I also read some mixed reports in *Micro-8 News*. I was really nervous, had bad dreams, and didn't sleep for nights.

2/10/76	Order placed with DG: 3-board system kit plus power supply. They promised 3-week delivery. Order placed with Herbach & Rademan: Clare/Pender Keyboard.
2/21	The three boards arrived, missing 74121 and 22uF capacitor.
2/25	Keyboard arrived.
2/26	Power supply arived.
3/3	Mother board arrived.

Total time: 3 weeks, one day. The missing parts took 4 weeks and two letters.

All parts are of good quality.

TV-Cassette and Mother boards are slightly warped.

5V 6A supply by Eentak Inc.; looks impressive.

Documentation fair; assumes a lot. Several minor errors.

Chassis, switches and connectors need to be ordered from other distributors at present.

Time Spent	
2.5 hours	I/O card
4.75	CPU card
3.25	TV-Cass. card
2.5	– 5, ± 12V power supply
.5	Mother board
6.5	TV modifications
17.5	Planning, cutting, mounting, wiring chassis
4.5	Checking things out

41.5 hours	Total time

I took my time and spread it over about 40 days. I must say I savored every minute of it.

I had trouble with the TV characters being out of focus. It finally dawned on me, after scratching my head for several days, that the TV interface was overdriving the TV video. I solved the problem by turning the contrast and brightness to zero. Later I plan to add a pot on the interface output.

When I had gone through their checks, I turned the system on and sure enough there was a message on the TV screen: "Read 8080 Initialize Cassette."

After dancing around the room, I proceeded to read in the cassette. Numbers flashed across the screen. First 1's, then 2's, and finally 7's, then a bunch of dots. The dots weren't suppose to happen. More scratching of the head and several days later I decided it must be the cassette recorder.

I borrowed a recorder from the school to replace my El Cheapo and everything happened just like it was supposed to. However, it still misses a few bits now and then. The 1100 baud rate is too fast for my El Cheapo. It looks like it would be possible to set the cassette read and write constant at a lower baud rate, re-record the Operation Monitor, and then every time the system is turned on key in the new constant from the front panel, and then read in the cassette on an El Cheapo. However, the DG system does not come with a front panel, just plans for one.

I've been spending most of my time figuring out what makes the Monitor work. The DC documentation is not much help. I've also found out that machine language is a far cry from Fortran.

I will echo what some others have said about the DG system:

> It does what they say it will do.
>
> It worked the first time I turned it on, which says a lot considering how complex it is. It's definitely not a beginner's kit.
>
> More documentation: flow charts for the Monitor (I'm working on a set), clearer instructions, spec sheets for the IC's and a better description of how it works would be nice. But that would mean more money and maybe in that case the documentation is OK.

Last week I got info about DG's Tiny BASIC. I plan to order that and another 8K of RAM from them.

Materials I'm finding helpful: *The Bugbook III.*

Hopefully my *Intel 8080A Users' Manual* will get here soon.

I want to get: *8080A Microcomputer System's Manual, Intellec 8/Mod 80 Microcomputer Development System Reference Manual*, SCELBI software manuals.

Keep up the good work.

Yours,

Ed C. Epp

Freeman Junior College
Freeman SD 57029

GOOD REPORTS ABOUT MOS TECHNOLOGY

We hear that MOS Technology has sold about 1000 KIM's. We also hear that they are very responsive to customer queries. If you have needs or interests, the "good guy" name we have been given is Don McLaughlin, Product Manager for KIM, (215) 666-7950.

PLAUDI~~TS FOR~~ MOS TECHNOLOGY

Dear Jim, May 11, 1976

Just received my third issue of *Dr. Dobb's Journal* and I thought I'd drop you a note of thanks for putting in an article on a MOS 6502. From the lack of MOS Technology articles, I got the feeling that "Intel Valley" was banning MOS Technology products in California.

Last week I called Intel, to get a software manual, and received the biggest runaround I have ever gotten. Unlike Intel, I have found MOS Technology will answer any and all questions on their products and it only takes one call to them to produce results. Many times I have called Will Mathis and Don McLaughlin of MOS Technology with what I, today, would consider to be stupid questions and received the time and help of their technical staff in getting me on the straight and narrow.

MOS gets a number one in my book and should be given more space in your *Journal.*

Very truly yours,

Gerald D. Severson

RUMOR: 16-BIT, 3-MEGAHERTZ MICRO?

We hear that MOS Technology is planning to exhibit a 3-megahertz, 16-bit microprocessor at this Fall's Wescon convention in San Francisco. They also expect to have a "rotate right" instruction in their 6502 by the time this issue reaches your hands.

GOING TO SUBMIT A PROGRAM TO A *MANUFACTURER'S* SOFTWARE LIBRARY? WHY NOT SUBMIT A COPY TO *YOUR* LIBRARY (THE *JOURNAL*)?

Dear Editor, May 12, 1976

I've got a gripe. I ordered my Altair back when *Pop 'Tronics* first published the article (took four months to get it, though). Anyway, they automatically gave me a 1 year's subscription to their *Computer Notes.* With each copy they include at least one page of *new* programs available in their *software library*. These are programs that they keep asking people to send in, for which they receive a couple of programs in turn *from* the library. We are told these programs are all free simply for a small handing and copying fee. These "small fees" are almost all $2 minimum—or more for the longer programs. The current offering consists of 29 programs totaling $61 in fees if one ordered all 29 (28 are $2 each, and 1 is $5).

What I don't understand is why they don't publish them in their *Computer Notes*. As it stands, *Computer Notes* consists of 16 pages of virtually nothing but their own advertising. They say a subscription to non-Altair owners is $30. Personally, I wouldn't give 30 *cents* for a year's subscription. I bet after the 12th issue goes out and they start selling subscriptions they are going to be in for a big surprise. Who will pay even $10 for their advertising sheet? They also issue software with a $500 price tag to the hobbyist and then lament the swapping, passing around, trading, of it. What do they expect? But that's another story.

The point is—why can't you publish these (or similar)? Just glancing at the latest list I see programs listed as being 60 lines, 83 bytes, 73 bytes, 21 lines, 46 lines, 121 bytes, 28/33 bytes, 56 bytes, 12 lines, 250 lines, 15 bytes, etc. You could get all eleven programs I listed on 3 or 4 pages, and that represents $22 of handling and copying fees. Publishing some would save us a bundle.

This is what I hope *Dr. Dobb's Journal* is all about. Actually, a lot of people just aren't going to get many of the programs unless there is a lot of the lamented swapping, trading, and exchanging—or, unless you become the "library" for all of us.

So far, you are doing fine. Keep up the good work.

Durward Landers
2509 Lakeside Dr.
Garland TX 75042

We will publish as many programs as we can, *if people will send them to us*. Spread the word: Whenever someone decides to submit a program to a manufacturer's software library or users' group, encourage them to also submit it to the *Journal*. If it's systems software or assembler-level, we will probably publish it. If it's a program coded in BASIC or some other HLL (High Level Language), it will probably be published in *PCC Newspaper*.

We see nothing wrong with offering programs to *manufacturers'* libraries. But at the same time, why not offer them to *your* library: *Dr. Dobb's Journal*?

As far as reproduction and postage costs are concerned: there *is* a problem. The Community Computer Center (CCC) is maintaining a Program Repository and Duplication Facility (see the *Journal*, March, 1976, issue) for all programs submitted to it. We think their charges are reasonable: $1/ounce for tapes plus 50 cents (orders under $5) or $1 (orders exceeding $5) for postage and handling. Note that these are quite comparable to MITS' charges . . . and we *know* CCC is doing little more than breaking even. Unfortunately, there is a lower limit on the cost of maintaining paper-tape equipment + purchasing supplies + paying a paltry pittance to a slave to operate the equipment and verify tapes that are punched and . . . etc. (Note: People's Computer Company is a publishing operation. We provide programs in *human*-readable form. We do *not* provide programs in *machine*-readable form, e.g., paper-tapes.)

ACCENTUATE THE SYSTEMS SOFTWARE; ELIMINATE THE GAMES

Dear Editor,

You can eliminate 90% of the games. Almost all other hobby publications specialize in them. Emphasize your uniqueness: a repository for systems software. It's a great idea, so far well executed; so don't drop the ball by trying to cover too many other things. I strongly recommend that you push APL as you did TBASIC

Robert C. Minnick
Box 306
Ouray CO 81427

To a large extent, we will leave the games for publication in *People's Computer Company*. We will reprint games from time to time, particularly when they are "games' systems," or are games written in assembly-level code. This might be considered to be, so to speak, systems software for home computer users. *Dr. Dobb's Journal* will definitely *not* be emphasizing games, however.

We would be delighted to "push" APL as we did Tiny BASIC. All we need is for someone to provide design criteria and details appropriate for hobbyist consumption. We are alwsys on the lookout for competent individuals interested in providing the leadership for such projects. Incidentally, as soon as he can find the time, the Editor of *Dr. Dobb's Journal* is planning on initiating a SMALL PASCAL project, to be pursued in much the same manner as Dinnis Allison's Tiny BASIC project. This will be based on Niklaus Wirth's PASCAL, a cleanly designed, excellent, block structured, high-level language similar to ALGOL, but with much more powerful data description and manipulation facilities, and structured for single-pass compilation.

One final note: PCC is not a program repository. We publish all available information about interesting software, including information as to how it may be obtained. However, we do *not* distribute such software in machine-readable form (e.g., paper tapes, cassettes, etc.). --JCW

SHORT ON LENGTH *BUT* LONG ON QUALITY

Dear Jim,

Having just read your February issue (Vol. 1, No. 2), I was sufficiently impressed to part with the money for a subscription. What your publication lacked in length was more than adequately offset by quality and subject matter. Your questionnaire scares me somewhat as you apparently are looking for some *new* directions. Additional coverage of other topics is fine and may tend to broaden your base of appeal. However, I for one, bought your publication for what it currently is—"a medium concerning the design, development and distribution of free and low-cost software." Should your enterprise maintain its stated goal of presenting "detailed information concerning low-cost systems software," I will have spent the subscription fee well.

I remain,

Dan Artman
1445 Adams Rd
Cincinnati OH 45231

A novice constructs an IMSAI

An attorney builds his first computer

by S.A. Cochran, Jr.

I am a little out of my field messing about with computers—far more than some school teachers are whose interest is in propagating math instruction, etc. But even my life was not untouched by some of the manifestations of the computerized society—about four years ago, I made use of the IBM Mag-card Selectric typewriter during a period of heavy work. Ever since, I had been struck with the convenience—and high cost—of mechanized typing.

More recently, I heard that John Arnold and Dick Whipple were assembling a computer for what appeared to be peanuts, compared to the charge gaily levied by IBM for its typing units, much less its Mag-card units, and still less than its computers. Based on this information, I could hope to install a powerful typing system with greater capabilities than anything that I could expect to purchase from IBM with available resources, and at less cost.

Having decided to get into the microcomputing stream, with the help of John Arnold, I decided to get an IMSAI rather than Altair because the IMSAI unit *with* memory was the same price as the MITS unith without memory. Also, MITS' prices for memory were substantially above those charged by IMSAI.

I placed my order for the basic IMSAI unit on January 22nd. They received this order on the 25th, and the unit was actually shipped on February 2nd. I learned more about the units actually available from IMSAI on January 25th, and sent in an additional order on that date. It was not filled until March 1st, when some of the parts were shipped. The I/O ports that were included in the second order did not arrive until about March 25th.

The serial I/O board was delayed by a considerable re-design of the board, which must have started in January, and must have concluded at the very end of February. The documentation received with the original equipment showed the manner of assembling the SIO 2-2 board, Rev. 1. I received at least one set of errata with the documentation, and one after I had already got the equipment. Ultimately, IMSAI sent me their SIO 2-2 Rev. 3 board, with all of the changes built into the board.

I would like to point out that IMSAI was very prompt in providing the kit buyers with errata when they discovered something that needed to be fixed. In addition, on certain rather complicated modifications, they offered to make the modifications themselves if the kit-builder did not trust himself to fix the unit satisfactorily. They have also been quite helpful with software for units of the equipment. For instance, with the CRI board they supplied paper tape software and a hexadecimal listing.

In the revised order, I had requested the EXP-22 mother board. I recall that I could not proceed beyond the assembly of the several independent boards during February while waiting for this unit to arrive.

I had a little confusion about the proper procedure for completing the power supply and collecting it. I had documentation for connection of the IMSAI power supply using two alternative transformers and they had shipped a *third* version of the main power supply. This was corrected quickly enough, and a minor problem with the 1K memory board was quickly corrected when someone pointed out that I had interchanged a .01 mfd capacitor and a 33 mfd one. Testing of the front panel board and the cpu board had to wait for arrival and assembly of the mother board and additional memory.

When the additional units arrived, everything tested out satisfactorily, except that there was a single bad LED on the front panel. I recall that there was an embarrassing pause after this LED was replaced—we thought that the entire equipment had gone berserk. However, I found that a piece of wire had worked its way behind the front panel, and was shorting the deposit switches. I have had no further problems with the computer, or with any of the parts supplied by IMSAI—except for the problems involved in learning to speak machine language like a native. (Apparently I don't do that yet.)

My remaining difficulties in getting the initial system into operation have revolved around input and output devices. I joined John Arnold and Dick Whipple in the acquisition of three Burroughs Model 9350-2 communicating typewriters from Herbach & Rademan of Philadelphia. These units were correctly advertised as receiving and transmitting a form of ASCII. They appear to be based on the Friden Model 2300 typewriter, a modernized version of the Flexowriter. They are not readily convertible to use as a computer input because there is a direct mechanical linkage between the keyboard and the keybars of this typewriter. Another thing that I found very hard to get used to was that this "typewriter" didn't have a backspace key! There was some additional major maintenance to be done on this equipment. Although it could be induced to type, thus far I have been unable to get the typewriter hooked up to the computer!

After making the decision to use a separate keyboard, I bought one of the keyboards originally built for RCA that have been advertised by Sargeant's in Los Angeles. This keyboard was advertised to be fully ASCII encoded, and it was, so far as it went. Unfortunately, this unit had provision for upper and lower case letters, numbers, and punctuation marks, but it did not have any provision for the non-printing control characters so common to computer work. In addition, upon applying power to the keyboard, we discovered that this keyboard carried a strobe that was valid as long as the key was pressed, and used negative logic. That is, the strobe output, and all the other outputs supposed to be made true when a key was pressed, went at that point from a voltage of 5.0 to 0.4 volts. It appeared that it would be necessary to add a fair number of IC's to the interface between the keyboard and the computer in addition to installing an additional key on the keyboard for use as a control key. With all these matters before me I decided to keep the keyboard for future modification, and get another for my present use. But I did get a pretty keyboard enclosure from Sargeant!

[Later] . . . I am now in the process of putting that pretty keyboard enclosure and keyboard to good use. It's going to take a certain amount of skill and understanding but one of my purposes in getting into this hobby was to acquire that sort of skill. Thanks to Sargeant's, any way, for providing me with an occasion for that sort of acquisition—even if it wasn't what I exactly expected.

I feel that I should mention the question of IMSAI software before closing. In the advertisements that they began to distribute just after I ordered my IMSAI unit, they stated that they would ship an assembler, loader, and monitor with every unit, together with BASIC and other languages thereafter. This assembler, etc., turned out to be a re-write of the assembler originally distributed by Processor Technology Corp. It uses all of a 4K memory, and needs an additional 2K of RAM, if not more. A complete source listing and paper tape of this assembler were enclosed with the unit. IMSAI also provided a listing and paper tape of software for their Cassette Recorder Interface board. On March 20th, IMSAI wrote all of their customers, stating that they were now ready to deliver their 4K BASIC, and expected to be ready to deliver the 8K and 12K BASIC languages on April 15, and May 15, respectively. The 4K BASIC was shipped at the end of March. I was ultimately charged $4.00 for their cost of duplication of the paper tape, and an additional $10.00 for a 70-page source code listing of the IMSAI BASIC. IMSAI had apparently enclosed it 'by mistake.'

IMSAI's price for its 4K BASIC thus amounts to $14. In addition, IMSAI will sell the 8K and 12K BASICs for $1 per kilobyte of memory required. The source code listing for these two extended BASICs will again be $2.50 per kilobyte. Compared to the longwinded philosophical discussions that one hears from MITS from time to time, this is probably a great bargain, notwithstanding that the IMSAI BASIC may not be quite as powerful as the MITS 4K BASIC.

After acknowledging the assistance of my friends in checking out the IMSAI 8080, I conclude that this equipment is a well-designed, sturdy unit easily capable of expansion to the full limits of addressable memory. IMSAI has acted in a very businesslike fashion, and has tried to be genuinely helpful within the limits that are proper to a business organization. IMSAI recently raised the price of the basic equipment, without memory, to $599. Certain persons of my acquaintance griped very strongly at IMSAI's action. I consider that in view of the high quality of the merchandise, the IMSAI equipment is worth this premium price to the individual who has never attempted to build an electronics kit before. Anyone who considers the IMSAI not worth the price, should consider whether he or she could duplicate the system with available resources. If he could match the high quality provided by IMSAI, could he deliver the goods to others, at the price? If so, why isn't he in there competing?

Yours very truly,

S.A. Cochran, Jr.
Attorney at Law

Box 607
Tyler TX 75701

Bootstrap for 8080

by Lichen Wang

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

If your 8080 microprocessor system is not equipped with non-volatile memory, you probably have to reload the memory from time to time. To read the Intel hex-format paper tape, you need to key in a loader of some eighty-odd bytes long. This is rather tedious and often leads to error. Altair BASIC has a bootstrap loader of twenty or twenty-one bytes long. In principle, you can use this bootstrap to load in your own loader which will then load in your program. I coded one myself, and what comes out is a bootstrap sixteen bytes long. This is still too long—maybe our professional experts can make it shorter. For the time being, you are welcome to copy mine.

The part that you have to key in looks like this:

```
0000 DB00    READ   IN   0        ;READ AND
0002 E620           ANI  20H      ;MASK THE STATUS BIT
0004 CA0000         JZ   READ     ;NOT READY YET
0007 DB01           IN   1        ;READY, READ IN A
                                   FRAME
0009 010900  HERE   LXI  B,HERE   ;LATER BECOMES INX B,
                                   STAX B, CPI
000C 02             STAXB         ;LATER BECOMES FF
000D C30000         JMP  READ     ;LATER BECOMES JNZ
                                   READ
```

And the paper tape should have the binary equivalent of the hex numbers shown below:

```
01 01 . . . 01 03 02 FE FF C2 00 00 XX XX . . . . . XX XX FF
<- leader -> <- bootstrapping -> <- your loader -> marker
```

Where your loader is punched in binary format on the paper tape between the 00 and the FF is denoted by XX XX . . . . . XX XX. Your loader cannot have any byte with the value FF. The marker FF tells the bootstrap to start your loader, starting at 10H. After the FF, the paper tape is read by your loader. Use whatever format you want.

If your loader cannot be loaded at 10H, then you will have to write another loader which can be loaded at 10H. Use it to load in your first loader to load in your program. This sounds very confusing, but that is how bootstrap works. Have you ever tried to get yourself off the ground by pulling your bootstrap?

Incidentally, the I/O ports at locations 1 and 8, the status bit mask at 3, and the jump condition at 4 may have to be changed for different I/O interface board. Your loader should copy them from the bootstrap rather than setting them up on their own. (Or, you can code your loader to change location 9 to RET, and use READ as your input routine.) This way the same paper tape can be used on different machines. To carry this one step further, your program should, in turn, copy them from your loader, so that it too can work on different machines.

HIGH SCHOOL CLUB IN CHICAGO

The University of Chicago Laboratory High School (1362 E. 59 St., Chicago IL 60637) has started a computer club.

BYTE SAVING PROGRAMMING TRICKS FOR THE 8080

by Tom Pittman
(reprinted with permission from
Homebrew Computer Club Newsletter)

These are some programming tricks I have accumulated over the years which can often save a byte or two in 8080 programs. Because of the peculiarities in the instruction sets, only a few of these also apply to 6800 programs and are so noted. Many of these tricks are widespread lore; some I have never seen elsewhere. I hope they can help you as well.

For 2's complement signed arithmetic, it is sometimes necessary to add a signed 1-byte number to a larger format. There are also other reasons for spreading a single bit (in the Carry FF) to a whole byte (in A). I found this one in the Scelbi book:

```
SBB A       Copy carry to all bits in A
```

The 8080 does not have a proper shift instruction which fills the vacated bits with zeroes. Normally, a *CLC* must precede the *RAR* instruction. However, for left shifts:

```
ADD         Shift with zero insert
```

To insert a single bit (in the Carry) into the left or right end of the A without altering the other seven bits:

```
RAL         Remove old left bit
RRC         Insert new from Carry
```

The right-end version is symmetrical. To divide a signed (2's complement) number in half, it is necessary to keep the sign bit (bit 7) unchanged while shifting A right. The 8080 does not have an instruction for this, but the *RAR* may be used if the Carry can bet set up to match the sign bit:

```
RLC         Copy bit 7 to Carry
RRC         Restore A
```

The 6800 has a single instruction for signed right shifts, but no circular rotate. To copy a sign into the Carry:

```
ASR A       (6800) Duplicate bit 7
ROL A       Restore A with bit 7 in Carry
```

Some of these other tricks with the Carry become more useful if the Carry can be set on the basis of the other conditions. A zero in A may be converted into either a one or a zero in the Carry (so that non-zero is the reverse) by one of the following instructions (this also works in the 6800 with appropriate opcode substitutions):

```
ADI 0FFH    C=0 if and only if A=00
SUI 1       C=1 if and only if A=00
```

It is easy to get the sign of A into the Carry (any left shift will do); to get the complement of the sign is a little trickier. This instruction leaves the contents of A unchanged, and also works for the 8080:

```
CPI 80H     Complement bit 7 to Carry
```

Finally, how do you pack a byte with some bits from A and some bits from B? The Univac 1108 has a special instruction called *Masked Load Upper* which does this. The 8080 (and also the 6800—but only when the second byte is in memory) can do this in three instructions! Assume that the data in A and B (or any other register or memory location) are already in the correct bit positions. The mask represents a byte with the ones where the data in A is to be substituted; the non-data bits of A and B may contain garbage, as they are ignored:

```
XRA B       XOR B to A data bits
ANI Mask    Delete A garbage
XRA B       Insert B data
```

The theory behind this trick lies in the fact that the *XOR* operation may be considered a "selective complement" instruction. In other words, where there are ones in B the bits in A are complemented, and where there are ones in B the bits in A are unchanged. The AND operation, on the other hand, may be thought of as selectively setting bits to zero in A, where the zeroes in the mask set bits in A to zero and ones in the mask leave the bits in A unchanged. Assume for the moment that the mask is all ones; the other two instructions exactly cancel each other, leaving A unchanged, since the ones in B complemented the corresponding bits in A the first time and recomplemented the same bits (back to their original states) the second time. Thus ones in the mask retain the original bits in A. Now consider zeroes in the mask: here the corresponding bits of A are cleared to zero by the AND operation so that the first *XOR* has no effect; the second *XOR* simply complements those zeroes in A which correspond to ones in B, which is to say that it copies the bits of B into A (remember A was cleared to zeroes by the AND operation). Thus zeroes in the mask copy in bits from B. Since each bit operates independently, there is no requirement that the selected bits of A or B be contiguous. Note also that no other registers or memory is required for this procedure, and that B is unchanged. I realize this operation looks suspicious, so I have included the following truth table:

FIGURE 1
Byte Packing Truth Table

A	B	MASK	1st XOR	AND	2nd XOR	
0	0	0	0	0	0	= B
0	0	1	0	0	0	= A
0	1	0	1	0	1	= B
0	1	1	1	1	0	= A
1	0	0	1	0	0	= B
1	0	1	1	1	1	= A
1	1	0	0	0	1	= B
1	1	1	0	0	1	= A

```
                ;
                ;   PROCESSOR TECHNOLOGY REFORMATOR
                ;
                ;   THIS PROGRAM TAKES 8080 ASSEMBLY
                ;   SOURCE PROGRAMS WRITTEN ON INTEL'S
                ;   INTELLEC 8 WHICH HAVE COLONS AFTER
                ;   LABELS, CONTROL-I'S FOR TABS,
                ;   AND SEMICOLONS TO DENOTE COMMENTS.
                ;
                ;
                ;   IT CONVERTS THEM TO PROCESSOR
                ;   TECHNOLOGY'S FORMAT WITH LINE
                ;   NUMBERS, '*' TO DENOTE COMMENTS,
                ;   AND NO SEMICOLONS AFTER LABELS.
                ;
                ;
                ;   THE READER MUST BE UNDER PROGRAM CONTROL.
                ;   THAT IS IT MUST BE STOPPED AFTER EACH
                ;   CHARACTER IS READ IN.
                ;
                ;
                ;   THIS RUNS ON THE INTELEC/8
                ;   IT STARTS AT LOCATION 10H
                ;   AND USES THE INTEL MONITOR
                ;   FOR I/O
                ;
000D            CR       EQU      0DH
000A            LF       EQU      0AH
3806            RI       EQU      3806H    ;READER INPUT
3809            CO       EQU      3809H    ;CONSOLE OUTPUT
                ;
                ;
0000                     ORG      10H
                ;
                ;
0010 310001     START:   LXI      SP,0100H        ;INITILIZE STACK
0013 CD8100              CALL     CRCHK    ;INPUT A CHARACTER
                ;
                ;
                ;   PRINT OUT 4 ASCII DECIMAL DIGITS
                ;
0016 F5         MDEC:    PUSH     PSW
0017 21A900              LXI      H,DNUM+3
001A 7E         MD1:     MOV      A,M
001B 3C                  INR      A
001C FE3A                CPI      '9'+1    ;TOO BIG?
001E C22700              JNZ      MD2
0021 3630                MVI      M,'0'
0023 2B                  DCX      H        ;DO THE NEXT DIGIT
0024 C31A00              JMP      MD1
0027 77         MD2:     MOV      M,A
0028 21A500              LXI      H,DNUM-1
002B CDA000              CALL     DPRT
002E CDA000              CALL     DPRT
0031 CDA000              CALL     DPRT
0034 CDA000              CALL     DPRT
0037 0E20                MVI      C,'
0039 CD0938              CALL     CO
                ;
                ;   FIRST COLUMN, CHECK FOR A LABEL
                ;
003C F1         FFCHK:   POP      PSW
003D FE3B                CPI      ';'      ;COMMENT?
003F C24E00              JNZ      LBCHK
0042 0E2A                MVI      C,'*'
0044 CD0938     FC1:     CALL     CO       ;PROCESS A COMMENT
0047 CD8100              CALL     CRCHK
004A 4F                  MOV      C,A
004B C34400              JMP      FC1
                ;
                ;
                ;   CHECK FOR A LABEL\
                ;
004E FE20       LBCHK:   CPI      ' '
0050 CA6100              JZ       POC      ;NO LABEL
0053 4F         LBC1:    MOV      C,A
0054 CD0938              CALL     CO
0057 CD8100              CALL     CRCHK
005A FE3A                CPI      ':'      ;DELETE ':'
005C C25300              JNZ      LBC1     ;LOOP TO PRINT
005F 3E20                MVI      A,' '    ;' ' SEPARTES LABEL AND OP-CODE
                ;
                ;
                ;   DO THE OPCODE, OPPERAND, AND COMMENT
                ;   MULTIPLE BLANKS BECOME SINGLE BLANKS
                ;
0061 4F         POC:     MOV      C,A
0062 CD0938              CALL     CO
0065 CD8100     POC1:    CALL     CRCHK
0068 FE20                CPI      ' '
006A CA6500              JZ       POC1
006D FE3B                CPI      ';'
006F CA4400              JZ       FC1
0072 4F         POC2:    MOV      C,A
0073 CD0938              CALL     CO
0076 CD8100              CALL     CRCHK
0079 FE20                CPI      ' '
007B C27200              JNZ      POC2
007E C36100              JMP      POC
                ;
                ;
                ;   READ A CHARACTER, MASK OFF PARITY.
                ;   IF ITS A CARRIAGE RETURN, THEN
                ;   DO THE END OF LINE THING
                ;   CONVERT CONTROL-I'S TO BLANKS.
                ;   REPRODUCE LEADER.
                ;
0081 CD0638     CRCHK:   CALL     RI       ;GET THE CHARACTER
0084 E67F                ANI      7FH      ;MASK PARITY
0086 FE0D                CPI      CR
0088 CA9500              JZ       CRC1     ;ITS THE END
008B B7                  ORA      A
008C CA9900              JZ       CRC2     ;REPRODUCE LEADER!!
008F FE09                CPI      09H      ;CONTROL-I IS A TAB
0091 C0                  RNZ               ;NOT CONTROL-I
0092 3E20                MVI      A,' '
0094 C9                  RET               ;REPLACE WITH ' '
0095 E1         CRC1:    POP      H        ;FORGET RETURN
0096 C3AA00              JMP      NLINE    ;GO TO END OF LINE
0099 4F         CRC2:    MOV      C,A
009A CD0938              CALL     CO       ;OUTPUT LEADER
009D C38100              JMP      CRCHK
                ;
                ;
                ;   PRINT OUT ((H,L)) AS AN
                ;   ASCII DECIMAL DIGIT.
                ;
00A0 23         DPRT:    INX      H
00A1 4E                  MOV      C,M
00A2 CD0938              CALL     CO
00A5 C9                  RET
                ;
                ;
00A6 30303030   DNUM:    DB       '0000'
                ;
                ;
                ;   TERMINATE A LINE WITH A
                ;   CARRIAGE-RETURN, LINE-FEED
                ;   AND GO PRINT THE NEXT LINE NUMBER.
                ;
00AA 0E0D       NLINE:   MVI      C,CR
00AC CD0938              CALL     CO
00AF 0E0A                MVI      C,LF
00B1 CD0938              CALL     CO
00B4 CD8100     NL2:     CALL     CRCHK
00B7 FE0A                CPI      LF
00B9 CAB400              JZ       NL2
00BC C31600              JMP      MDEC
                ;
                ;
0000                     END

P=
```

LISTING PRODUCED COURTESY OF COMMUNITY COMPUTER
CENTER, 1919 Menalto, MENLO PARK. (415) 326-4444

THEY HAVE COMPUTER GAMES FOR KIDS, BIRTHDAY PARTIES,
TIME ON A PDP 11 AND PSP

TIME ON A PDP/11 AND PDP/8, AND WILL REPRODUCE PAPER
TAPES.....

AN EXERCISE FOR NOVICE TRANSLATOR IMPLEMENTORS

An Arithmetic Expression Evaluator, Coded in BASIC

by Bill Thompson

Greetings: *April, 26, 1976*

I have been studying compilers, interpreters and the like, and thought that some of the methods that I have used to gain a proper acquiantance with such a complicated subject might aid other uninitiated persons.

As such, having access to an HP9830 (programmable calculator–programs in BASIC) I have constructed in BASIC, *an expression evaluator–sort of an interpreter. Since it is in BASIC, instead of assembly, the flow is a bit more obvious.*

Thanks and take care,

Bill Thompson

614 - 35 St
Evans CO 80620

Following is a program and sample run of a simple expression evaluator, written in BASIC. The program uses a transition table to "crunch" an expression. I have restrained myself from numerous embellishments which have occurred to me as I worked on the program—had I started on that route I would soon have succeeded in writing a BASIC interpreter in BASIC! Nevertheless, I do suggest that the beginner who wishes to learn enough to write a compiler or an interpreter will find it particularly helpful to write this routine in assembler code. If you have access to a version of BASIC with strings, add some of those embellishments I left out, such as program storage, exponential functions, and assigning an expression to a variable. All of these will get you into the program, and hopefully into your own language.

A TRANSITION TABLE EVALUATOR FOR ARITHMETIC EXPRESSIONS (IN BASIC)

This program illustrated the use of stack techniques and a transition table to evaluate an arithmetic expression. There are 2 stacks: a transition stack, T, and an execution stack, E (arrays "T" and "E"). The program reads the expression once from left to right and takes various actions as directed by reference to the Transition Table (array "D"). As the expression is read, if the new symbol is an identifier (name of a variable, its value is pushed on stack E. If the new symbol is an operator: b̸ (+ – * /) then the program goes to the transition table for instructions. It does this by comparing the current symbol with the top one on the translator stack (T).

Current Symbol

top of stack T	b̸ (blank)	(	+	–	*	/	)
b̸	7	1	1	1	1	1	6
(	5	1	1	1	1	1	3
+	4	1	2	2	1	1	4
–	4	1	2	2	1	1	4
*	4	1	4	4	1	1	4
/	4	1	4	4	1	1	4

Transition Table

INSTRUCTIONS:

1. Push the current operator on translator stack T, and continue reading.
2. Perform an operation, push the current symbol on T, continue.
3. Pop Stack T, continue (deletes parenthesis).
4. Perform an operation. Pop T, then repeat the table look-up with the current symbol and the new top of T.
5. Error: missing right parenthesis.
6. Error: missing left parenthesis.
7. End—evaluation complete.

Notes:

A "stack" is a last-in-first-out data vector.

All operations are performed on the top two members of the expression Stack, E.

All operations performed use the top of the T stack.

All expressions must be followed by a blank.

A blank is denoted in the table by "b̸".

Values are assigned by expressions of the form:
'LET E=5'.

Reference: *Translation of Computer Languages* by Weingarten, 1973, Holden-Day, Inc., ISBN 0-8162-9423-2. (Warning: the reference though good, contains errors in diagrams, etc.)

```
10 DIM A$[80],B$[10],C$[26],C[26],T[80],E[80],D[6,7]
20 REM
30 REM    SET UP THE TRANSITION TABLE
40 REM
50 FOR I=1 TO 6
60 FOR J=1 TO 7
70 READ D[I,J]
80 NEXT J
90 NEXT I
100 DATA 7,1,1,1,1,1,6
110 DATA 5,1,1,1,1,1,3
120 DATA 4,1,2,2,1,1,4
130 DATA 4,1,2,2,1,1,4
140 DATA 4,1,4,4,1,1,4
150 DATA 4,1,4,4,1,1,4
160 FOR I=1 TO 26
170 C[I]=0
180 NEXT I
190 FOR I=1 TO 80
200 T[I]=1
210 E[I]=0
220 NEXT I
230 B$=" (+-*/)"
240 C$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
250 DISP "INPUT EXPRESSION";
260 INPUT A$
270 IF A$[1,3]#"LET" THEN 300
280 GOSUB 380
290 GOTO 250
300 K=1
310 L=POS(B$,A$[K,K])
320 IF L#0 THEN 350
330 GOSUB 430
340 GOTO 360
```

```
350 GOSUB 530
360 K=K+1
370 GOTO 310
380 M=POS(C$,A$[5,5])
390 N=POS(A$,"=")
400 C[M]=VAL(A$[N+1])
410 PRINT "*  ";C$[M,M];"=";C[M];"  *"
420 RETURN
430 M=POS(C$,A$[K,K])
440 IF M=0 THEN 500
450 FOR I=80 TO 2 STEP -1
460 E[I]=E[I-1]
470 NEXT I
480 E[1]=C[M]
490 RETURN
500 GOSUB 1320
510 PRINT "INVALID SYMBOL"
520 GOTO 190
530 GOTO D[T[1],L] OF 540,590,650,710,770,830,890
540 REM
550 REM    INSTRUCTION I
560 REM
570 GOSUB 990
580 RETURN
590 REM
600 REM    INSTRUCTION II
610 REM
620 GOSUB 1050
630 GOSUB 990
640 RETURN
650 REM
660 REM    INSTRUCTION III
670 REM
680 GOSUB 1220
690 RETURN
700 REM
710 REM     INSTRUCTION IV
720 REM
730 GOSUB 1050
740 GOSUB 1220
750 GOSUB 530
760 RETURN
770 REM
780 REM    INSTRUCTION V
790 REM
800 GOSUB 1320
810 PRINT "MISSING RIGHT PARENTHESIS"
820 GOTO 190
830 REM
840 REM    INSTRUCTION VI
850 REM
860 GOSUB 1320
870 PRINT "MISSING LEFT PARENTHESIS"
880 GOTO 190
890 REM
900 REM    INSTRUCTION VII
910 REM
920 PRINT A$;" ="
930 PRINT "*    ";E[1];"    *"
940 PRINT
950 PRINT "*   STOP   *"
960 E[1]=0
970 GOTO 250
980 END
990 REM THIS ROUTINE ADDS A SYMBOL TO STACK T
1000 FOR I=80 TO 2 STEP -1
1010 T[I]=T[I-1]
1020 NEXT I
1030 T[1]=L
1040 RETURN
1050 REM THIS ROUTINE GENERATES AN OPERATION
1060 GOTO T[1] OF 1070,1070,1100,1130,1160,1190,1070
1070 GOSUB 1320
1080 PRINT "ERROR IN OPERATION GENERATOR"
1090 GOTO 190
1100 E[1]=E[2]+E[1]
1110 GOSUB 1270
1120 RETURN
1130 E[1]=E[2]-E[1]
1140 GOSUB 1270
1150 RETURN
1160 E[1]=E[2]*E[1]
1170 GOSUB 1270
1180 RETURN
1190 E[1]=E[2]/E[1]
1200 GOSUB 1270
1210 RETURN
1220 REM  THIS ROUTINE POPS STACK T
1230 FOR I=1 TO 79
1240 T[I]=T[I+1]
1250 NEXT I
1260 RETURN
1270 REM THIS ROUTINE SHIFTS STACK E
1280 FOR I=2 TO 79
1290 E[I]=E[I+1]
1300 NEXT I
1310 RETURN
1320 PRINT A$
1330 PRINT TAB(K-1);"^"
1340 PRINT
1350 RETURN
```

```
RUN
INPUT EXPRESSION?LET A=5
*   A=  5
INPUT EXPRESSION?LET Z=6
*   Z=  6
INPUT EXPRESSION?LET D=4
*   D=  4
INPUT EXPRESSION?LET X=3.25
*   X=  3.25
INPUT EXPRESSION?LET P=3.14159
*   P=  3.14159

INPUT EXPRESSION?A*A
A*A   =
*        25        *

*    STOP    *
INPUT EXPRESSION?A/Z*P
A/Z*P
       ↑

INVALID SYMBOL
```
(missing ƀ, that is, missing terminating blank)

```
INPUT EXPRESSION?A/Z*P
A/Z*P   =
*     0.265258463        *

*   STOP   *
INPUT EXPRESSION?(A*Z)+D*X
(A*Z)+D*X  =
*      43         *

*   STOP   *
INPUT EXPRESSION?((A-Z)/(X*Z)+P
((A-Z)/(X*Z)+P
                ↑

MISSING RIGHT PARENTHESIS
INPUT EXPRESSION?((A-Z)/(X*Z)+P)
((A-Z)/(X*Z)+P)        =
*    3.090307949        *

*   STOP  *
```

A *Classy* 8080 Text Editor

by F. J. Greeb
1915 S. Cape Way, Denver CO 80227
(303) 986-6651
May 6, 1976 [received at PCC June 21st]

Hello People, *May 12, 1976*

Enclosed is a description and source listing of my test editor program, along with some comments on conversion of the program to other 8080 systems. This material is being submitted approximately simultaneously to both the Denver Amateur Computer Society Newsletter, *and to* Dr. Dobb's Journal *for publication as either (or both) organization sees fit. As far as I know, all the bugs have been removed from the program. I have been using an earlier version, which is essentially the same except for the variable storage locations and teletype routines, for several months.*

I am also including a current description of my system, which will probably be out of date by the time you receive this, since I keep changing it, and some other general comments.

Keep up the good work.
Fred J. Greeb

GENERAL COMMENTS 5-12-76

TEXT EDITOR SOURCE LISTING—The listing is not generated directly by the 8080 assembler. It is the result of playing the source tape generated by the assembler into a system which has a high speed printer. The playback timing is not perfect and some errors do occur. All known errors have been corrected, but consider this factor as a potential source of errors when implementing the program on another system.

TEXT EDITOR PROGRAM—The program has not been optimized for either memory utilization or speed. The original design goal was to use less than 4K, for compatability with the assembler. The first version used about 2K, and therefore no size reduction was attempted.

Most commands execute with no noticeable delay. A long string search or deletion of many lines will cause a few seconds' pause.

DESIRABLE HIGH LEVEL LANGUAGE FEATURES (personal preference)—Efficient utilization of memory, possibly by converting source code to opcode (binary) equivalents, rather than storing source code directly. This conversion could be accomplished at load time, or by a separate program (a compiler?).

User definable I/O handling, including multiple I/O ports. Considering the price of PROM's, I suspect most I/O routines will end up in PROM sooner or later, with everyone using different techniques and addresses.

External subroutine call capability, including variable transfer capability.

User definable integer and floating point variable capability.

Several others I can't think of off the top of my head, and will undoubtedly remember after I mail this.

SUPPLIERS—Excellent: James Electronics, Bill Godbout (2 week service on custom-programmed PROM's). Major supplier gripe: refund credit slips rather than cash refund on out-of-stock items. These have a habit of getting lost when returned for a cash refund. I don't know how the two mentioned suppliers handle this problem. Out of numerous orders for a variety of merchandise, they have never been out-of-stock on any item. Poor suppliers: why bother to mention them—most have already received an abundance of criticism.

WANT LIST—High level language. Floating point arithmetic and I/O routines. Floating point arithmetic hardware and/or schematics. Scope driver software using D/A converters. Games. Cheap paper tape reader.

DREAM LIST—Cassette tape controllers, hardware and software. High speed CRT terminal, 72 column line minimum. Discs and controllers. A 16-bit system. High speed printer and controller. And on and on and on . . .

PLANNED APPLICATIONS—Indefinite. I designed and built the system to learn more about the hardware. That purpose was accomplished: I did learn a lot. As for what I do with it, only time will tell.

THE EDITOR

The text editor program is a strong/line oriented program written in 8080 assembly language. The program is designed for use in the development of source programs to be processed by an assembler or compiler, or for general purpose ASCII file generation. 29 separate commands are recognized by the program

The editor does not require line numbers to be present in the ASCII file. It has the capability to search for and locate any string of valid ASCII characters in the file, irrespective of their location within a line. Lines can be added, deleted, replaced, modified, or printed with simple input commands. Once initialized, the program contains self protection features so that it cannot overwrite itself.

HARDWARE REQUIREMENTS—The program occupies approximately 2.5K words of memory, plus memory space for the file being edited. An additional 128 words of memory are used for the 8080 stack. Peripherals supported are a TV-Typewriter, Baudot teletype (output only), and a cassette tape. Several of the driver routines for the peripherals are contained in the system monitor ROM, and must be supplied externally for conversion to other 8080 systems.

COMMAND FORMAT AND DESCRIPTION—All commands to the editor are input as ASCII data terminated by a carriage return. The only non-printing ASCII characters recognized by the program are carriage return (C/R, octal 15), end of file (EOF, octal 1), and Tab (Control T, octal 24). The program outputs a greater-than symbol, >, as a prompt indicating that it is waiting for a command to be input.

The commands recognized may be classified into three general categories: Initialization, Edit, and Utility. All commands must be followed by a space and/or terminated by a C/R. Additional parameters associated with a command (numerical or string data) must be separated from the command by one or more spaces.

Initialization Commands—The initialization commands set the file start address and define the end of file. All initialization commands request the file starting address, which must be input from the keyboard.

The initialization commands and their results are:

NEWF Defines a new file location starting at the input address, and enters the input mode.

EDIT Edit an existing file at the input address. Outputs the first line or page of the file, as specified by the output mode.

LOAD Loads a file from tape, beginning at the starting address. Loading begins wieh a C/R is input following the address input to allow time for manual tape setup.

Edit Commands—The edit commands are used to display and/or edit lines within the file. All edit commands operate on, or with respect to, the current line. In most cases, the current line is defined as the last line displayed on the TVT screen. The program utilizes a line pointer which always contains the starting address of the current line. This address changes as different lines within the file are accessed.

In the following descriptions, a string is defined as any sequence, of any length, of valid ASCII characters. Parameters contained within parentheses are optional parameters which may be included in the command line. Only the parameters, and not the parentheses, are included if the optional parameters are used.

A String — Append the string to the end of the current line and display the result.

BOTM — Set the current line pointer to the end of file.

C %string1%string2 — Find the first occurence of string1 in the current line and change it to string2. The two string lengths need not be equal, and the second string can be null (i.c., a C/R following the second delimiter). The delimiters (%) may be any printing ASCII character.

D (M) — Delete the current line (or M lines beginning with the current line) from the file. The file is moved in memory so that no empty space exists in the file. M is input as a decimal number, maximum value = 255.

F string — Find and display the first line in the file which begins with the string. The search begins with the line following the current line and continues until a match is found or the EOF is reached. The found line becomes the current line.

I string — Insert the string as a new line following the current line. The file is moved up in memory to make space for the new line. If no string is included, or if only a C/R is input as a command, the editor enters the continuous input mode. In this mode, multiple lines may be entered in the file by typing in each line followed by a C/R. Exit from the continuous input mode is accomplished by inputting a null line (C/R only). When the continuous input mode is entered, the message INPUT will be displayed. Upon exiting this mode, the message EDIT will be displayed. No prompt is issued between multiple input lines, which indicates that the editor is in the input mode.

INSM M — Insert M lines from memory following the current line (M = 1 to 255). The file is moved in memory to accomodate the new lines. The location (starting address) of the new lines will be requested and must be input from the keyboard. This command is designed for merging together of two files, but may also be used to move lines within the same file if the destination is at a higher memory address than the source. If this is not the case, only one line at a time may be moved correctly within the file.

LIST — List the entire file on the output device (TVT or TTY).

L string — Locate and display the first line in the file which contains the string anywhere within the line. The search begins with the line following the current line and continues until a match is found or the EOF is reached. The located line becomes the current line.

N (M) — Move the current line pointer to the next line in the file (or move M lines) and display the new current line. M may be positive or negative (max. range = ± 255).

P (M) — Print the current line (or M lines). The last line printed becomes the new current lines.

PAGE — List one page (15 lines), beginning with the current line. The current line is unchanged.

R string — Replace the current line with the input string and display the result.

T — Set the current line pointer to the top of the file and display the first line or page of the file.

Utility Commands—The utility commands allow displaying of the various pointers used by the program; specifying parameters to the program; and outputting files to tape. All addresses output by these commands are displayed in split octal, low order address first, followed by the high order address. The utility commands interface with the TVT only, and do not putput to the TTY.

The utility commands recognized, and the functions they perform, are:

CLRS — Clear TVT screen

DISP — Displays current line pointer. This command is useful for the INSM command to determine the starting address of the lines to be inserted.

DEOF — Display end of file address.

DISM — Display current setting of maximum memory size.

SETM — Set maximum meory address. This value is preset to 7.5K for use in an 8K system, leaving .5K free for later additions to a large file. This command requests an address input.

MODE L
P — Sets the output to the line (L) or page (P) mode. In the line mode, only the current line is displayed following a command. In the page moade, 15 lines are displayed. The first line displayed is the current line.

OUTM S
T(C) — Sets the output device to the TVT (S) or teletype (T). The T parameter initializes the TTY only (set to Baudot letters mode), and the TC parameter also outputs a carriage return/line feed.

RUBO X — Sets the rubout character to X. X (initialized to ") may be any printing ASCII character. The rubout character erases the previous input character in a command line. Multiple rubouts may be used to erase (back up) multiple characters.

KILL X — Sets the kill character to X. X (initialized to ?) may be any printing ASCII character. The kill character deletes the entire input line. If the kill and rubout are set to the same character, the kill function will take precedence.

Q — Quit. Exit to monitor.

TAPE — Transmitts the entire file to cassette tape. Two subcommands are associated with this command and require responses to queries displayed on the TVT. The first TVT output is "REMOVE TABS?". An input of Y (yes) will cause tabs to be converted to spaces prior to transmission to the tape recorder. If N (no) is input, the file will be taped unmodified. The next output message is "FULL OR PARTIAL FILE?". If an F (full) is input, the file is terminated by a double end of file on the tape. If P (partial) is input, the file is terminated by a single end of file followed by an end of record (octal 3). These two tape end formats are not used directly by the editor program but are for use in an associated assembler, where they signal the assembler either that more data is required or that the end of the source code has been reached. Transmission of data to tape begins when the C/R following the F or P response is input.

ERROR MESSAGES—The program will output the error message "WHAT?" in response to unrecognizable or improperly formatted commands. In addition to this general error message, several other error messages may be displayed.

On all commands which require an address input, the address is tested against the minimum useable file address. If the input address is less than the minimum, the error message "MIN ADDR (LH) = XXX YYY" will be displayed. This prevents overwriting of the editor program by the file being edited.

If a command is entered which increases the size of the file, the new end of file location is tested against the set maximum memory value. If the maximum would be exceeded by the command, the message "MEM OVERFLOW" is displayed and execution of the command is inhibited. During the LOAD command, the maximum is not tested until after the load from tape is complete, and can overwrite data stored above the meximum limit.

During execution of the INSM command, the data to be inserted is verified to be valid ASCII data. (Note: ASCII data, as defined in this program, is the 64 character upper case subset) If a non-ASCII character, other than a control character recognized by the program, is encountered, the message "BAD DATA XXX YYY" is displayed, where XXX YYY is the address of the invalid data. Execution of the INSM command is terminated if this error is displayed.

If execution of a command, such as Print M, causes the end of file to become the current line, the message "BOTTOM" will be displayed. This message will also be displayed if a Find or Locate command fails to match the input string, indicating that the sting is not present in the portion of the file searched.

CONVERSION TO OTHER SYSTEMS—Conversion to other 8080 systems should not be exceedingly difficult. Several hardware dependent I/O routines, which are contained in ROM, are called by the program. These routines will have to be supplied by the user. The routines called, the functions they perform, and the registers which may be modified by these routines are:

CRLF	A register	Output a carriage return/line feed to the TVT
CLRS	A register	Clear TVT screen
TAPI	A, B, D, E registers	Single character input from tape. Data returned in A and D registers
TAPO	B, C registers	Single character output to tape. Data in A register output and returned unmodified
TMDL	A, B, C registers	Time delay (approximately 5 seconds) for tape output routine. Enter at TMDL+6 for 0.05xC-Register value delay
HL	All except D, E	Address input from keyboard to HL registers. Carry set for normal return. Carry clear if input error occurred.
MONT	–	System monitor
LTRS, FIGS, BEQV		ASCII/BAUDOT conversion tables. Numerical values of Baudot symbols listed in ASCII sequence. Bit 8 set for ASCII symbols which have no Baudot equivalence, with 5 LSB's containing the relative address of the double character equivalence in table BEQV.

In addition to these routines and tables, a memory area for the 8080 stack is required. The program uses a 128 word memory dedicated to this purpose. The stack depth requirement has not been determined, but 20 or 30 words should be sufficient. The two TVT I/O routines (TVTI and TVTO) may also have to be modified. These routines use hardware control of the 8080 ready line, rather than flag testing or software timing.

The most convenient location for these additional memory requirements is at the end of the present editor program. Only the value of MMIN (Minimum useable file address) would have to be changed, and this value is referenced only in the address input routine (HLIN).

SYSTEM DESCRIPTION—HARDWARE May 12, 1976

Custom design and construction. Based on 8080 microprocessor. 1.25 MHz clock. Full front panel control and display.

Memory
8K RAM, Address 000 000 to 040 000
128 word RAM, Address 347 000 to 247 2000 (normally used as stack)
512 word PROM, Address 340 000 to 342 000

Peripherals
TV Typewriter (Radio Electronics TVT-1). Multiplexed half duplex type parallel data interface with hardware control of the 8080 ready line. No "echo" required.
Cassette tape mass data storage. Suding type interface with software timing (inPROM) at approximately 660 baud.
Two channel D/A converter.
Baudot teletype with UART interface. Primarily used for hard copy output.

SYSTEM DESCRIPTION—SOFTWARE

Monitor (PROM)—Includes load from tape, dump to tape, keyboard input to memory, display memory contents, execute a program, and ASCII/BAUDOT conversion tables.

Assembler—Modified Processor Technology version. Modifications include four character (max.) symbols, octal output, multiple block data input from tape, unlimited ASCII string (data) length, object code output to tape (optional), and source listing to tape (optional).

Text Editor—General purpose ASCII data handling for source code generation and modification. Line/string oriented; no line numbers required.

Denver Tiny BASIC—Features integer arithmetic, 120 variables, single dimensioned variables, remarks, and a random number generator.

Other programs—File list; memory check; octal editor; LIFE (from *PCC*, Vol. 4, No. 2—September, 1975); hex memory dump to TTY; etc.

NEW CLUB: TRACE, IN ONTARIO

There are about 50 members currently in TRACE (Toronto Region Association of Computer Enthusiasts). It covers the greater Toronto-Hamilton-Kitchener areas of Ontario, and usually holds meetings on the first or second Friday of each month. Address: TRACE, Box 545, Streetsville, Ontario, L5M 2C1 Canada.

```
  8080 TEXT EDITOR **
000 100                     * TEXT EDITOR PROGRAM
000 100                     * WRITTEN IN 8080 ASSEMBLY LANGUAGE
000 100                     * VERSION 4. APRIL 16, 1976
000 100                     * TVT OR BAUDOT TTY OUTPUT
000 100                     * USES ROM MONITOR ROUTINES
000 100                     *
000 100                     * SET INITIAL CONDITIONS
000 100                     *
000 100   061 200 347            LXI  SP,STAK       SET STACK
000 103   076 077                MVI  A,'?'   SET KILL
000 105   062 043 011            STA  KILL    CHARACTER
000 110   076 042                MVI  A,'"'   SET RUBOUT
000 112   062 047 011            STA  RUBO    CHARACTER
000 115   257                    XRA  A       CLEAR A REGISTER
000 116   062 050 011            STA  MODE    SET TO LINE MODE
000 121   041 377 035            LXI  H,1DFFH       SET MAX
000 124   042 051 011            SHLD MMAX    MEMORY
000 127   315 220 340            CALL CLRS    CLEAR SCREEN
000 132   041 267 000       IDON LXI  H,EDMS OUTPUT EDIT
000 135   315 102 002            CALL OUTR    MESSAGE
000 140   076 076           CMRT MVI  A,'>'
000 142   315 071 003            CALL TVTO    OUTPUT PROMPT
000 145   315 045 004            CALL DTIN    INPUT COMMAND
000 150   315 145 006            CALL SCNB    SCAN OFF BLANKS
000 153                     *
000 153                     * INPUT MODE IF COMMAND = C/R
000 153                     *
000 153   332 074 003            JC   INMD    JUMP TO INPUT MODE
000 156   305               SCRH PUSH B       SAVE B&C REG
000 157   021 171 007            LXI  D,CTB4  COMMAND TABLE ADDR
000 162   052 055 011            LHLD IPNT    COMMAND ADDR IN IBUF
000 165   042 053 011            SHLD ADDS    FOR SEARCH ROUTINE
000 170   076 004                MVI  A,4     4 CHAR. COMMANDS
000 172   062 044 011            STA  NCHR    FOR SEARCH ROUTINE
000 175   006 021                MVI  B,N4CS  # OF 4 CHAR. CMMDS
000 177   315 204 006            CALL CTSH    CALL SEARCH ROUTINE
000 202                     *
000 202                     * ZERO IF MATCH
000 202                     *
000 202   312 222 000            JZ   MTCH    JUMP IF MATCH
000 205   076 001                MVI  A,1     1 CHAR CMMDS
000 207   062 044 011            STA  NCHR    FOR SEARCH ROUTINE
000 212   006 014                MVI  B,N1CS  # OF 1 CHAR CMMDS
000 214   315 204 006            CALL CTSH    SEARCH TABLE
000 217   302 020 002            JNZ  WHAT    ERROR, NO MATCH
000 222   042 067 011       MTCH SHLD TEMP    SAVE ADDRESS
000 225   072 044 011            LDA  NCHR    # OF CHAR IN CMMD
000 230   117                    MOV  C,A     TO C REG
000 231   006 000                MVI  B,0     ADD BC TO HL TO
000 233   052 055 011            LHLD IPNT    GET ADDR OF CHAR
000 236   011                    DAD  B       AFTER CMMD
000 237   042 055 011            SHLD IPNT    SET IBUF POINTER
000 242   176                    MOV  A,M     GET NEXT CHAR
000 243   376 041                CPI  ' '+1   TEST IF C/R OR BLANK
000 245   322 020 002            JNC  WHAT    ERROR IF NOT
000 250   315 145 006       MTGO CALL SCNB    SCAN OFF BLANKS
000 253   042 055 011            SHLD IPNT    SET IBUF POINTER
000 256   052 067 011            LHLD TEMP    RECALL MATCH ADDR
000 261   301                    POP  B       RESTORE B&C
000 262   021 140 000       EXCT LXI  D,CMRT SET RETURN ADDRESS
```

```
000 265   325                        PUSH D       TO STACK
000 266   351                        PCHL         EXECUTE CMMD
000 267   105 104 111 124       EDMS DW    'EDIT' EDIT MESSAGE
000 273   215                        DB   8DH     MSB SET ON LAST CHAR
000 274                         *
000 274                         * TAPE OUTPUT COMMAND ROUTINE
000 274                         *
000 274   041 075 001           TPCR LXI H,TRMS  OUTPUT TAB
000 277   315 102 002                CALL OUTR    MESSAGE
000 302   315 045 004                CALL DTIN    INPUT RESPONSE
000 305   053                        DCX  H       SET TO CHAR ADDR
000 306   176                        MOV  A,M     GET RESPONSE
000 307   376 131                    CPI  'Y'     TEST IF YES
000 311   312 322 000                JZ   $+6     JUMP IF YES
000 314   376 116                    CPI  'N'     TEST IF NO
000 316   302 020 002                JNZ  WHAT    ERROR IF NOT YES OR NO
000 321   257                        XRA  A       CLEAR A REG
000 322   062 045 011                STA  TEM1    SET TAB FLAG
000 325   041 011 010                LXI  H,EFER OUTPUT FULL OR
000 330   315 102 002                CALL OUTR    PARTIAL MESSAGE
000 333   315 045 004                CALL DTIN    INPUT RESPONSE
000 336   041 072 011                LXI  H,IBUF SET ADDRESS
000 341   176                        MOV   A,M    GET RESPONSE
000 342   376 106                    CPI  'F'     TEST IF FULL
000 344   312 355 000                JZ   STCD    JUMP IF FULL
000 347   376 120                    CPI  'P'     TEST IF PARTIAL
000 351   302 020 002                JNZ  WHAT    ERROR IF NOT F OR P
000 354   257                        XRA  A       CLEAR A
000 355   062 067 011           STCD STA  TEMP    SET F OR P MODE
000 360   052 057 011                LHLD TOPL    GET START ADDR
000 363   315 207 341                CALL TMDL    TIME DELAY TO TAPE
000 366   026 001                    MVI  D,1     SET TAB COUNTER
000 370   072 045 011           NXTP LDA  TEM1    GET TAB FLAG
000 373   267                        ORA  A       SET 8080 FLAGS
000 374   176                        MOV  A,M     GET DATA FOR OUTPUT
000 375   043                        INX  H       INCREMENT ADDRESS
000 376   312 032 001                JZ   TBOK    JUMP IF TABS STAY
001 001   376 024                    CPI  TAB     TEST IF TAB
001 003   302 032 001                JNZ  TBOK    JUMP IF NOT TAB
001 006                         *
001 006                         * CONVERT TABS TO SPACES
001 006                         *
001 006   172                        MOV  A,D     GET COLUMN COUNTER
001 007   326 006                    SUI  6       SUBTRACT 6 REPEATEDLY
001 011   322 007 001                JNC  $-5     UNTIL OVERFLOW OCCURS
001 014   137                        MOV  E,A     #(-) OF SPACES TO E REG
001 015   076 040                    MVI  A,' '   SPACE
001 017   315 151 341                CALL TAPO    TO TAPE
001 022   024                        INR  D       INCREMENT COLUMN COUNT
001 023   034                        INR  E       INCREMENT SPACE COUNT
001 024   302 017 001                JNZ  $-8     LOOP UNTIL SPACES DONE
001 027   303 370 000                JMP  NXTP    GET NEXT CHAR
001 032   315 151 341           TBOK CALL TAPO    OUTPUT TO TAPE
001 035   024                        INR  D       INCREMENT COLUMN COUNT
001 036   376 001                    CPI  1       TEST IF EOF
001 040   312 053 001                JZ   TDON    JUMP IF EOF
001 043   376 015                    CPI  13      TEST IF C/R
001 045   302 370 000                JNZ  NXTP    NEXT CHAR IF NOT C/R
001 050   303 366 000                JMP  NXTP-2 RESET COLUMN COUNT
001 053   006 001               TDON MVI  B,1     SET FOR EOF OUTPUT
001 055   072 067 011                LDA  TEMP    GET F/P FLAG
001 060   267                        ORA  A       SET 8080 FLAGS
001 061   302 066 001                JNZ  TPLC    JUMP IF FULL FILE
001 064   004                        INR  B       INCREMENT B TO
001 065   004                        INR  B       EOR
001 066   170                   TPLC MOV  A,B     EOF OR EOR TO A
```

```
001 067   315 151 341          CALL TAPO   AND TO TAPE
001 072   303 203 002          JMP  DEOF   DISPLAY END ADDRESS
001 075   122 105 115 117 TRMS DW   'REMOVE TABS?'
001 101   126 105 040 124
001 105   101 102 123 077
001 111   240                  DB   ' '+80H
001 112                   *
001 112                   * NEXT COMMAND ROUTINE
001 112                   *
001 112   001 002 000     NXTC LXI  B,2    SET B&C
001 115   332 243 001          JC   NXUP   UP 1 IF NO NUMBER INPUT
001 120   305                  PUSH B      SAVE B&C
001 121   376 055              CPI  '-'    TEST IF NEGATIVE
001 123   302 141 001          JNZ  CNVV   JUMP IF POSITIVE
001 126   301                  POP  B      RESTORE B&C
001 127   006 001              MVI  B,1    SET B FOR SIGN FLAG
001 131   305                  PUSH B      SAVE B&C
001 132   052 055 011          LHLD IPNT   GET NUMBER ADDR
001 135   043                  INX  H      INCREMENT PAST -SIGN
001 136   042 055 011          SHLD IPNT   UPDATE IBUF POINTER
001 141   052 055 011     CNVV LHLD IPNT   GET NUMBER ADDRESS
001 144   104                  MOV  B,H    SET ADDRESS IN
001 145   115                  MOV  C,L    B&C
001 146   315 325 006          CALL DBCV   CONVERT TO BINARY
001 151   045                  DCR  H      TEST IF LESS THAN 255
001 152   362 020 002          JP   WHAT   JUMP IF HL > 255
001 155   301                  POP  B      RESTORE B&C
001 156   115                  MOV  C,L    # OF LINES TO C
001 157   014                  INR  C      PLUS 1
001 160   170                  MOV  A,B    SIGN FLAG TO A
001 161   267                  ORA  A      SET 8080 FLAGS
001 162   312 243 001          JZ   NXUP   MOVE UP IF POSITIVE
001 165                   *
001 165                   * MOVE BACK TOWARDS TOP
001 165                   *
001 165   175                  MOV  A,L    # OF LINES TO A
001 166   062 067 011     NXBK STA  TEMP   SAVE IN TEMP
001 171   052 057 011          LHLD TOPL   FILE START ADDR
001 174   353                  XCHG        TO DE REGISTERS
001 175   052 061 011          LHLD PNTR   CURRENT ADDRESS
001 200   053             NATP DCX  H      BACK UP 2
001 201   053                  DCX  H      CHARACTERS
001 202   315 327 003          CALL OVTS   TEST IF AT TOP
001 205   332 217 001          JC   NAOK   JUMP IF NOT TOP
001 210   353                  XCHG        TOP ADDR TO HL
001 211   042 061 011          SHLD PNTR   SET LINE POINTER
001 214   303 263 001          JMP  NDON   JUMP TO EXIT
001 217                   *
001 217                   * BACK UP TO START OF CURRENT LINE
001 217                   *
001 217   176             NAOK MOV  A,M    GET CURRENT CHAR.
001 220   376 015              CPI  13     TEST FOR C/R
001 222   302 201 001          JNZ  NATP+1 LOOP UNTIL C/R FOUND
001 225   043                  INX  H      SET ADDR TO LINE START
001 226   042 061 011          SHLD PNTR   SET LINE POINTER
001 231   072 067 011          LDA  TEMP   # OF LINES TO A
001 234   075                  DCR  A      SUBTRACT 1
001 235   302 166 001          JNZ  NXBK   LOOP UNTIL # = ZERO
001 240   303 263 001          JMP  NDON   JUMP TO EXIT
001 243                   *
001 243                   * MOVE TOWARDS END OF FILE
001 243                   *
001 243   015             NXUP DCR  C      # OF LINES - 1
001 244   312 263 001          JZ   NDON   JUMP IF DONE
001 247   315 003 002          CALL NLST   GET NEXT LINE ADDR
001 252   332 336 005          JC   BOTM   JUMP IF AT EOF
```

```
001 255   042 061 011          SHLD PNTR   SET LINE POINTER
001 260   303 243 001          JMP  NXUP   LOOP UNTIL DONE
001 263                   *
001 263                   * EXIT TO PRINT ROUTINE
001 263                   *
001 263   072 050 011     NDON LDA  MODE   MODE FLAG TO A
001 266   052 061 011          LHLD PNTR   LOAD LINE POINTER
001 271   267                  ORA  A      SET 8080 FLAGS
001 272   302 105 006          JNZ  LIST   JUMP TO PAGE OUTPUT
001 275   315 220 340          CALL CLRS   CLEAR SCREEN
001 300   303 016 003          JMP  LNOT   JUMP TO LINE OUTPUT
001 303                   *
001 303                   * RUBOUT COMMAND ROUTINE
001 303                   * SETS NEW RUBOUT CHARACTER
001 303                   *
001 303   052 055 011     RBCM LHLD IPNT   LOAD IBUF POINTER
001 306   176                  MOV  A,M    GET NEW RUBOUT CHAR
001 307   376 041              CPI  ' '+1  TEST IF BLANK OR
001 311   332 020 002          JC   WHAT   CONTROL CHAR
001 314   062 047 011          STA  RUBO   VALID, SET RUBOUT
001 317   311                  RET         EXIT TO EDIT MONITOR
001 320                   *
001 320                   * OUTPUT MODE SET COMMAND
001 320                   *
001 320   052 055 011     MSCR LHLD IPNT   LOAD IBUF ADDRESS
001 323   176                  MOV  A,M    GET NEW MODE
001 324   376 120              CPI  'P'    TEST FOR PAGE
001 326   312 337 001          JZ   STMD   JUMP IF PAGE MODE
001 331   376 114              CPI  'L'    TEST IF LINE
001 333   302 020 002          JNZ  WHAT   ERROR IF NOT P OR N
001 336   257                  XRA  A      CLEAR A
001 337   062 050 011     STMD STA  MODE   SET MODE
001 342   311                  RET         DONE
001 343                   *
001 343                   * TOP COMMAND ROUTINE
001 343                   *
001 343   052 057 011     TOPR LHLD TOPL   LOAD TOP ADDRESS
001 346   042 061 011          SHLD PNTR   SET LINE POINTER
001 351                   *
001 351                   * JUMP TO LINE OR PAGE OUTPUT
001 351                   * AS DETERMINED BY CURRENT MODE
001 351                   *
001 351   072 050 011          LDA  MODE
001 354   267                  ORA  A
001 355   302 105 006          JNZ  LIST   PAGE MODE
001 360   315 220 340     TPLE CALL CLRS
001 363   303 016 003          JMP  LNOT   LINE MODE
001 366                   *
001 366                   * KILL COMMAND ROUTINE
001 366                   *
001 366   052 055 011     KLRT LHLD IPNT   LOAD IBUF ADDRESS
001 371   176                  MOV  A,M    GET NEW KILL CHAR
001 372   376 041              CPI  ' '+1  STORE NEW KILL
001 374   332 020 002          JC   WHAT   UNLESS SPACE OR
001 377   062 043 011          STA  KILL   CONTROL CHARACTER
002 002   311                  RET
002 003                   *
002 003                   * NLST GETS NEXT LINE ADDRESS
002 003                   *
002 003   052 061 011     NLST LHLD PNTR   GET CURRENT ADDRESS
002 006   176                  MOV  A,M    GET CURRENT CHAR
002 007   043                  INX  H      INCR. ADDR
002 010   376 015              CPI  13     TEST FOR C/R
002 012   310                  RZ          DONE IF C/R
002 013   322 006 002          JNC  NLST+3 LOOP IF NOT EOF
002 016   053                  DCX  H      BACK UP 1
```

```
002 017   311                      RET
002 020                       *
002 020                       * ERROR MESSAGE OUTPUT
002 020                       *
002 020   041 034 002         WHAT LXI  H,WTMS LOAD MESSAGE ADDR
002 023   061 200 347              LXI  SP,STAK       RESET STACK
002 026   315 102 002              CALL OUTR   OUTPUT MESSAGE
002 031   303 140 000              JMP  CMRT   EXIT TO EDIT MONITOR
002 034   040 127 110 101     WTMS DW   ' WHAT?'
002 040   124 077
002 042   215                      DB   8DH
002 043                       *
002 043                       * NEW FILE COMMAND
002 043                       *
002 043   041 075 002         NEWF LXI  H,ADMS LOAD ADDRESS MSG ADDR
002 046   315 102 002              CALL OUTR   OUTPUT MESSAGE
002 051   315 236 006              CALL HLIN   INPUT ADDRESS
002 054   042 061 011              SHLD PNTR   SET LINE POINTER
002 057   042 063 011              SHLD EFPN   SET EOF POINTER
002 062   042 057 011              SHLD TOPL   SET TOP POINTER
002 065   066 001                  MVI  M,1    PUT EOF IN FILE AREA
002 067   315 076 340              CALL CRLF   OUTPUT C/R
002 072   303 074 003              JMP  INMD   JUMP TO INPUT ROUTINE
002 075   101 104 104 122     ADMS DW   'ADDR'
002 101   240                      DB   ' '+80H
002 102                       *
002 102                       * MESSAGE OUTPUT ROUTINE
002 102                       *
002 102   176                 OUTR MOV  A,M    LOAD CHARACTER
002 103   006 002                  MVI  B,2    B = 2
002 105   267                      ORA  A      SET FLAGS
002 106   362 114 002              JP   OTCH   JUMP IF MSB ZERO
002 111   326 200                  SUI  80H    CLEAR MSB
002 113   005                      DCR  B      B = 1 NOW
002 114   315 071 003         OTCH CALL TVTO   OUTPUT CHARACTER
002 117   005                      DCR  B      B = 0 OR 1 NOW
002 120   310                      RZ          DONE IF B = 0
002 121   043                      INX  H      INCREMENT ADDRESS
002 122   303 102 002              JMP  OUTR   LOOP UNTIL DONE
002 125                       *
002 125                       * DISPLAY  POINTER
002 125                       *
002 125   052 061 011         DCPL LHLD PNTR   LOAD POINTER
002 130   353                      XCHG        TO D&E
002 131   001 156 002              LXI  B,OTMS LOAD MESSAGE ADDR
002 134   041 173 002              LXI  H,OTLC OCTAL STORAGE ADDR
002 137   173                 EEOF MOV  A,E    LOW BYTE TO A
002 140   315 064 007              CALL BINH+1 CONVERT TO OCTAL
002 143   172                      MOV  A,D    HIGH BYTE TO A
002 144   043                      INX  H      INCREMENT STORAGE ADDR
002 145   315 063 007              CALL BINH   CONVERT HIGH BYTE
002 150   140                      MOV  H,B    MESSAGE ADDR
002 151   151                      MOV  L,C    TO HL REGISTERS
002 152   315 102 002              CALL OUTR   OUTPUT RESULTS
002 155   311                      RET         EXIT
002 156   120 117 111 116     OTMS DW   'POINTER (LH) '
002 162   124 105 122 040
002 166   050 114 110 051
002 172   040
002 173                       OTLC DS   3
002 176   040                      DB   ' '
002 177                            DS   3
002 202   215                      DB   8DH
002 203                       *
002 203                       * DISPLAY EOF LOCATION
002 203                       *
```

```
002 203   052 063 011         DEOF LHLD EFPN   LOAD EOF ADDRESS
002 206   353                      XCHG        TO DE REGISTERS
002 207   001 220 002              LXI  B,EMSG LOAD MESSAGE ADDR
002 212   041 231 002              LXI  H,EOLC LOAD STORAGE ADDR
002 215   303 137 002              JMP  EEOF   JUMP TO DISPLAY
002 220   105 117 106 040     EMSG DW   'EOF (LH) '
002 224   050 114 110 051
002 230   040
002 231                       EOLC DS   3
002 234   040                      DB   ' '
002 235                            DS   3
002 240   215                      DB   8DH
002 241                       *
002 241                       * DISPLAY  MAX  MEMORY  VALUE
002 241                       *
002 241   052 051 011         DISM LHLD MMAX   LOAD MAX MEM ADDR
002 244   353                      XCHG        TO DE REGISTERS
002 245   001 256 002              LXI  B,MXSG LOAD MSSG ADDR
002 250   041 273 002              LXI  H,MXLC LOAD STORAGE ADDR
002 253   303 137 002              JMP  EEOF   JUMP TO DISPLAY
002 256   115 101 130 040     MXSG DW   'MAX MEM (LH) '
002 262   115 105 115 040
002 266   050 114 110 051
002 272   040
002 273                       MXLC DS   3
002 276   040                      DB   ' '
002 277                            DS   3
002 302   215                      DB   8DH
002 303                       *
002 303                       * TAPE INPUT COMMAND
002 303                       *
002 303   041 075 002         ITCR LXI  H,ADMS OUTPUT ADDR
002 306   315 102 002              CALL OUTR   MESSAGE
002 311   315 236 006              CALL HLIN   INPUT ADDRESS
002 314   042 061 011              SHLD PNTR   SET LINE POINTER
002 317   042 057 011              SHLD TOPL   SET TOP POINTER
002 322   315 045 004              CALL DTIN   WAIT FOR C/R
002 325   052 057 011              LHLD TOPL   LOAD TOP ADDR
002 330   315 263 340         TPIN CALL TAPI   GET DATA FROM TAPE
002 333   162                      MOV  M,D    MOVE DATA TO MEMORY
002 334   376 001                  CPI  1      TEST FOR EOF
002 336   312 345 002              JZ   TDIN   JUMP IF EOF
002 341   043                      INX  H      INCREMENT ADDRESS
002 342   303 330 002              JMP  TPIN   LOOP
002 345   042 063 011         TDIN SHLD EFPN   SET EOF POINTER
002 350   315 302 003              CALL EFFN   TEST FOR OVERFLOW BY
002 353   311                      RET         SEARCHING FOR EOF
002 354                       *
002 354                       * EDIT COMMAND ROUTINE
002 354                       *
002 354   041 075 002         EDCR LXI  H,ADMS OUTPUT ADDR
002 357   315 102 002              CALL OUTR   MESSAGE
002 362   315 236 006              CALL HLIN   INPUT ADDRESS
002 365   042 061 011              SHLD PNTR   SET LINE POINTER
002 370   042 057 011              SHLD TOPL   SET TOP POINTER
002 373   315 302 003              CALL EFFN   FIND  EOF
002 376   042 063 011              SHLD EFPN   SET EOF POINTER
003 001   052 061 011              LHLD PNTR   LOAD CURRENT ADDR
003 004   072 050 011              LDA  MODE   LOAD MODE FLAG
003 007   267                      ORA  A      SET 8080 FLAGS
003 010   302 105 006              JNZ  LIST   JUMP IF PAGE MODE
003 013   315 220 340              CALL CLRS   CLEAR SCREEN
003 016                       *
003 016                       * LINE OUTPUT ROUTINE.
003 016                       *
003 016   006 001             LNOT MVI  B,1    SET COLUMN COUNT
```

```
003 020  176                MOV  A,M     GET CHARACTER
003 021  376 002            CPI  2       TEST FOR EOF
003 023  330                RC           RETURN IF EOF
003 024  043                INX  H       INCREMENT ADDRESS
003 025  376 024            CPI  TAB     TEST FOR TAB
003 027  302 040 003        JNZ  LNO1    JUMP IF NOT TAB
003 032  315 362 006        CALL TBST    CONVERT TAB TO SPACES
003 035  303 020 003        JMP  LNOT+2 LOOP
003 040  004           LNO1 INR  B       INCREMENT COLUMN COUNT
003 041  315 052 003        CALL DTOT    OUTPUT CHARACTER
003 044  376 015            CPI  13      TEST IF C/R
003 046  310                RZ           RETURN IF C/R
003 047  303 020 003        JMP  LNOT+2 LOOP
003 052                *
003 052                * DATA OUT ROUTINE.  CHANGED BETWEEN
003 052                * TVT AND TTY OUTPUT DURING PROGRAM
003 052                * EXECUTION.  ALL REGISTERS PRESERVED
003 052                *
003 052  345           DTOT PUSH H       SAVE ALL REGISTERS
003 053  325                PUSH D
003 054  305                PUSH B
003 055  365                PUSH PSW
003 056                *
003 056                * FOLLOWING INSTRUCTION CHANGED
003 056                * BY OUTM COMMAND (TVT OR TTY OUT)
003 056                *
003 056  315 071 003   DTO1 CALL TVTO    OUTPUT
003 061  361                POP  PSW     RESTORE REGISTERS
003 062  301                POP  B
003 063  321                POP  D
003 064  341                POP  H
003 065  311                RET
003 066                *
003 066                * TVT INPUT ROUTINE
003 066                *
003 066  333 000       TVTI IN   TVT     INPUT DATA
003 070  311                RET
003 071                *
003 071                * TVT OUTPUT ROUTINE
003 071                *
003 071  323 000       TVTO OUT  TVT     OUTPUT DATA
003 073  311                RET
003 074                *
003 074                * INPUT COMMAND ROUTINE
003 074                *
003 074  061 200 347   INMD LXI  SP,STAK       RESET STACK
003 077  041 003 010        LXI  H,INMS MSSG ADDR
003 102  315 102 002        CALL OUTR    OUTPUT MESSAGE
003 105  315 045 004   INLP CALL DTIN    INPUT NEW LINE
003 110  041 072 011        LXI  H,IBUF LOAD IBUF START ADDR
003 113  176                MOV  A,M     GET FIRST CHARACTER
003 116  376 015            CPI  13      TEST FOR C/R
003 116  312 132 000        JZ   IDON    DONE IF C/R
003 121  315 003 002        CALL NLST    GET NEXT LINE ADDR
003 124  042 061 011        SHLD PNTR    SET LINE POINTER
003 127  042 065 011        SHND MVAD    SET MOVE LIMIT
003 132  315 140 003        CALL CENT    INSERT LINE
003 135  303 105 003        JMP  INLP    LOOP FOR ANOTHER INPUT
003 140                *
003 140                * LINE INSERT ROUTINE
003 140                *
003 140  052 063 011   CENT LHLD EFPN    LOAD EOF ADDR
003 143  014                INR  C       INCREMENT CHAR COUNT
003 144  006 000            MVI  B,0     B = 0
003 146  011                DAD  B       EOF ADDR + CHAR COUNT
003 147  353                XCHG         IS NEW EOF (TO DE)
```

```
003 150  052 051 011        LHLD MMAX    LOAD MAX MEM VALUE
003 153  315 327 003        CALL OVTS    TEST FOR OVERFLOW
003 156  322 173 004        JNC  MOFL    JUMP IF OVERFLOW
003 161  052 063 011   MSOK LHLD EFPN    LOAD EOF ADDR
003 164  315 221 003        CALL RMOV    MOVE FILE UP
003 167  052 063 011        LHLD EFPN    LOAD EOF ADDR
003 172  011                DAD  B       ADD CHAR. COUNT
003 173  042 063 011        SHLD EFPN    SET NEW EOF ADDR
003 176  052 055 011        LHLD IFNT    NEW LINE ADDR (IBUF)
003 201  015                DCR  C       CHAR COUNT - 1
003 202  011                DAD  B       FORM LINE END ADDR
003 203  042 065 011        SHLD MVAD    SET MOVE LIMIT
003 206  052 055 011        LHLD IFNT    LINE START ADDR
003 211  353                XCHG         TO DE REGISTERS
003 212  052 061 011        LHLD PNTR    LOAD INSERT START ADDR
003 215  315 012 004        CALL LMOV    MOVE IN NEW LINE
003 220  311                RET
003 221                *
003 221                * RMOV - RIGHT (UP) MOVE.  MOVES DATA
003 221                * FROM HL ADDRESS TO DE ADDRESS UNTIL
003 221                * HL IS DECREMENTED TO MVAD
003 221                * ADDRESS (INCLUSIVE)
003 221                *
003 221  305           RMOV PUSH B       SAVE B&C
003 222  104                MOV  B,H     SOURCE ADDR TO
003 223  115                MOV  C,L     BC REGISTERS
003 224  052 065 011        LHLD MVAD    LOAD LIMIT ADDR
003 227  012           NXRM LDAX B       GET DATA
003 230  022                STAX D       STORE AT NEW ADDR
003 231  175                MOV  A,L     TEST IF AT
003 232  271                CMP  C       LOW LIMIT
003 233  302 243 003        JNZ  RMCT    JUMP IF NOT
003 236  174                MOV  A,H     TEST IF AT
003 237  270                CMP  B       HIGH LIMIT
003 240  312 250 003        JZ   RDON    JUMP IF AT LIMIT
003 243  013           RMCT DCX  B       DECREMENT
003 244  033                DCX  D       ADDRESSES
003 245  303 227 003        JMP  NXRM    MOVE NEXT CHAR
003 250  301           RDON POP  B       RESTORE B&C
003 251  311                RET
003 252                *
003 252                * SINGLE LINE INPUT COMMAND
003 252                *
003 252  332 074 003   INSL JC   INMD    JUMP IF NO STRING
003 255  015                DCR  C       DECREMENT CHAR COUNT
003 256  015                DCR  C       TWICE
003 257  041 074 011        LXI  H,IBUF+2     LINE START ADDR
003 262  042 055 011        SHLD IFNT    SET IBUF POINTER
003 265  315 003 002        CALL NLST    GET NEXT LINE ADDR
803 270  042 065 011        SHLD MVAD    SET MOVE LIMIT
003 273  042 061 011        SHLD PNTR    SET LINE POINTER
003 276  315 140 003        CALL CENT    INSERT LINE
003 301  311                RET
003 302                *
003 302                * EFFN ROUTINE - FINDS EOF AND
003 302                * TESTS FOR MEMORY OVERFLOW
003 302                *
003 302  052 051 011   EFFN LHLD MMAX    LOAD MAX MEM VALUE
003 305  353                XCHG         TO DE REGISTERS
003 306  052 061 011        LHLD PNTR    LOAD CURRENT ADDR
003 311  315 327 003   EFF1 CALL OVTS    TEST FOR OVERFLOW
003 314  332 173 004        JC   MOFL    JUMP IF OVERFLOW
003 317  176                MOV  A,M     GET CHARACTER
003 320  376 001            CPI  1       TEST FOR EOF
003 322  310                RZ           RETURN IF EOF
003 323  043                INX  H       INCREMENT ADDRESS
```

```
003 324   303 311 003          JMP  EFF1   LOOP
003 327                   *
003 327                   * HL/DE COMPARE - CARRY SET IF
003 327                   * HL GREATER THAN DE
003 327                   *
003 327   173             OVTS MOV  A,E
003 330   225                  SUB  L      FORM E - L
003 331   172                  MOV  A,D
003 332   234                  SBB  H      FORM D-H-BORROW
003 333   311                  RET
003 334                   *
003 336                   * DELETE COMMAND ROUTINE
003 334                   *
003 334   016 002         DELE MVI  C,2    # OF LINES + 1
003 336   332 357 003          JC   DLOC   JUMP IF NO # INPUT
003 341   052 055 011          LHLD IPNT   LOAD # ADDR
003 344   104                  MOV  B,H    TO B&C REGISTERS
003 345   115                  MOV  C,L
003 346   315 325 006          CALL DBCV   CONVERT # TO BINARY
003 351   115                  MOV  C,L    RESULT TO C
003 352   014                  INR  C      PLUS 1
003 353   045                  DCR  H      TEST IF < 256
003 354   362 020 002          JP   WHAT   ERROR IF > 255
003 357   015             DLOC DCR  C      DECREMENT LINE COUNT
003 360   310                  RZ          DONE IF ZERO
003 361   052 063 011          LHLD EFPN   LOAD EOF ADDR
003 364   042 065 011          SHLD MVAD   SET MOVE LIMIT
003 367                   *
003 367                   * GET NEXT LINE START ADDRESS AS
003 367                   * MOVE START ADDRESS
003 367                   *
003 367   315 003 002          CALL NLST
003 372   332 336 005          JC   BOTM   JUMP IF AT EOF
003 375   353                  XCHG        SOURCE ADDR TO DE
003 376   052 061 011          LHLD PNTR   DESTINATION ADDR
004 001   315 012 004          CALL LMOV   MOVE FILE DOWN
004 004   042 063 011          SHLD EFPN   SET NEW EOF ADDR
004 007   303 357 003          JMP  DLOC   LOOP
004 012                   *
004 012                   * LEFT (DOWN) MOVE.  MOVES DATA
004 012                   * FROM DE ADDR TO HL ADDR UNTIL
004 012                   * DE INCREMENTED TO MVAD LIMIT
004 012                   *
004 012   305             LMOV PUSH B      SAVE B&C
004 013   102                  MOV  B,D    SOURCE ADDR TO BC
004 014   113                  MOV  C,E
004 015   353                  XCHG        SAVE HL IN DE
004 016   052 065 011          LHLD MVAD   LOAD LIMIT ADDR
004 021   353                  XCHG        ADDR'S TO PROPER REGS
004 022   012             LMLP LDAX B      GET CHARACTER
004 023   167                  MOV  M,A    STORE AT NEW ADDR
004 024   171                  MOV  A,C    TEST LOW LIMIT
004 025   273                  CMP  E
004 026   302 036 004          JNZ  LMCT   JUMP IF NOT AT LIMIT
004 031   170                  MOV  A,B    TEST HIGH LIMIT
004 032   272                  CMP  D
004 033   312 043 004          JZ   LDON   JUMP IF AT LIMIT
004 036   043             LMCT INX  H      INCREMENT ADDRESSES
004 037   003                  INX  B
004 040   303 022 004          JMP  LMLP   LOOP
004 043   301             LDON POP  B      RESTORE B&C
004 044   311                  RET
004 045                   *
004 045                   * DATA INPUT ROUTINE
004 045                   *
004 045   041 072 011     DTIN LXI  H,IBUF LOAD BUFFER ADDR
```

```
004 050   042 055 011          SHLD IPNT   SET BUFFER POINTER
004 053   072 043 011          LDA  KILL   KILL CHARACTER
004 056   137                  MOV  E,A    TO E REGISTER
004 057   072 047 011          LDA  RUBO   RUBOUT CHARACTER
004 062   127                  MOV  D,A    TO D REGISTER
004 063   001 000 001          LXI  B,100H SET B&C
004 066   315 066 003     NXCH CALL TVTI   INPUT FROM TVT
004 071   273                  CMP  E      TEST IF KILL
004 072   312 045 004          JZ   DTIN   JUMP IF KILL
004 075   272                  CMP  D      TEST IF RUBOUT
004 076   302 112 004          JNZ  STOR   JUMP IF NOT RUBOUT
004 101   015                  DCR  C      DECREMENT CHAR COUNT
004 102   372 045 004          JM   DTIN   JUMP IF NEGATIVE
004 105   005                  DCR  B      DECREMENT COLUMN COUNT
004 106   053                  DCX  H      DECREMENT ADDR
004 107   303 066 004          JMP  NXCH   GET NEXT CHARACTER
004 112   167             STOR MOV  M,A    CHARACTER TO BUFFER
004 113   376 015              CPI  13     TEST FOR C/R
004 115   310                  RZ          RETURN IF C/R
004 116   171                  MOV  A,C    CHAR COUNT TO A REG
004 117   376 110              CPI  CMAX   TEST FOR MAX INPUT
004 121   312 066 004          JZ   NXCH   LOOP IF MAX
004 124   176                  MOV  A,M    RECALL CHAR
004 125   043                  INX  H      INCREMENT ADDRESS
004 126   014                  INR  C      INCREMENT CHAR COUNT
004 127   004                  INR  B      INCREMENT COLUMN COUNT
004 130   376 024              CPI  TAB    TEST FOR TAB
004 132   302 066 004          JNZ  NXCH   NEXT CHAR IF NOT TAB
004 135   005                  DCR  B      DECREMENT COLUMN COUNT
004 136   315 362 006          CALL TBST   ECHO BACK SPACES
004 141   303 066 004          JMP  NXCH   GET NEXT CHAR
004 144                   *
004 144                   * SET MAX MEMORY COMMAND
004 144                   *
004 144   041 164 004     MMCR LXI  H,MADS OUTPUT MAX
004 147   315 102 002          CALL OUTR   MEM MESSAGE
004 152   315 236 006          CALL HLIN   INPUT ADDRESS
004 155   042 051 011          SHLD MMAX   SET MAX MEM
004 160   315 076 340          CALL CRLF   C/R OUTPUT TO TVT
004 163   311                  RET
004 164   115 101 130 040 MADS DW   'MAX ME'
004 170   115 105
004 172   315                  DB   'M'+80H
004 173                   *
004 173                   * MEMORY OVERFLOW ERROR
004 173                   *
004 173   041 212 004     MOFL LXI  H,OFMS MESSAGE ADDR
004 176   061 200 347          LXI  SP,STAK      RESET STACK
004 201   315 076 340          CALL CRLF   C/R TO TVT
004 204   315 102 002          CALL OUTR   OUTPUT MESSAGE
004 207   303 140 003          JMP  CMRT   RETURN
004 212   115 105 115 040 OFMS DW   'MEM OVERFLOW'
004 216   117 126 105 122
004 222   106 114 117 127
004 226   215                  DB   8DH
004 227                   *
004 227                   * APPEI D COMMAND
004 227                   *
004 227   171             APND MOV  A,C    CHAR COUNT TO A
004 230   326 003              SUI  3      MINUS 3
004 232   332 020 002          JC   WHAT   ERROR IF ONLY 2 INPUT
004 235   015                  DCR  C      DECREMENT CHAR COUNT
004 236   015                  DCR  C      TWICE
004 237   315 003 002          CALL NLST   NEXT LINE ADDR
004 242   332 336 005          JC   BOTM   JUMP IF AT EOF
004 245   053                  DCX  H      SET ADDR TO C/R
```

```
004 246   042 067 011         SHLD TEMP   SET TEMP FOR CHANGE
004 251   041 073 011         LXI  H,IBUF+1     STRING ADDR - 1
004 254   042 045 011         SHLD TEM1   SET TEM1 FOR CHANGE
004 257   006 000             MVI  B,0    B = 0
004 261   303 007 005         JMP  CNGO   JUMP TO CHANGE ROUTINE
004 264                  *
004 264                  * CHANGE COMMAND
004 264                  *
004 264   332 020 002    CCRT JC   WHAT   ERROR IF NO STRING
004 267   052 055 011         LHLD IPNT   DELIMITER ADDR
004 272   126                 MOV  D,M    FIRST DELIMITER
004 273   006 000             MVI  B,0    ZERO CHAR COUNTER
004 275                  *
004 275                  * COUNT CHARACTERS IN FIRST STRING
004 275                  *
004 275   043            CCR1 INX  H      CHAR ADDRESS
004 276   176                 MOV  A,M    LOAD CHARACTER
004 277   376 015             CPI  13     TEST FOR C/R
004 301   312 020 002         JZ   WHAT   ERROR IF C/R
004 304   272                 CMP  D      TEST FOR DELIMITER
004 305   312 314 004         JZ   LSTR   JUMP IF DELIMITER
004 310   004                 INR  B      INCREMENT CHAR COUNT
004 311   303 275 004         JMP  CCR1   LOOP
004 314   042 045 011    LSTR SHLD TEM1   SAVE DELIMITER ADDR
004 317   052 055 011         LHLD IPNT   FIRST DELIMITER ADDR
004 322   043                 INX  H      INCREMENT TO STRING
004 323   042 055 011         SHLD IPNT   ADDR AND SAVE
004 326   052 061 011         LHLD PNTR   LOAD CURRENT FILE ADDR
004 331   110            STSH MOV  C,B    CHAR COUNT TO C
004 332   042 067 011         SHLD TEMP   SAVE CURRENT ADDR
004 335   353                 XCHG        ADDR TO DE
004 336   052 055 011         LHLD IPNT   STRING 1 ADDR
004 341   315 163 006         CALL SEAR   SEARCH FOR STRING
004 344   312 367 004         JZ   STMT   JUMP IF FOUND
004 347   052 067 011         LHLD TEMP   FILE ADDR
004 352   176                 MOV  A,M    CHARACTER FROM LINE
004 353   376 015             CPI  13     TEST FOR C/R
004 355   312 165 005         JZ   CCDN   JUMP TO EXIT IF C/R
004 360   332 336 005         JC   BOTM   AT BOTTOM IF EOF
004 363   043                 INX  H      INCREMENT ADDR TO NEXT
004 364   303 331 004         JMP  STSH   CHAR AND CONTINUE SEARCH
004 367                  *
004 367                  * COUNT CHARACTERS IN SECOND STRING
004 367                  *
004 367   052 045 011    STMT LHLD TEM1   DELIMITER ADDR
004 372   016 000             MVI  C,0    ZERO CHAR COUNTER
004 374   043            STM1 INX  H      INCREMENT ADDR
004 375   176                 MOV  A,M    LOAD STRING 2 CHAR
004 376   376 015             CPI  13     TEST FOR C/R
005 000   312 007 005         JZ   CNGO   END OF STRING 2 IF C/R
005 003   014                 INR  C      INCREMENT CHAR COUNT
005 004   303 374 004         JMP  STM1   LOOP UNTIL C/R
005 007                  *
005 007                  * BEGIN CHANGE.  B CONTAINS # OF CHAR
005 007                  * IN STRING 1; C HAS # IN STRING 2
005 007                  *
005 007   170            CNGO MOV  A,B    B TO A
005 010   271                 CMP  C      COMPARE STRING 2 LENGTH
005 011   312 142 005         JZ   EQUL   JUMP IF EQUAL
005 014   322 100 005         JNC  LESS   JUMP IF B > C
005 017                  *
005 017                  * HERE IF FILE LENGTH INCREASES
005 017                  *
005 017   052 067 011         LHLD TEMP   LOAD MATCH ADDR
005 022   205                 ADD  L      ADD STRING 1 LENGTH
005 023   157                 MOV  L,A    TO FORM ADDRESS
```

```
005 024   174                 MOV  A,H    WHERE INSERT MODE STARTS
005 025   316 000             ACI  0      ADD CARRY
005 027   147                 MOV  H,A
005 030   171                 MOV  A,C    NUMBER TO ADD IS STRING 2
005 031   220                 SUB  B      - STRING 1 (LENGTHS)
005 032   042 065 011         SHLD MVAD   SET MOVE LIMIT
005 035   052 063 011         LHLD EFPN   LOAD EOF ADDR
005 040   305                 PUSH B      SAVE STRING LENGTHS
005 041   117                 MOV  C,A    DIFFERENCE TO C
005 042   006 000             MVI  B,0    EOF PLUS DIFFERENCE
005 044   011                 DAD  B      IS NEW EOF ADDR
005 045   353                 XCHG        NEW EOF ADDR TO DE
005 046   052 051 011         LHLD MMAX   MAX MEM VALUE
005 051   315 327 003         CALL OVTS   TEST FOR OVERFLOW
005 054   322 173 004         JNC  MOFL   JUMP IF OVERFLOW
005 057   052 063 011         LHLD EFPN   LOAD EOF ADDR
005 062   315 221 003         CALL RMOV   MOVE FILE UP
005 065   052 063 011         LHLD EFPN   LOAD EOF ADDR
005 070   011                 DAD  B      FORM NEW EOF ADDR
005 071   042 063 011         SHLD EFPN   SAVE NEW EOF ADDR
005 074   301                 POP  B      RESTORE STRING LENGTHS
005 075   303 142 005         JMP  EQUL   INSERT NEW STRING
005 100                  *
005 100                  * HERE IF FILE SIZE DECREASES
005 100                  *
005 100   052 067 011    LESS LHLD TEMP   LOAD MATCH ADDR
005 103   175                 MOV  A,L    LOW ADDR TO A
005 104   201                 ADD  C      ADD STRING 2 LENGTH
005 105   157                 MOV  L,A    TO FORM MOVE
005 106   174                 MOV  A,H    DESTINATION ADDR
005 107   316 000             ACI  0      ADD CARRY
005 111   147                 MOV  H,A
005 112   353                 XCHG        SAVE IN DE
005 113   052 063 011         LHLD EFPN   LOAD EOF ADDR
005 116   042 065 011         SHLD MVAD   SET MOVE LIMIT
005 121   052 067 011         LHLD TEMP   LOAD MATCH ADDR
005 124   175                 MOV  A,L    FORM MOVE START
005 125   200                 ADD  B      AS MATCH ADDR PLUS
005 126   157                 MOV  L,A    STRING 1 LENGTH
005 127   174                 MOV  A,H
005 130   316 000             ACI  0      ADD CARRY TO HIGH ADDR
005 132   147                 MOV  H,A
005 133   353                 XCHG        DE=SOURCE, HL=DEST
005 134   315 012 004         CALL LMOV   MOVE FILE
005 137   042 063 011         SHLD EFPN   SET NEW EOF ADDR
005 142                  *
005 142                  * HERE IF FILE SIZE UNCHANGED
005 142                  *
005 142   052 067 011    EQUL LHLD TEMP   MATCH ADDR
005 145   353                 XCHG        TO DE
005 146   052 045 011         LHLD TEM1   SECOND DELIMITER ADDR
005 151   043                 INX  H      TO STRING 2 ADDR
005 152   015            EQLP DCR  C      DECREMENT STRING 2 COUNT
005 153   372 165 005         JM   CCDN   JUMP IF NEGATIVE
005 156   176                 MOV  A,M    GET STRING 2 CHAR
005 157   022                 STAX D      PUT IN FILE
005 160   043                 INX  H      INCREMENT ADDRESSES
005 161   023                 INX  D
005 162   303 152 005         JMP  EQLP   LOOP
005 165   052 061 011    CCDN LHLD PNTR   LINE START ADDR
005 170   072 050 011         LDA  MODE   OUTPUT MODE FLAG
005 173   267                 ORA  A      SET 8080 FLAGS
005 174   312 016 003         JZ   LNOT   LINE MODE OUTPUT
005 177   303 105 006         JMP  LIST   PAGE MODE OUTPUT
005 202                  *
005 202                  * LOC1 ROUTINE  - FINDS LENGTH OF STRING
```

```
005 202                       * STARTING IN COLUMN 2 OF IBUF
005 202                       *
005 202  041 073 011          LOC1 LXI  H,IBUF+1     1ST LOCATION
005 205  006 376                   MVI  B,-2     CHAR COUNTER
005 207  176                  LNCX MOV  A,M      GET CHARACTER
005 210  376 015                   CPI  13       TEST FOR C/R
005 212  312 222 005               JZ   LOC2     JUMP IF C/R
005 215  004                       INR  B        INCREMENT CHAR COUNT
005 216  043                       INX  H        INCREMENT ADDR
005 217  303 207 005               JMP  LNCX     LOOP
005 222  170                  LOC2 MOV  A,B      COUNTER TO A
005 223  267                       ORA  A        SET FLAGS
005 224  372 020 002               JM   WHAT     ERROR IF STRING < 2
005 227  004                       INR  B        SET TO ACTUAL COUNT
005 230  311                       RET
005 231                       *
005 231                       * LOCATE COMMAND - STRING SEARCH
005 231                       * AT EACH CHARACTER POSITION.  FROM
005 231                       * NEXT LINE START TO EOF
005 231                       *
005 231  315 202 005          LOCT CALL LOC1     GET STRING LENGTH
005 234  041 074 011               LXI  H,IBUF+2     STRING ADDR
005 237  042 055 011               SHLD IPNT     SET IBUF POINTER
005 242  315 003 002               CALL NLST     NEXT LINE ADDR
005 245  332 336 005               JC   BOTM     JUMP IF AT EOF
005 250  110                  LNCH MOV  C,B      STRING LENGTH TO C
005 251  042 067 011               SHLD TEMP     SAVE FILE ADDR
005 254  353                       XCHG          ALSO IN DE
005 255  052 055 011               LHLD IPNT     STRING ADDR
005 260  315 163 006               CALL SEAR     SEARCH
005 263  312 303 005               JZ   LMTH     JUMP IF FOUND
005 266  052 067 011               LHLD TEMP     CURRENT ADDR
005 271  043                       INX  H        INCREMENT ADDR
005 272  176                       MOV  A,M      NEXT CHAR
005 273  376 001                   CPI  1        TEST IF EOF
005 275  312 336 005               JZ   BOTM     JUMP IF EOF
005 300  303 250 005               JMP  LNCH     CONTINUE SEARCH
005 303                       *
005 303                       * WHEN FOUND, BACK UP TO START OF LINE
005 303                       *
005 303  052 067 011          LMTH LHLD TEMP     MATCH ADDR
005 306  053                       DCX  H        BACK UP 1
005 307  176                       MOV  A,M      LOAD CHAR
005 310  376 015                   CPI  13       TEST FOR C/R
005 312  302 306 005               JNZ  LMTH+3   LOOP IF NOT C/R
005 315  043                       INX  H        INCREMENT TO LINE START
005 316  042 061 011               SHLD PNTR     SET LINE POINTER
005 321  072 050 011          LMT1 LDA  MODE     OUTPUT MODE FLAG
005 324  267                       ORA  A        SET FLAGS
005 325  302 105 006               JNZ  LIST     PAGE OUTPUT
005 330  315 220 340               CALL CLRS     CLEAR SCREEN
005 333  303 016 003               JMP  LNOT     LINE OUTPUT
005 336                       *
005 336                       * ERROR ROUTINE WHEN EOF REACHED
005 336                       *
005 336  061 200 347          BOTM LXI  SP,STAK      RESET STACK
005 341  052 063 011               LHLD EFPN     EOF ADDRESS
005 344  042 061 011               SHLD PNTR     SET LINE POINTER
005 347  041 360 005               LXI  H,BTMS MESSAGE ADDR
005 352  315 102 002               CALL OUTR     OUTPUT MESSAGE
005 355  303 140 000               JMP  CMRT     RETURN
005 360  102 117 124 124      BTMS DW   'BOTTOM'
005 364  117 115
005 366  215                       DB   8DH
005 367                       *
005 367                       * FIND ROUTINE - COLUMN 1 LOCATE
```

```
005 367                       *
005 367  315 202 005          FIND CALL LOC1     GET STRING LENGTH
005 372  041 074 011               LXI  H,IBUF+2     STRING ADDR
005 375  042 055 011               SHLD IPNT     SET IBUF POINTER
006 000  315 003 002          FIN1 CALL NLST     NEXT LINE ADDR
006 003  042 061 011               SHLD PNTR     SET LINE POINTER
006 006  332 336 005               JC   BOTM     JUMP IF AT EOF
006 011  110                       MOV  C,B      STRING LENGTH TO C
006 012  353                       XCHG          LINE POINTER TO DE
006 013  052 055 011               LHLD IPNT     STRING ADDR
006 016  315 163 006               CALL SEAR     SEARCH
006 021  052 061 011               LHLD PNTR     LINE POINTER
006 024  312 321 005               JZ   LMT1     TO OUTPUT IF MATCH
006 027  303 000 006               JMP  FIN1     CONTINUE SEARCH
006 032                       *
006 032                       * BOTTOM COMMAND ROUTINE
006 032                       *
006 032  052 063 011          BTMM LHLD EFPN     EOF ADDR
006 035  042 061 011               SHLD PNTR     SET LINE POINTER
006 040  311                       RET
006 041                       *
006 041                       * REPLACE COMMAND
006 041                       *
006 041  171                  RLCR MOV  A,C      LINE LENGTH TO A
006 042  326 003                   SUI  3        MINUS 3
006 044  332 020 002               JC   WHAT     ERROR
006 047  015                       DCR  C        CHAR COUNT = INPUT
006 050  015                       DCR  C        MINUS 2
006 051  041 073 011               LXI  H,IBUF+1     STRING ADDR
006 054  042 045 011               SHLD TEM1     SAVE FOR CHANGE ROUTINE
006 057  052 061 011               LHLD PNTR     CURRENT ADDR
006 062  042 067 011               SHLD TEMP     SAVE FOR CHANGE
006 065  006 000                   MVI  B,0      CHARACTER COUNTER
006 067  176                  LNGT MOV  A,M      COUNT CHAR IN CURRENT
006 070  376 015                   CPI  13       LINE UNTIL C/R FOUND
006 072  312 007 005               JZ   CNGO     JUMP TO CHANGE ROUTINE
006 075  332 336 005               JC   BOTM     JUMP IF EOF
006 100  043                       INX  H        INCREMENT ADDR
006 101  004                       INR  B        INCREMENT CHAR COUNT
006 102  303 067 006               JMP  LNGT     LOOP
006 105                       *
006 105                       * PAGE COMMAND
006 105                       *
006 105  052 061 011          LIST LHLD PNTR     CURRENT ADDR
006 110  315 220 340               CALL CLRS     CLEAR SCREEN
006 113  315 076 340               CALL CRLF     C/R OUTPUT
006 116  026 020                   MVI  D,16     LINE COUNTER
006 120  025                  NLS1 DCR  D        DECREMENT LINE COUNT
006 121  310                       RZ            DONE IF ZERO
006 122  315 016 003               CALL LNOT     LINE OUTPUT
006 125  303 120 006               JMP  NLS1     LOOP
006 130                       *
006 130                       * LIST COMMAND
006 130                       *
006 130  315 220 340          LCHR CALL CLRS     CLEAR SCREEN
006 133  052 057 011               LHLD TOPL     TOP ADDRESS
006 136  315 016 003               CALL LNOT     OUTPUT 1 LINE
006 141  322 136 006               JNC  LCHR+6 LOOP IF NOT EOF
006 144  311                       RET
006 145                       *
006 145                       * SCNB ROUTINE - SCAN OFF BLANKS IN IBUF
006 145                       * CARRY SET IF TAB OR C/R FOUND
006 145                       *
006 145  052 055 011          SCNB LHLD IPNT     IBUF POINTER
006 150  176                       MOV  A,M      GET CHARACTER
006 151  376 040                   CPI  ' '      TEST FOR BLANK
```

If you would be willing to distribute **Journal** subscription leaflets (just like this)—

- * put them on all your programmers' and engineers' desks
- * place them on a table at your next club meeting
- * post them on appropriate bulletin boards
- * include them in your next club newsletter

Then please fill out and return this form:

NAME ______________________________

MAILING ADDRESS ______________________________

CITY ______________________________

STATE ______________ ZIP ______________

Quantity of leaflets desired ______________________________

Suggestions ______________________________

____

____

Mail to: People's Computer Company, Box 310, Menlo Park CA 94025.

January, 1976 * Volume 1, Number 1

Tiny BASIC Status Letter - *Dennis Allison*
16-Bit Binary-to-Decimal Conversion Routine - *Dennis Allison*
Build Your Own BASIC [reprinted from *PCC*, **Vol. 3, No. 4]** - *Dennis Allison & others*
Build Your Own BASIC, Revived [reprinted from *PCC*, **Vol. 4, No. 1]** - *Dennis Allison & Michael Christoffer*
Design Notes for Tiny BASIC [reprinted from *PCC*, **Vol. 4, No. 2]** - *Dennis Allison, Happy Lady, & friends*
Tiny BASIC [reprinted from *PCC*, **Vol. 4, No. 3]** - *Dennis Allison, Bernard Greening, Happy Lady, & lots of friends*
Extendable Tiny BASIC - *John Rible*
Corrected Tiny BASIC IL - *Bernard Greening*
Tiny BASIC, Extended Version (TBX), Part 1: Example, Command Set, Loading Instructions, Octal Listing - *Dick Whipple & John Arnold*
Letter & Schematics: Using a calculator chip to add mathematical functions to Tiny BASIC - *Dr. Robert Suding*

February, 1976 * Volume 1, Number 2

What? *Another* **Computer Hobbyist Magazine?** - *Editor*
A Critical Look at BASIC - *Dennis Allison*
Music of a Sort - *Steve Dompier*
SCELBAL: a higher level language for 8008/8080 systems: descriptive information - *Mark Arnold & Nat Wadsworth*
Tiny BASIC, Extended (TBX), Part 2: complete implementation documentation, annotated source listing, error corrections, & notes on two relocated versions - *Dick Whipple & John Arnold*
Computers that Talk: unlimited English language voice synthesis equipment, available in kit form under $1000 -*Jim Day & the Editor*
Letters & Notes
TBX Mods for a SWTP-TVT-2 - *Adolph Stumpf*
Tiny BASIC Available for the 6800 - *Tom Pittman*
Byte Swap (classified ads)
Database Questionnaire, and Subscription Blank

March, 1976 * Volume 1, Number 3

Denver Tiny BASIC, including 1-D arrays: user & implementation details, and source code
A Breakpoint Routine for MOS Technology 6502's
Keyboard Loader for Octal Code via the TVT-2
Program Repository & Tape Duplication Facility
Programming [letters]
Division remainder & Multiplication overflow
Notes to Tiny BASIC Implementers
Parser Saves Pain & Another BASIC Bombout
Tiny BASIC Suggestions & Mark-8 Needs
APL's Appeal
6800 Tiny BASIC for $5
Signetics 2650 Tiny BASIC
Proposed Functions for Tiny BASIC
Music & Computers
Mods to Dompier's Music Program
Computer Process for Rapid Production of Musical Compositions *(big machine stuff)*
Computer Speech & Computer Sensing
It Can Talk . . . But, Can It Sing?
Touchless Sensing for Under $100
Bugs & Fixes [letters]
Altair Hardware Glitches & Fixes
Grammar Glitch in TBX
Quik Bits
SPHERE-ical Complaint
16K BASIC for the 8008 (public domain)
Seattle Computer Hobbyists United
New Jersey Computer Festival
Southern California Computer Society
Diablo Printers
Monterey Computer Phreaques
Microcomputer APL
8080 Systems for the Wealthy
Future Stuff
Where Do We Go From Here? - *Editor*
Our 'Want' List
Public Interest Communications Satellite
Miscellaneous & Nitty Gritty Stuff
Signetics 2650 Kit for Under $200: includes 1K monitor ROM & 512 bytes of RAM
Submitting Items for Publication
PCC Bookstore
Byte Swap [ads]

April, 1976 * Volume 1, Number 4

Editorial: History Repeats Itself . . . I Hope - *Jim C. Warren Jr.*
Scanning the Industry Periodicals: Information derived from the May24th issue of *Electronic News*
Feature Articles
First Word on a Floppy-Disc Operating System: Command Language & Facilities Similar to DECSYSTEM-10
Hardware & Software for Speech Synthesis: Detailed discussion of Techniques & Hardware/Software Trade-offs - *Lloyd Rice*
Systems Software
MINOL—Tiny BASIC with Strings in 1.75K Bytes: An outstanding implementation by a high school junior; user & implementation documentation, & complete, annotated source code - *Erik T. Mueller*
System Monitor for 8080-Based Microcomputers: Keyboard control over program loading, examination, modification & execution; user & implementation details, & annotated program listing - *Charlie Pack*
Data
Subscription & information form
Submitting items for publication
TV Dazzler Contest
Reprint privileges
PCC Bookstore titles

May, 1976 * Volume 1, Number 5

Copyright Mania - *Jim C. Warren, Jr.*
1976 Trenton Computer Festival: a photo essay
Tidbits
40 Chars/Line Printer for $250
Day Recommends *Science* Articles
Votrax Makes the Offer: Speech synthesis kit for under $1K
Bad Bit Getters: Memory test program - *Ray Boaz*
UNIZAP: A modification of the Shooting Stars game: Programmed in 8080 Assembler Code - *John Shepard*
Palo Alto Tiny BASIC: User documentation & complete, annotated source code - *Li-Chen Wang*
A Note to Members of SCCS Concerning Lou Fields
Index to *The Computer Hobbyist*
Texas Tiny BASIC (TBX) Marries TV-Cassette Operating System (TVCOS)
Notes
The Alpha-Numeric Music System - *M. Wright*
Program Repository & Tape Duplication Facility
10% off IMSAI Products for *Journal* Readers
BASIC Just Won't Cut it [letter]
How 'Bout Small PASCAL [editor's reply]
Subscription & Information Form
What's a BAMUG?
JIPDEC Visits PCC
Chicago Stores
Byte Swap [want ads]
PCC Bookstore Offerings

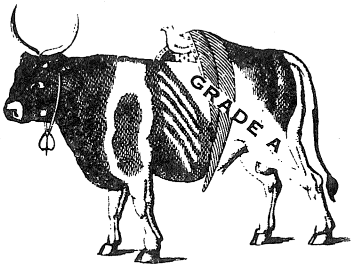
June/July, 1976 * Volume 1, Number 6

Consumer Comments [9 articles, letters & replies]
Software
A Bootstrap for the 8080 - *Lichen Wang*
Byte-Saving Programming Tricks for the 8080 - *Tom Pittman*
An Exercise for Novice Translator Implementors - *Bill Thompson*
A Classy 8080 Text Editor, similar to a PDP-9 Editor: Includes general comments, user documentation & extensively annotated source code - *F.J. Greeb*
Tiny Trek for Mueller's MINOL - *Erik Mueller*
Button, Button Game in 8080 Machine Code - *Ron Santore*
Corrections & Improvements
Errors in & Improvements for Texas Tiny BASIC (TBX) -*Charles Skeldon*
Errata & Additions to Wang's Palo Alto Tiny BASIC - *Lichen Wang*
MinErrata for MINOL, plus Tiny Trek - *Erik Mueller*
Video Output
48 Lines of 64 Characters on a TV for $499.95 - *Video Terminal Tech.*
512-Character Video RAM from Canada - *Matrox Electronic Systems*
Variable Character Spacing in Video Displays - *Jim Day*
TVT-II Mods to Get 64 Characters per Line - *David Valliere*
Homebrew TV Display System with Graphics: Complete design & implementation details, including complete schematics - *Glendon Smith*
$98.50 Graphics Terminal Kit - *Southwest Texas Products Corporation*
Miscellaneous Hardware [6 short articles]
Random Data Points [17 short articles & notes]

A reference Journal for home computer users

from the People's Computer Company—

DR. DOBB'S JOURNAL of COMPUTER CALISTHENICS & ORTHODONTIA



— 8½ x 11 inch magazine format

— "all meat" content; no display ads

— published monthly, except July & December

Content regularly includes:

Complete documentation on systems software

- Tiny BASIC, interpreters, debuggers, assemblers, compilers, cassette & floppy disc file systems, TV Dazzler software, graphics programs, music programs, etc.
- User documentation, implementation details, complete annotated source code listings

Design notes for build-your-own software

Detailed 'blue skying' about practical systems projects for the immediate future

- Tiny BASIC was the first such project (proposed, March, 1975; detailed, September, 1975; 5 systems up & running, March, '76)
- English language voice synthesis kits
- Computer music & graphics systems
- Shared mass storage
- & much, much more
- Electronic telephone book
- Community memory
- Biofeedback

Reprints of articles & schematics from computer club newsletters (*all* of 'em)

Directories: used equipment sources, users & their equipment, clubs & organizations, etc.

Indices: *All* articles in *all* major hobbyist publications, & selected articles from other publications

Active consumer advocacy for home computer users

- Supported by magazine sales—not by ads
- No vested interest in good will of manufacturers

Published ten times per year, monthly except in July and December.(Volume 1, Number 1 is January, 1976.)

□ $1.50 for a single copy: Vol.___ No.___

□ $10. per year (10 issues/year), to begin with Vol.___ No.___ □ This is a renewal.

for foreign subscriptions

□ *add* $4. per year for surface mail

□ *add* $12. per year for air mail

Payment must accompany order. We do not invoice for individual subscriptions or single copies.
Please make your check or money order payable to People's Computer Company. Thank you.

Name ____

Mailing Address ____

____

City ______________________ State ________ Zip Code ________

This information may be published in directories and lists of individuals interested in computers in non-commercial environments: □ *YES* □ *NO*

Please return this form to: PCC, Box 310, Menlo Park CA 94025; (415) 323-3111

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.__________

____
____
____
____
____
____
____

```
006 153   300                     RNZ          DONE IF NOT BLANK
006 154   043                     INX  H
006 155   042 055 011             SHLD IPNT    UPDATE POINTER
006 160   303 150 006             JMP  SCNB+3  LOOP
006 163                      *
006 163                      * SEAR ROUTINE - STRING SEARCH
006 163                      * AT ADDRESS HL & DE.  LENGTH IN C
006 163                      * ZERO FLAG SET IF MATCH
006 163                      *
006 163   032                SEAR LDAX D       GET CHAR
006 164   276                     CMP  M       TEST FOR MATCH
006 165   300                     RNZ          RETURN IF NO MATCH
006 166   043                     INX  H       INCREMENT ADDRESS
006 167   023                     INX  D
006 170   015                     DCR  C       DECREMENT CHAR COUNT
006 171   302 163 006             JNZ  SEAR    LOOP IF NOT ZERO
006 174   311                     RET
006 175                      *
006 175                      * INAD - INCREMENTS ADDRESS FOR CTSH
006 175                      *
006 175   023                INAD INX  D       INCREMENT ADDRESS
006 176   015                     DCR  C       DECREMENT CHAR COUNT
006 177   302 175 006             JNZ  INAD    LOOP UNTIL ZERO
006 202   014                     INR  C       CLEAR ZERO FLAG
006 203   311                     RET
006 204                      *
006 204                      * CTSH ROUTINE - COMMAND TABLE SEARCH
006 204                      * TABLE ADDRESS IN DE.  NUMBER OF
006 204                      * COMMANDS IN B
006 204                      *
006 204   052 053 011        CTSH LHLD ADDS    LOAD COMMAND ADDR
006 207   072 044 011             LDA  NCHR    COMMAND LENGTH
006 212   117                     MOV  C,A     TO C REGISTER
006 213   315 163 006             CALL SEAR    SEARCH
006 216   304 175 006             CNZ  INAD    INCREMENT IF NO MATCH
006 221   032                     LDAX D       LOW ADDRESS
006 222   157                     MOV  L,A     TO L REG
006 223   023                     INX  D       INCREMENT ADDR
006 224   032                     LDAX D       HIGH ADDRESS
006 225   147                     MOV  H,A     TO H REG
006 226   310                     RZ           RETURN IF MATCH
006 227   023                     INX  D       INCREMENT ADDR
006 230   005                     DCR  B       DECREMENT CMMD COUNT
006 231   302 204 006             JNZ  CTSH    LOOP IF NOT ZERO
006 234   004                     INR  B       CLEAR ZERO FLAG
006 235   311                     RET
006 236                      *
006 236                      * HLIN - ADDRESS INPUT WITH MIN TEST
006 236                      *
006 236   315 103 340        HLIN CALL HL      INPUT ADDRESS
006 241   322 020 002             JNC  WHAT    ERROR IF CARRY CLEAR
006 244   353                     XCHG         ADDR TO DE
006 245   041 202 011             LXI  H,MMIN MIN ADDR
006 250   315 327 003             CALL OVTS    COMPARE
006 253   353                     XCHG         INPUT ADDR TO HL
006 254   320                     RNC          DONE IF MIN OK
006 255   001 273 006             LXI  B,UFMS MESSAGE ADDR
006 260   341                     POP  H       POP WHATEVER'S ON STACK
006 261   041 140 000             LXI  H,CMRT RETURN ADDR
006 264   345                     PUSH H       TO STACK
006 265   041 315 006             LXI  H,UFLL STORAGE AREA
006 270   303 137 002             JMP  EEOF    DISPLAY MIN ADDR
006 273   015                UDMS DB   0FH
006 274   115 111 116 040         DW   'MIN ADDRESS (LH)
006 300   101 104 104 122
006 304   105 123 123 040
```

```
006 310   050 114 110 051
006 314   040
006 315                      UFLL DS   3
006 320   040                     DB   '
006 321                           DS   3
006 324   215                     DB   8DH
006 325                      *
006 325                      * DBCV - CONVERT ASCII NUMBER TO BINARY
006 325                      *
006 325   041 000 000        DBCV LXI  H,0     ZERO PARTIAL RESULT
006 330   012                DBC1 LDAX B       GET NUMBER
006 331   376 015                 CPI  13      TEST FOR C/R
006 333   310                     RZ           DONE IF C/R
006 334   124                     MOV  D,H     PARTIAL RESULT TO DE
006 335   135                     MOV  E,L
006 336   051                     DAD  H       TIMES 2
006 337   051                     DAD  H       TIMES 2
006 340   031                     DAD  D       ADD ORIGINAL PART RES
006 341   051                     DAD  H       TIMES 2, TOTAL=TIMES 10
006 342   326 060                 SUI  48      SUBTRACT ASCII BIAS
006 344   376 012                 CPI  10      TEST FOR VALID #
006 346   077                     CMC
006 347   332 020 002             JC   WHAT    ERROR IF NOT 0 TO 9
006 352   137                     MOV  E,A     NUMBER TO E
006 353   026 000                 MVI  D,0     ZERO D
006 355   031                     DAD  D       ADD TO RESULT
006 356   003                     INX  B       INCREMENT ADDRESS
006 357   303 330 006             JMP  DCB1    LOOP
006 362                      *
006 362                      * TAB ROUTINE - CONVERTS TABS TO SPACES
006 362                      * TAB TO COLUMNS 6, 12, 18, ETC
006 362                      *
006 362   170                TBST MOV  A,B     COLUMN COUNT TO A
006 363   326 006                 SUI  6       SUBTRACT 6 REPEATEDLY
006 365   322 363 006             JNC  $-5     UNTIL OVERFLOW OCCURS
006 370   325                     PUSH D       SAVE D&E
006 371   127                     MOV  D,A     SPACE COUNT(-) TO D
006 372   076 040                 MVI  A,' '   SPACE TO A
006 374   315 052 003             CALL DTOT    OUTPUT SPACE
006 377   004                     INR  B       INCREMENT COLUMN COUNT
007 000   024                     INR  D       INCREMENT SPACE COUNT
007 001   302 374 006             JNZ  $-8     LOOP UNTIL ZERO
007 004   321                     POP  D       RESTORE D&E
007 005   311                     RET
007 006                      *
007 006                      * OUTM COMMAND - OUTPUT DESTINATION SET
007 006                      *
007 006   052 055 011        OUTM LHLD IPNT    IBUF POINTER ADDR
007 011   176                     MOV  A,M     DESTINATION CHAR
007 012   376 123                 CPI  'S'     TEST FOR TVT (SCREEN)
007 014   312 054 007             JZ   TVST    JUMP IF TVT
007 017   376 124                 CPI  'T'     TEST FOR TELETYPE
007 021   302 020 002             JNZ  WHAT    ERROR IF NOT S OR T
007 024   353                     XCHG         SAVE IBUF ADDR
007 025   041 237 010             LXI  H,ABOT TTY ROUTINE ADDR
007 030   042 057 003             SHLD DTO1+1 SET IN OUTPUT ROUTINE
007 033   076 137                 MVI  A,LTCD BAUDOT LETTERS CODE
007 035   062 071 011             STA  MDTY    SET IN TTY MODE FLAG
007 040   315 224 010             CALL SCOT    OUTPUT TO TTY
007 043   353                     XCHG         IBUF ADDR TO HL
007 044   043                     INX  H       INCREMENT ADDR
007 045   176                     MOV  A,M     GET NEXT CHAR
007 046   376 103                 CPI  'C'     TEST IF C
007 050   300                     RNZ          DONE IF NOT C
007 051   303 000 011             JMP  TYCR    C/R L/F OUTPUT TO TTY
007 054   041 071 003        TVST LXI  H,TVTO TVT OUTPUT ROUTINE ADDR
```

```
007 057   042 057 003          SHLD DTO1+1 SET IN OUTPUT ROUTINE
007 062   311                  RET
007 063                   *
007 063                   * BINH - CONVERT BINARY TO OCTAL
007 063                   *
007 063   043             BINH INX  H       ENTRY FOR SPACE FIRST
007 064   053                  DCX  H       NORMAL ENTRY
007 065   305                  PUSH B       SAVE B&C
007 066   016 003              MVI  C,3     LOOP COUNTER
007 070   007                  RLC          SHIFT LEFT
007 071   007                  RLC          TWO PLACES
007 072   107                  MOV  B,A     SAVE VALUE
007 073   241                  ANA  C       2 MSB'S IN A NOW
007 074   043             STBN INX  H       INCREMENT STORAGE ADDR
007 075   306 060              ADI  48      ADD ASCII BIAS
007 077   167                  MOV  M,A     STORE RESULT
007 100   015                  DCR  C       DECREMENT LOOP COUNT
007 101   312 116 007          JZ   BHDN    DONE IF ZERO
007 104   170                  MOV  A,B     RECALL VALUE TO A
007 105   007                  RLC          SHIFT LEFT
007 106   007                  RLC          THREE PLACES
007 107   007                  RLC
007 110   107                  MOV  B,A     SAVE VALUE
007 111   346 007              ANI  7       MASK THREE BITS
007 113   303 074 007          JMP  STBN    LOOP AND STORE
007 116   301             BHDN POP  B       RESTORE B&C
007 117   311                  RET
007 120                   *
007 120                   * PRINT COMMAND
007 120                   *
007 120   016 002         PRTN MVI  C,2     SET LINE COUNT
007 122   332 145 007          JC   PLP1    JUMP IF NO # INPUT
007 125   052 055 011          LHLD IPNT    # ADDR IN IBUF
007 130   104                  MOV  B,H     ADDR TO B&C
007 131   115                  MOV  C,L
007 132   315 325 006          CALL DBCV    CONVERT TO BINARY
007 135   115                  MOV  C,L     RESULT TO C
007 136   045                  DCR  H       TEST IF > 255
007 137   362 020 002          JP   WHAT    ERROR IF > 255
007 142   171                  MOV  A,C     LINE COUNT TO A
007 143   014                  INR  C       INCREMENT COUNT
007 144   267                  ORA  A       SET FLAGS
007 145   304 220 340     PLP1 CNZ  CLRS    CLEAR IF NOT ZERO
007 150   052 061 011          LHLD PNTR    CURRENT LINE ADDR
007 153   015             PLP2 DCR  C       DECREMENT LINE COUNT
007 154   310                  RZ           DONE IF ZERO
007 155   042 061 011          SHLD PNTR    SET LINE POINTER
007 160   315 016 003          CALL LNOT    OUTPUT 1 LINE
007 163   332 336 005          JC   BOTM    JUMP IF AT EOF
007 166   303 153 007          JMP  PLP2    LOOP
007 171                   *
007 171                   * COMMAND TABLE - COMMAND FOLLOWED
007 171                   * BY ROUTINE ADDRESS
007 171                   *
007 171   105 104 111 124 CTB4 DW   'EDIT'
007 175   354 002              DW   EDCR
007 177   117 125 124 115      DW   'OUTM'
007 203   006 007              DW   OUTM
007 205   103 114 122 123      DW   'CLRS'
007 211   220 340              DW   CLRS
007 213   111 116 123 115      DW   'INSM'
007 217   037 010              DW   IFMC
007 221   120 101 107 105      DW   'PAGE'
007 225   105 006              DW   LIST
007 227   116 105 127 106      DW   'NEWF'
007 233   043 002              DW   NEWF
```

```
007 235   104 111 123 120      DW   'DISP'
007 241   125 002              DW   DCPL
007 243   104 105 117 106      DW   'DEOF'
007 247   203 002              DW   DEOF
007 251   113 111 114 114      DW   'KILL'
007 255   366 001              DW   KLRT
007 257   122 125 102 117      DW   'RUBO'
007 263   303 001              DW   RBCM
007 265   124 101 120 105      DW   'TAPE'
007 271   274 000              DW   TPCR
007 273   102 117 124 115      DW   'BOTM'
007 277   032 006              DW   BTMM
007 301   114 117 101 104      DW   'LOAD'
007 305   303 002              DW   ITCR
007 307   104 111 123 115      DW   'DISM'
007 313   241 002              DW   DISM
007 315   115 117 104 105      DW   'MODE'
007 321   320 001              DW   MSCR
007 323   123 105 124 115      DW   'SETM'
007 327   144 004              DW   MMCR
007 331   114 111 123 124      DW   'LIST'
007 335   130 006              DW   LCHR
007 337                   *
007 337                   * SINGLE CHARACTER COMMANDS
007 337                   *
007 337   101                  DB   'A'
007 340   227 004              DW   APND
007 342   104                  DB   'D'
007 343   334 003              DW   DELE
007 345   111                  DB   'I'
007 346   252 003              DW   INSL
007 350   105                  DB   'E'
007 351   354 002              DW   EDCR
007 353   114                  DB   'L'
007 354   231 005              DW   LOCT
007 356   116                  DB   'N'
007 357   112 001              DW   NXTC
007 361   103                  DB   'C'
007 362   264 004              DW   CCRT
007 364   124                  DB   'T'
007 365   343 001              DW   TOPR
007 367   120                  DB   'P'
007 370   120 007              DW   PRTN
007 372   106                  DB   'F'
007 373   367 005              DW   FIND
007 375   122                  DB   'R'
007 376   041 006              DW   RLCR
010 000   121                  DB   'Q'
010 001   000 340              DW   MONT
010 003                   *
010 003                   * TWO MESSAGES
010 003                   *
010 003   111 116 120 125 INMS DW   'INPUT'
010 007   124
010 010   215                  DB   8DH
010 011   106 125 114 114 EFER DW   'FULL OR PARTIAL FILE?'
010 015   040 117 122 040
010 021   120 101 122 124
010 025   111 101 114 040
010 031   106 111 114 105
010 035   077
010 036   240                  DB   ' '+80H
010 037                   *
010 037                   * INSM COMMAND - INSERT FROM MEMORY
010 037                   *
010 037   052 055 011     IFMC LHLD IPNT    IBUF ADDR
```

```
010 042  104                    MOV  B,H     TO B&C
010 043  115                    MOV  C,L
010 044  315 325 006            CALL DBCV    # TO BINARY
010 047  045                    DCR  H       TEST VALUE
010 050  362 020 002            JP   WHAT    TOO LARGE
010 053  175                    MOV  A,L
010 054  267                    ORA  A       SET FLAGS
010 055  312 020 002            JZ   WHAT    ERROR, ZERO LINES
010 060  062 044 011            STA  NCHR    SAVE VALUE
010 063  041 075 002            LXI  H,ADMS
010 066  315 102 002            CALL OUTR    OUTPUT MESSAGE
010 071  315 236 006            CALL HLIN    INPUT ADDRESS
010 074  353                    XCHG         TO DE REG
010 075  016 002           IFM4 MVI  C,2     SET CHAR COUNTER
010 077  041 074 011            LXI  H,IBUF+2     SET ADDR
010 102  032               IFM3 LDAX D       GET NEW CHAR
010 103  267                    ORA  A       SET FLAGS
010 104  372 171 010            JM   BADT    BAD IF BIT 8 SET
010 107  376 040                CPI  ' '     TEST IF BLANK
010 111  322 126 010            JNC  IFM1    OK IF >= BLANK
010 114  376 024                CPI  TAB     TEST FOR TAB
010 116  312 126 010            JZ   IFM1    OK IF TAB
010 121  376 015                CPI  13      TEST IF C/R
010 123  302 171 010            JNZ  BADT    BAD IF NOT C/R
010 126                    *
010 126                    * NEW LINE TO IBUF
010 126                    *
010 126  167               IFM1 MOV  M,A     DATA TO IBUF
010 127  043                    INX  H       INCREMENT ADDR
010 130  023                    INX  D
010 131  376 015                CPI  13      TEST FOR C/R
010 133  312 145 010            JZ   IFM2    EOL IF C/R
010 136  014                    INR  C       INCREMENT CHAR COUNT
010 137  171                    MOV  A,C     TO A
010 140  376 110                CPI  CMAX    TEST FOR MAX LENGTH
010 142  302 102 010            JNZ  IFM3    CONTINUE IF OK
010 145  267               IFM2 ORA  A       CLEAR CARRY
010 146  325                    PUSH D       SAVE ADDRESS
010 147  315 252 003            CALL INSL    INSERT LINE
010 152  321                    POP  D
010 153  072 044 011            LDA  NCHR    LOAD LINE COUNTER
010 156  075                    DCR  A       MINUS 1
010 157  062 044 011            STA  NCHR    SAVE LINE COUNT
010 162  302 075 010            JNZ  IFM4    LOOP IF NOT ZERO
010 165  315 076 340            CALL CRLF    C/R TO TVT
010 170  311                    RET
010 171  [illegible] 202 010    BADT LXI  B,BDMS MESSAGE ADDR
010 174  041 214 010            LXI  H,BDLC STORAGE ADDR
010 177  303 137 002            JMP  EEOF    OUTPUT BAD DATA ADDR
010 202  015               BDMS DB   0DH
010 203  102 101 104 040        DW   'BAD DATA '
010 207  104 101 124 101
010 213  040
010 214                    BDLC DS   3
010 217  040                    DB   ' '
010 220                         DS   3
010 223  215                    DB   8DH
010 224                    *
010 224                    * OUTPUT TO TTY ROUTINE
010 224                    *
010 224  107               SCOT MOV  B,A     SAVE CHAR
010 225  333 005           SCO1 IN   STAT    INPUT UART STATUS
010 227  007                    RLC          TMST FLAG TO CARRY
010 230  322 225 010            JNC  SCO1    LOOP UNTIL READY
010 233  170                    MOV  A,B     RECALL DATA
010 234  323 004                OUT  TTY     OUTPUT
```

```
010 236  311                    RET
010 237                    *
010 237                    * ASCI:/BAUDOT CONVERSION ROUTINE
010 237                    *
010 237  117               ABOT MOV  C,A     SAVE CHAR
010 240  376 040                CPI  20H     TEST FOR SPACE
010 242  332 341 010            JC   CNTL    CONTROL CHAR IF < SPACE
010 245  302 255 010            JNZ  $+5     JUMP IF NOT SPACE
010 250  076 004                MVI  A,4     BAUDOT SPACE
010 252  303 335 010            JMP  OT1C    OUTPUT
010 255  346 140                ANI  60H     GET BITS 5 & 6
010 257  107                    MOV  B,A     SAVE IN B
010 260  072 071 011            LDA  MDTY    CURRENT TTY MODE
010 263  127                    MOV  D,A     SAVE
010 264  240                    ANA  B       AND WITH CHAR MODE
010 265  302 301 010            JNZ  MDD1    JUMP IF UNCHANGED
010 270  076 144                MVI  A,64H   LOAD 01100100 BINARY
010 272  252                    XRA  D       FORM NEW MODE
010 273  062 071 011            STA  MDTY    SET FLAG
010 276  315 224 010            CPLL SCOT    OUTPUT LTRS/FIGS CODE
010 301  076 037           MDD1 MVI  A,31    SET 5 LSB'S
010 303  241                    ANA  C       GET LSB'S OF CHAR
010 304  137                    MOV  E,A     SAVE
010 305  026 000                MVI  D,0
010 307  171                    MOV  A,C     RECALL CHAR
010 310  376 100                CPI  64      TEST IF LTRS OR FIGS
010 312  322 323 010            JNC  LTSH    JUMP IF LTRS
010 315  041 315 341            LXI  H,FIGS FIGS TABLE ADDR
010 320  303 326 010            JMP  $+3
010 323  041 255 341       LTSH LXI  H,LTRS TABLE ADDR
010 326  031                    DAD  D       ADD CHAR ADDRESS
010 327  176                    MOV  A,M     LOAD BAUDOT CHAR
010 330  376 200                CPI  128     TEST IF BIT 8 SET
010 332  322 017 011            JNC  BDEQ    JUMP IF SET
010 335  315 224 010       OT1C CALL SCOT    OUTPUT CHAR
010 340  311                    RET
010 341  376 012           CNTL CPI  10      LINE FEED
010 343  302 353 010            JNZ  $+5
010 346  076 002                MVI  A,BDLF BAUDOT L/F
010 350  303 335 010            JMP  OT1C    OUTPUT
010 353  376 015                CPI  13      C/R
010 355  312 000 011            JZ   TYCR
010 360  376 007                CPI  7       BELL
010 362  300                    RNZ
010 363  076 073                MVI  A,FGCD FIGS CODE
010 365  062 071 011            STA  MDTY    SET FLAG
010 370  315 224 010            CALL SCOT    OUTPUT
010 373  076 005                MVI  A,BDBL BAUDOT BELL
010 375  303 335 010            JMP  OT1C    OUTPUT
011 000  076 010           TYCR MVI  A,BDCR BAUDOT C/R
011 002  315 224 010            CALL SCOT    OUTPUT
011 005  016 006                MVI  C,6     SET C
011 007  315 215 341            CALL TMDL+6 0.3 SEC DELAY
011 012  076 002                MVI  A,BDLF BAUDOT L/F
011 014  303 335 010            JMP  OT1C    OUTPUT
011 017                    *
011 017                    * TWO CHARACTER EQUIVALENCES
011 017                    *
011 017  346 037           BDEQ ANI  1FH     GET 5 LSB'S
011 021  117                    MOV  C,A     SAVE
011 022  006 000                MVI  B,0
011 024  041 355 341            LXI  H,BEQV TABLE ADDR
011 027  011                    DAD  B       ADD OFFSET
011 030  176                    MOV  A,M     GET FIRST CHAR
011 031  345                    PUSH H       SAVE ADDR
011 032  315 237 010            CALL ABOT    OUTPUT 1ST
```

Pointers to other good stuff

WE WANTED TO INCLUDE MUCH MORE IN THIS ISSUE THAN WE COULD AFFORD. PART OF IT WILL JUST HAVE TO WAIT FOR FUTURE ISSUES. THE REST OF IT (ALONG WITH STILL OTHER USEFUL TIDBITS) HAS BEEN PUBLISHED ELSEWHERE:

Southwest Texas Products Corp., 219 W. Rhapsody, San Antonio TX 78216, has put out the first issue of their *Newsletter*, a 49-page, loose-leaf job. We would like to applaud their work and their approach to hobbyist software. This issue of the SWTPC *Newsletter* contains extensive information on 6800 software, including some "bug" notices and corrections, a list of available 6800 games, some hardware notes and schematics, and complete listings of:

A Black Jack game-playing program (9 pages, full-size, hex-coding only)
A Memory-Dump program (2 user-documentation pages, 2 pages of unannotated source code)
A 1.3K Editor (3 user-documentation pages, 6 pages of unannotated source code)
A 3.15K Micro BASIC (5 user-documentation pages, 15 pages of unannotated source code)

The Editor was written by Robert Uiterwyk, 4402 Meadowwood Way, Tampa FL 33624. Micro BASIC was done by Uiterwyk and Bill Turner. We have spoken with Mr. Uiterwyk several times (we originally planned to publish Micro BASIC in this issue), and think "his head's in the right place." He and his associates are actively pursuing the production of free and *very* inexpensive systems software for hobbyists. We would like to praise their efforts and urge them onward.

The July issue of *People's Computer Company*, Box 310, Menlo Park CA 94025, contains its usual load of exciting items, notably including:

Lichen Wang's Star Trek, written for Palo Alto Tiny BASIC [*DDJ*, Vol. 1, No. 5] (We wanted to publish it in this issue of the *Journal* but didn't have room.)

An update of the comprehensive list of computer stores in the May issue of *PCC*.

An update of the list of computer clubs that was given in the preceding issue of *PCC*.

```
011 035   341                   POP  H
011 036   043                   INX  H         NEXT ADDR
011 037   176                   MOV  A,M       SECOND CHAR
011 040   303 237 010           JMP  ABOT      OUTPUT
011 043                  *
011 043                  * VARIABLES
011 043                  *
011 043                  SP     EQU  6         STACK POINTER
011 043                  PSW    EQU  6         STATUS WORD
011 043                  TVT    EQU  0         TVT PORT
011 043                  TAPE   EQU  1         TAPE PORT
011 043                  N4CS   EQU  17        # OF 4 CHAR CMMDS
011 043                  N1CS   EQU  12        # OF 1 CHAR CMMDS
011 043                  CMAX   EQU  72        MAX INPUT LENGTH
011 043                  TAB    EQU  20        TAB CHAR
011 043                  CRLF   EQU  0E03EH    TVT C/R OUTPUT
011 043                  CLRS   EQU  0E090H    CLEAR SCREEN ROUTINE
011 043                  HL     EQU  0E043H    ADDR IN ROUTINE
011 043                  TMDL   EQU  0E187H    TIME DELAY ROUTINE
011 043                  TAPO   EQU  0E169H    TAPE OUTPUT ROUTINE
011 043                  TAPI   EQU  0E0B3H    TAPE INPUT ROUTINE
011 043                  LTRS   EQU  0E1ADH    LETTERS TABLE ADDR
011 043                  FIGS   EQU  0E1CDH    FIGURES TABLE ADDR
011 043                  BEQV   EQU  0E1EDH    DOUBLE EQUIV TABLE ADDR
011 043                  STAK   EQU  0E780H    8080 STACK
011 043                  MONT   EQU  0E000H    MONITOR ADDR
011 043                  KILL   DS   1         KILL CHAR
011 044                  NCHR   DS   1         # OF CHAR
011 045                  TEM1   DS   2         TEMPORARY STORAGE
011 047                  RUBO   DS   1         RUBOUT FLAG
011 050                  MODE   DS   1         OUTPUT MODE FLAG
011 051                  MMAX   DS   2         MAX MEMORY POINTER
011 053                  ADDS   DS   2         ADDRESS STORAGE
011 055                  IPNT   DS   2         IBUF POINTER
011 057                  TOPL   DS   2         TOP POINTER
011 061                  PNTR   DS   2         LINE POINTER
011 063                  EFPN   DS   2         EOF POINTER
011 065                  MVAD   DS   2         MOVE ADDR POINTER
011 067                  TEMP   DS   2         TEMPORARY STORAGE
011 071                  MDTY   DS   1         TTY MODE FLAG
011 072                  IBUF   DS   72        INPUT BUFFER
011 202                  BDLF   EQU  2         BAUDOT L/F
011 202                  BDCR   EQU  8         BAUDOT C/R
011 202                  BDBL   EQU  5         BAUDOT BELL
011 202                  FGCD   EQU  3BH       FIGURES CODE
011 202                  LTCD   EQU  5FH       LETTERS CODE
011 202                  TTY    EQU  4         TTY PORT
011 202                  STAT   EQU  5         STATUS PORT
011 202                  MMIN   EQU  $         MIN USABLE ADDR
011 202                  *
011 202                  ******* END OF EDITOR
```

48 LINES OF 64 CHARACTERS ON A TV

Kit Price is $499.95

by Video Terminal Technology staff
6108 Elmbridge Dr., San Jose CA 95129

I've seen this running on a small Sony teevee, and was very impressed. The characters were clear and sharp. They bypass the RF and amp, and go directly to the tube to avoid character smear and obtain higher bandwidth. Screen update was fast. The company is small, run by good people, and I believe they will be quite responsive to their customers. —Jim Warren

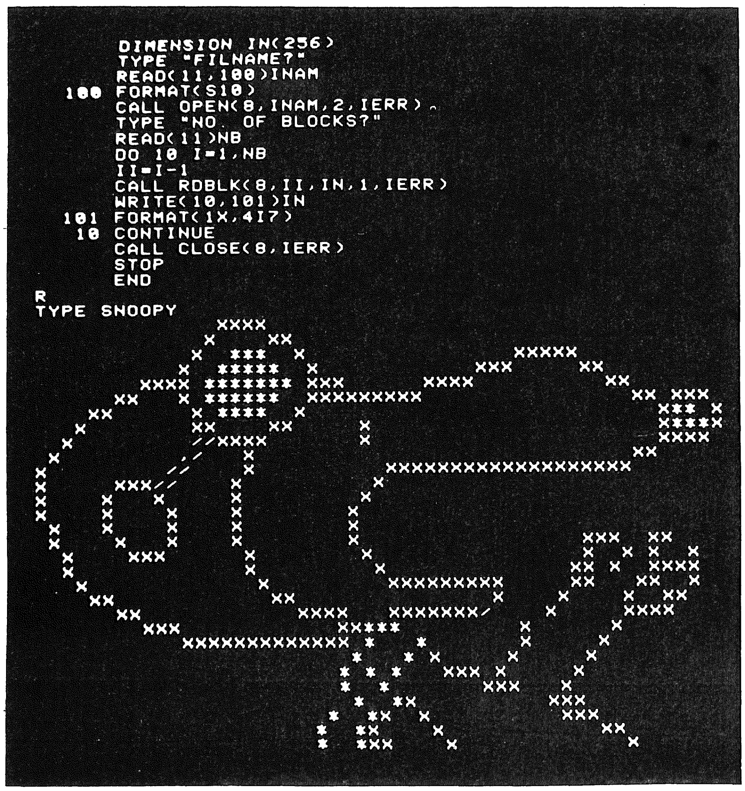
Video Terminal Technology announces a new video computer terminal with all the features of a professional terminal at a hobbyist price. The VT-4000 video terminal displays 48 lines of 64 characters in a 5x7 font. This provides the capability to display 3076 (3K) characters simultaneously—8 times the standard tv typewriter's 16 lines of 32 characters.

The VT-4000 gives the operator complete control over his or her display. The keyboard interface card decodes all 32 of the standard ASCII control functions. These control functions are user designated and can be strapped to match any software operating system. The selected controls can move the cursor up, down, right, left, and home. Direct cursor addressing uses two control characters to position the cursor anywhere on the CRT screen. Other control functions can be used to selectively clear the displayed page, clear the entire memory, or clear the character positions from the present cursor position to the end of the line. Two more control characters allow the operator to display individual characters either white on black or black on white. This leaves 16 control characters available for the requirements of the particular software operating system.

The VT-4000 video terminal also offers other standard on/off features such as power-on clear, clear to end of line with line feed, scroll up, and scroll down. The scroll up/down feature allows up to 16K of RAM to be scrolled through before any data is lost. After all of the available RAM has been scrolled through, the VT-4000 then starts to overwrite the previous data. The VT-4000 basic configuration comes with 4K of RAM, expandable to 16K.

The VT-4000 has been designed to easily interface to any computer and any video monitor or slightly modified television receiver. The computer I/O available is either RS232, TTL serial, or TTL parallel at any of the standard BAUD rates from 110 to 9600. The video monitor input available is either composit video/sync, separate video and composit sync, or separate video, separate horizontal sync, and separate vertical sync. A television receiver may be used as a video monitor if the following modification is made. Break the signal path between the IF section and the video section, and insert the composit video/sync at this point. However, if a sharper display is desired, insert the composit sync at this point and apply the video directly to the cathode of the CRT. Any questions about this modification will be answered by Video Terminal Technology (VTT).

The VT-4000 is available from VTT primarily in kit form in any configuration from single boards to 100% complete kits. Assembled and tested boards or complete models can be purchased for a standard assembly fee. All such options carry a six-month parts and labor guarantee.



```
      DIMENSION IN(256)
      TYPE "FILNAME?"
      READ(11,100)INAM
  100 FORMAT(S10)
      CALL OPEN(8,INAM,2,IERR)
      TYPE "NO. OF BLOCKS?"
      READ(11)NB
      DO 10 I=1,NB
      II=I-1
      CALL RDBLK(8,II,IN,1,IERR)
      WRITE(10,101)IN
  101 FORMAT(1X,4I7)
   10 CONTINUE
      CALL CLOSE(8,IERR)
      STOP
      END
R
TYPE SNOOPY
              XXXX
             X    XX
            X  ***  X                XXXXX
           X  *****  X            XXX     XX
        XXXX *******  XXX      XXXX          XX
      XX   X ******  XXXXXXXXX                XX XXX
    XX      X ****  X                           X** X
   X        XX    XX     X                      X***X
  X         //XXXX       X                      XXXX
 X         //   X                             XX
X         //    X          XXXXXXXXXXXXXXXXXXX
X     XXX//    X          X
X    X   X     X         X
X    X    X    X        X
 X   X    X    X        X
 X    X   X    X        X                 XXX  XX
  X    XXX      X        X                X  X X  X
  X             X         X              XX X  XXXX
   X             X         XXXXXXXXX     XX   XX  X
    XX            XX               X    X    X  XX
      XX            XXXX   XXXXXXX/    X     XXXX
        XXX            XX***         X      X
           XXXXXXXXXXXXX *   *       X     X
                        *   * X     X     X
                       * * *   XXX X     X
                       *  *       XXX   X
                        *  *X          XXX
                      *  *X  X          XXX
                     *  *X    X            XX
                     *  *XX    X             X
```

Larry Balch photo
2366 Mossdale Way, San Jose CA 95133

512 - CHARACTER VIDEO RAM

Matrox Electronic Systems [P.O. Box 56, Ahuntsic Syn., Montreal, Quebec, Canada, H3L 3N5, (514) 481-6838] has announced a most interesting widgit:

Their MTX-1632 is a single physical component. Its input pins can be directly connected to any M-P bus and appear to be input to a 512x8 RAM. The output, however, is a video signal that directly drives a TV monitor. It displays 16 lines of 32 characters each, interpreting the bytes in its RAM as ASCII character codes. It requires only a single 5-volt power supply, can drive up to 25 TV monitors, offers character-blink, and has an access time under 650 nanoseconds.

SONOMA COUNTY COMPUTERS HOLD MEETINGS

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

The SONOMA COUNTY MICRO COMPUTER CLUB in Northern California is small but powerful. We are a group of several ALTAIR's, an IMSAI, a JOLT, two PDP-8's, an APPLE, and some others on order. We all have people up and running.

We meet the first Tuesday in each month at LO*OP CENTER in Cotati. Meeting time is 7:30 p.m. Any interested systems are invited to attend with their operators.

BYE BYE BIRDIE
LO*OP CENTER CLASSIC PDP-8

LO*OP CENTER
8099 La Plaza
Cotati CA 94928

VARIABLE CHARACTER SPACING IN VIDEO DISPLAYS

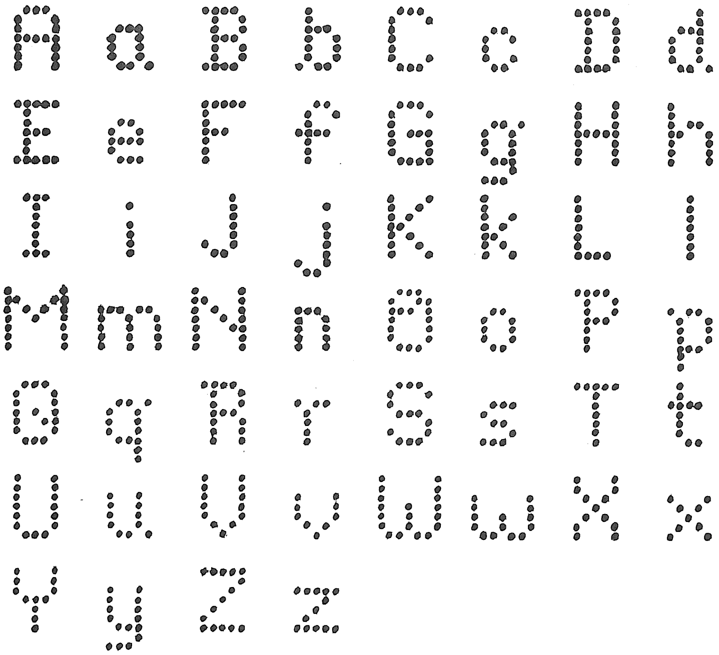
by Jim Day
17042 Gunther St.
Granada Hills CA 91344

Figure 1 shows a typical dual-case TVT alphabet, each letter of which is generated via a 7 by 9 dot matrix. If two "undots" (using the terminology of Don Lancaster's *TV Typewriter Cookbook*) are appended following the seventh dot position of each line of each letter, each letter will require 9 dots of width on the tv screen. The alphabet could be stored in a ROM, the dot pattern of each letter being represented by 9 bytes. Figure 2 shows 9 bytes representing the letter "A". Figure 3 shows how the string "even spacing" would be displayed using this alphabet. Notice how much empty space appears on both sides of the letter "i". This is because each letter is centered left-to-right in the matrix which is 7 dots wide.

Wouldn't it be an improvement to move the dot pattern of each letter as far to the left as possible, within the matrix, and display each letter in a variable-width format? This could be done conveniently by preceding the first byte of the dot code for each letter (in the ROM) by an extra byte indicating the width of that character. (Or perhaps the unused low-order bits of each code group could be used instead.) Figure 4 shows the 10 bytes of ROM that would then represent the letter "A". The first byte indicates a width of 5 dots for that letter. Two undots are understood to follow the rightmost dot of each letter, but are not included in the width value. Figure 5 shows how the string "Variable spacing" would be displayed using this scheme.

It can be seen that about 50% more letters can be displayed on one line by use of variable spacing. This format is also easier to read. There are complications, though. Hardware would have to be added to the TVT to latch the width values and adjust the character-generation timing accordingly. Moreover, it would be necessary to keep track of cumulative width values in the current line, tocontrol line format (e.g., if the Basic TAB function were to be used). But this would be a small price to pay for the benefits obtained.

Figure 1. Typical Dual-Case TVT Alphabet



```
00111000
01000100
01000100
01111100
01000100
01000100
01000100
00000000
00000000
```

Figure 2. Nine bytes representing an "A".

```
00000101
01110000
10001000
10001000
11111000
10001000
10001000
10001000
00000000
00000000
```

Figure 4. Ten bytes including width value.

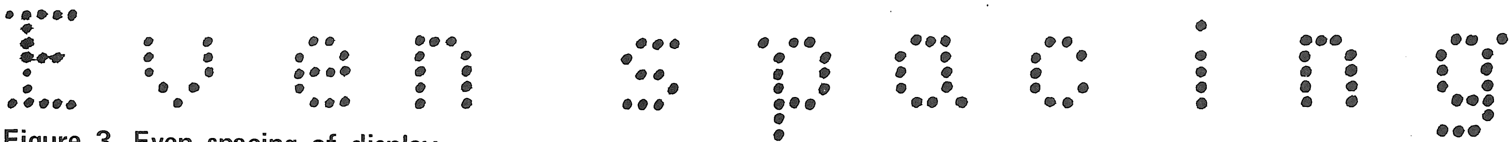


Figure 3. Even spacing of display.

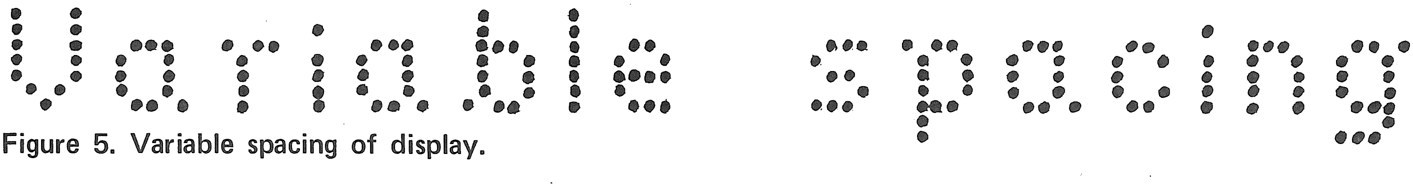


Figure 5. Variable spacing of display.

TVT-II Mods to get 64 characters per line

by David O. Valliere
Digital Designs
Box 4241
Victoria TX 77901

Dear Editor: *May 10, 1976*

If you are using your TVT-II as a computer I/O you may have found the 32 character/line format somewhat limiting. By making minor modifications to the TVT-II board you can lengthen the 32 character line to 64 characters/line, and thereby expand your system's capabilities.

Here are installation instructions for my 32 to 64 character/line TVT-II modification board and 2K memory board. The modifications can be made easily by wire wrapping or a set of boards can be purchased. My TVT-II has been modified since early October, and I am using a very old tv with no bandpass problems.

My board manufacturer is tooled up for manufacturing the boards and can guarantee shipment within 3 weeks after receiving orders. I also have layouts completed for an uppercase/lowercase auxiliary board for the TVT-II, as well as the computer-controlled cursor interface. These boards will also be provided if there is enough interest.

Board prices are $5 for the auxiliary board, $12 for the 2K memory board, and $16 for the set. Shipping is included in these prices. Texas residents add 5% tax. Please make checks payable to Digital Designs.

Sincerely yours,

David O. Valliere *Box 4241*
Digital Designs *Victoria TX 77901*

The TVT-II memory is continuously being addressed through nine address lines to generate the video data used by the teelvision display. The tenth address line (A9) is used to switch from page one to page two. By using the A9 address line for continuous addressing, the TVT-II con be modified to display 64 characters/line. Since the additional 512 characters being displayed are what used to be page two, additional memory will have to be added to provide storage of a second page.

HOW IT WORKS

The basic design of the TVT-II make the modifications required to make it display 64 characters/line quite simple. IC21 and IC14 on the main TVT-II board normally count up 32 characters and upon reaching the 33rd count, pin 11, IC14 and address AO go high. This disables the "dot clock" until the next line is started. Being in the 33rd character position also enables the video blanking circuit through IC12C and IC5B. The line is blanked until a new line is started. By allowing the video generation and the "dot clock" to continue operating until the 65th character position is reached, 64 characters/line will be counted. This can be done by disconnecting pin 11, IC14 from the video blanking circuit and connecting it to address line A9, after having disconnected A9 from the page 1-2 flip-flop. Pin 11, IC14 is also tied to pin 14, the input of the unused counter is IC14 whose output (pin 12) is then tied to the video blanking circuit. Thus we have effectively added an additional 32 counts to the address lines through pin 12, IC14 and transferred the video blanking function to the 65th character position. Since the RC oscillator network of the "dot clock," IC18B, was originally tuned for 32 characters/line, capacitor C4 will have to be replaced with an 18 pF unit to provide for 64 characters/line.

We are now addressing through ten lines/page. The cursor-compare circuitry must be modified to provide comparison of the A9 address bit. This modification will require providing an additional cursor-position count-bit and a comparator. The designer used a 74193 BCD counter to allow preloading the additional cursor bit through a computer cursor position interface. The additional 74193 is attached to the carry and borrow bits of the original cursor counter, IC35, after disconnecting them from the 5th-bit flip-flop, IC27A. Carry and borrow bits are generated by the new counter through NAND gates IC4A and IC4B, and are sent to the original 5th-bit flip-flop IC27A. The cursor bount bit is tied to pin 15, IC42, on the main board and compared with the A4 address bit. The output of the 5th-bit flip-flop IC27A which was originally compared with the A4 address is brought on to the new circuitry and compared with address A9 by the comparator. The cascaded "=" pulse from IC 42 on the main board is input to the comparator. The output "=" pulse is sent to IC41. This provides an additional cursor count bit which is compared with address A4. The new A9 address is compared to the old 5th-bit flip-flop whose output has become the 6th-bit count. IC42 and IC41 on the main board and the new comparator provide the pulse required to position the cursor on the 64 character line.

An additional six 2102's will be required to store a second page of data. By tying the CE pins of each group of memories to pins 8 and 9 of the page flip-flop, IC27B, the pages will roll over as originally designed.

[Editor's Note: We have omitted eight pages, containing instructions for assembly, memroy modifications, 2K memory, piggybacking, early TVT-II mods, start-up, and schematics. Those interested should write to Digital Designs for complete details.]

PARTS LIST

64 Character Board
- one 74193
- one 7485
- one 7404
- one 7400
- one 0.10 mfd disc
- one 18 pf
- Wire, 26 Ga.

2K Memory Board
- twelve 2102 memories
- fourteen 0.10 mfd capacitors
- two 2102 memories (optional)
- two 15-pin Molex board connectors

The auxiliary board and 2K memory boards are available from Digital Design. Both boards are Milspec with tin/lead fused plating and silk-screen component placement. The auxiliary board is single-sided whereas the 2K board is double-sided with plated-through holes.

Shipment within 3 weeks is guaranteed.

CENTRAL OKLAHOMA COMPUTER GROUP

The Central Oklahoma Amateur Computing Association (CENOACA) organized in January. It now has about 30 members. It meets the 2nd Saturday of each month at 10 a.m. in the Oklahoma City Warr Acres Branch Library, NW 63d & MacArthur. It has programming seminars & workshops in addition to the monthly meetings. For details, contact: Lee Lilly, Box 2213, Norman OK 73069.

HOMEBREW TV DISPLAY WITH GRAPHICS

by Glendon Smith

Gentlepersons: May 20, 1976

This is a short description of a tv display circuit I use in my Altair 8800. Although I have made only limited use of the graphics capability, it should be useful, as is, for games requiring a playing board. With synchronization as discussed, fast games should be clearer.

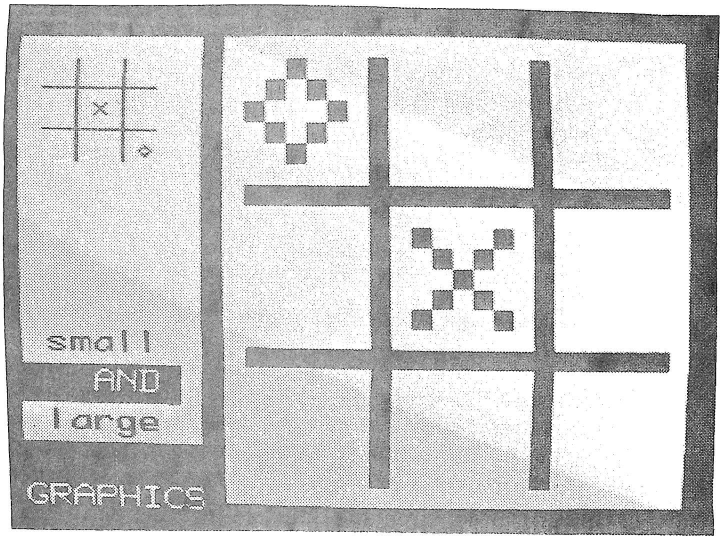
Others may wish to make changes in the logic design. It was sometimes the result of space limitations. If fast data selectors are used as specified, the memory probably can run without wait states.

Sincerely,

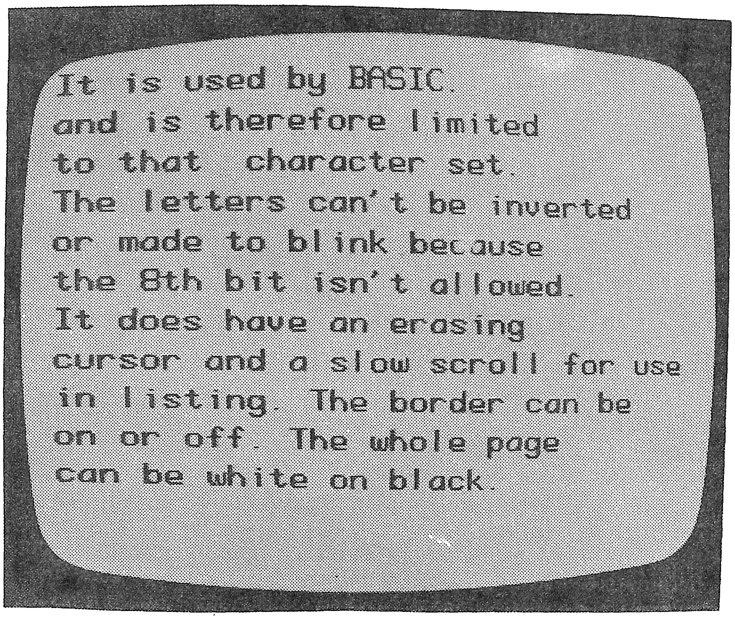
Glendon C. Smith
5822 Daffodil
Dayton OH 45449
513-435-0214

The tv display described in this report is intended for direct plug-in to the bus of an Altair 8800 or other similar microcomputer. The circuits could be adapted to CPU's other than the 8080.

This display differs from the tv typewriter circuit in three major areas. 1) The screen refresh memory is connected to the bus when it is being loaded or altered. 2) The display can produce 128 characters stored in a Motorola ROM (12 lines of up to 32 characters each) and/or up to 128 graphic shapes (8x8 picture elements) stored in RAM (24 lines of 32 shapes). 3) A crystal-controlled commercial sync generator IC is used to provide vertical interlace and a jitter-free display. Other features include the ability to have the 8th bit in the byte used to specify a character or a graphic shape, the ability to cause that character or shape to blink or to reverse itself, the ability to reverse the entire display by software, the ability to display a boarder, and the capability of having software scrolls, an erasing cursor, or other custom features.



As presently implemented, switching from refresh operation to the bus is not synchronized with the blanking for borders so that an insertion of a character causes the loss of about two sweep lines (a white or dark band about 1 mm wide provides notice that a letter was written). This is not annoying to those who have seen the display. For fast games it might be advisable to switch the memories back to the bus during FIELD, and dealy the CPU if these memories are addressed during FIELD. This would slow the display slightly. Without synchronization, a software line feed or scroll up (moving 384 characters) takes about 10 milliseconds or about one-half of a vertical sweep of the tv screen.

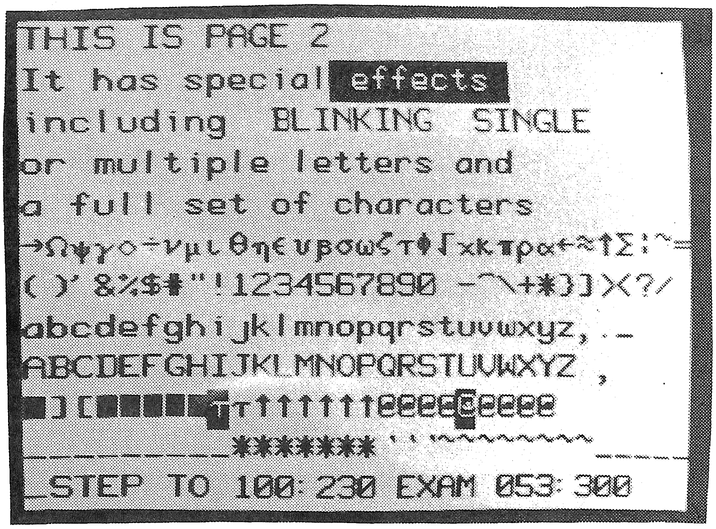


It is used by BASIC.
and is therefore limited
to that character set.
The letters can't be inverted
or made to blink because
the 8th bit isn't allowed.
It does have an erasing
cursor and a slow scroll for use
in listing. The border can be
on or off. The whole page
can be white on black.

The construction of the prototype of this display was eased by using two commercially available boards (and associated components) which were connected together by hinged bars the length of the connector spacing on my Altair mother board. The memory board (MB-2 from Solid State Music) has its copper traces connecting all 8 of the 2102's comprising a bank (1K x 8 bits) before connecting the next bank. Before mounting the sockets it is necessary to cut many copper traces between banks. The bank nearest the bus connector will become bank 0 (lowest address). It is not used by the tv display.

The next higher bank (bank 1) stores the 128 graphic shapes (8x8 bits each). Bank 2 stores the codes for the graphic shapes (24x32 bytes) and has some space which may be used for subroutines. If the graphic capabilities are not being used all three lower banks may be used as part of main memory. The highest, bank 3, stores up to 1024 characters which may be arranged as 32 lines (only 12 displayed) of 32 characters, or as two pages with enough space left over for routines which write on either page (page 1 has scrolling, cursor, etc.). The latter system is the one I have used thus far, but I can imagine applications such as text editing which might use several K of memory for character storage with more elaborate scrolling schemes.

The other board used is a Universal I/O Board (IO-1)



```
THIS IS PAGE 2
It has special effects
including  BLINKING  SINGLE
or multiple letters and
a full set of characters
→Ωψγ◇÷νμιθηϵυρσωζτφΓxκπρα←≈↑Σ!~=
( )' &%$#"!1234567890 -^\+*}]X?/
abcdefghijklmnopqrstuvwxyz, .-
ABCDEFGHIJKLMNOPQRSTUVWXYZ  ,
■][■■■■■■T↑↑↑↑↑↑↑eeeee@eeee
________*******  ''~~~~~~~~~~~   ____
_STEP TO 100:230 EXAM 053:300
```

from Solid State Music. It just barely has ~~spacefor~~ all the circuits for the tv display plus one INPUT PORT for a keyboard (Clare-Pendar). Eight pieces of 8 or 10 conductor ribbon cable handle the interconnections between boards and help in keeping the bits in order.

One of the changes to the memory board which is not shown in the diagrams concerns chip enable and R/W̅ inputs to the 2102's. Pin 3 of each bank of 2102's was disconnected from pin 11 of 7400 A and now receives its input from one of the address selectors as shown. Pin 12 on the 74L42A was ungrounded and connected to pin 11 of 7400 A. The outputs of the 74L42A then became R/W̅ signals feeding the address selectors and the pin 13's of the 2102's formerly connected here are all connected to ground so the chip outputs are enabled.

Several other points will come up in preparing the Solid State Music boards for this use. The designer of the I/O Universal board ran +5 and gnd lines to many positions, expecing you to use 16 or 14 pin 1C's there. However, he did not leave a space between the ends of the 1C positions as their length requires, so many of these traces must be cut before sockets are installed. Because the output port (200 octal in my system) does not need an output connector, traces to this 14 pin pad are cut and a 1C is installed there. On the memory board MB-Z, all the data input lines are left intact as are all the address lines from the connector to the nearest 1K bank of memory (which will become bank 0). All the data outputs are isolated by cutting the traces at appropriate points, as are the address lines to banks 1, 2, and 3. The chip enable and R/W̅ lines are discussed above.

Other arrangements of the 2K of memory used in the generation of the graphics portion might be useful. For example, a 128x96 display of individually addressable points (each point, however, is 4 times the area of a picture element of the current display and the blinking and reversing possibilities appear to be out). One might built only the character portion or only the graphic portion (and generate the needed characters). The display described here may stimulate club members to design a special display as a group project and to produce p.c. boards to ease the labor for all concerned.

I use the Hitachi PO-3 12" B&W tv. It is easy to interface, is all solid state with instant on, and is available for about $68. Interface information is available.

OUTPUT PORT - CONTROL BYTE FUNCTIONS
Port (200 octal in my system)

Bit 0	High to display page 2 characters
Bit 1	High to blink preselected characters
Bit 2	High to blink preselected graphics
Bit 3	High to invert (reverse) preselected characters or graphics
Bit 4	High to invert (reverse) entire display
Bit 5	High to display surround (border)
Bit 6	Low to connect bank 3 (character storage) to bus
Bit 7	Low to connect banks 1 and 2 to bus

The 8212 output port is cleared by the front panel switch so that the 3 banks of memory can be dumped (or loaded) without special instructions in existing programs.

SUPPLIERS

MB-2, I0-1 boards and kits

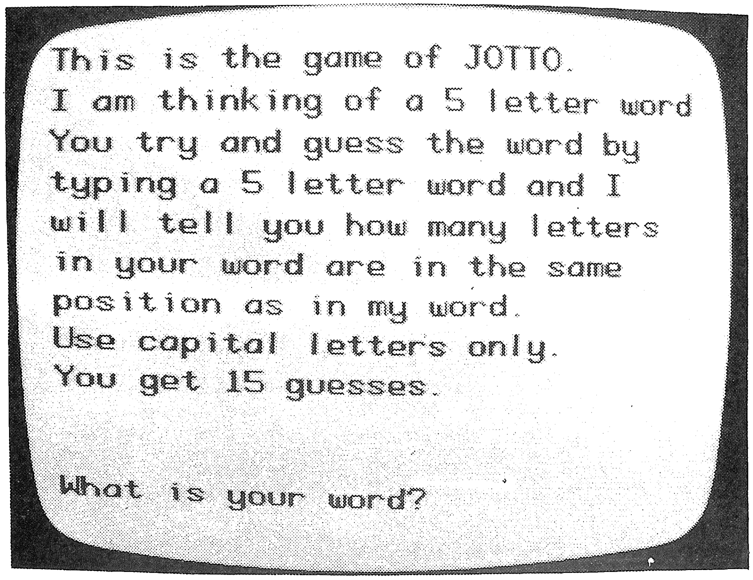
Solid State Music
2102A Walsh Ave.
Santa Clara CA 95050

MIKOS
419 Protofino Dr.
San Carlos CA 94070

6.13635 MHz, 26C Series Crystal @ $5.50 postpaid

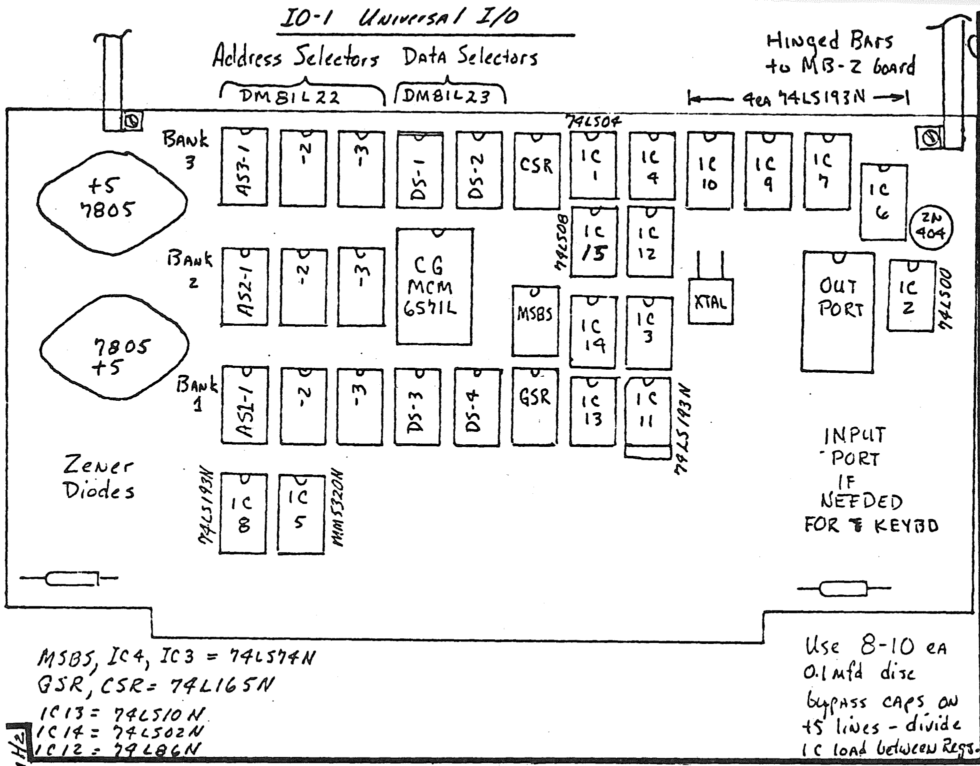
International Crystal Mfg.
10 N. Lee
Oklahoma City OK 73102

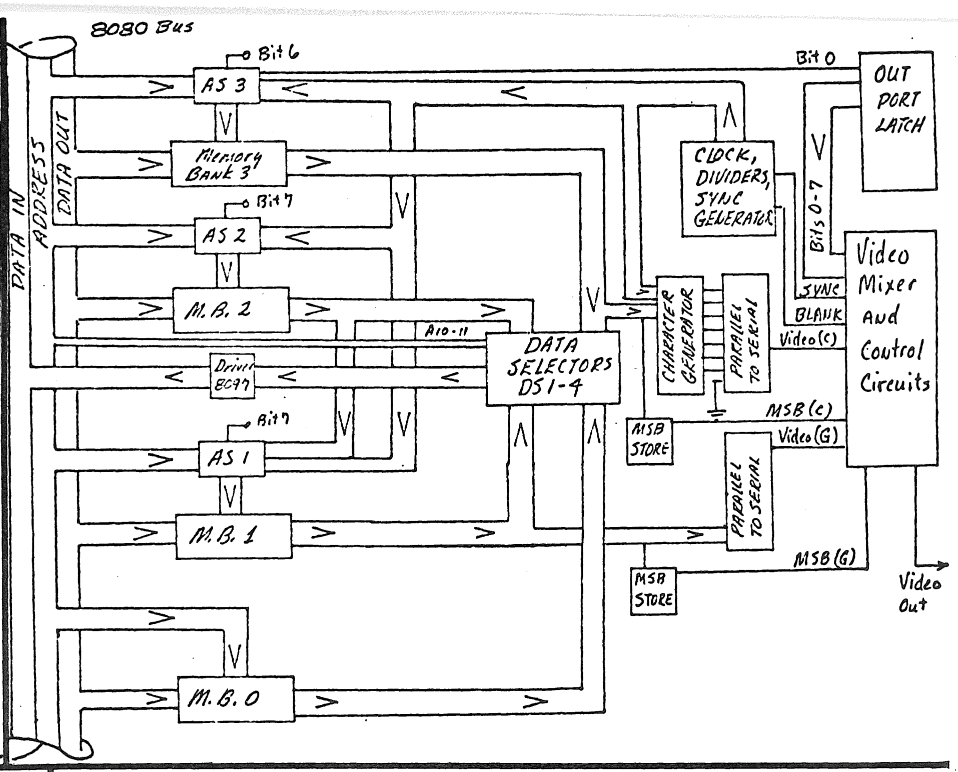
The MCM6571C character gen. came from the Digital Group but I understand that a new version only requires +5 volts. The 5320 (National) Sync Generator ($4) came from Solid State Music, as did most of the I.C.'s.

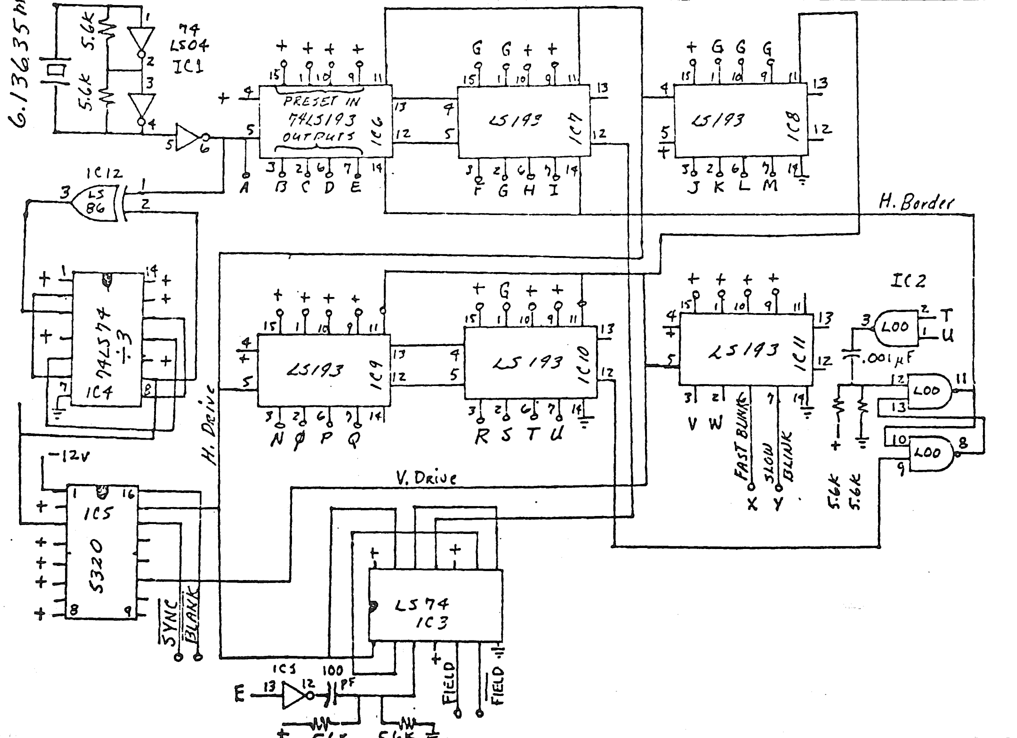


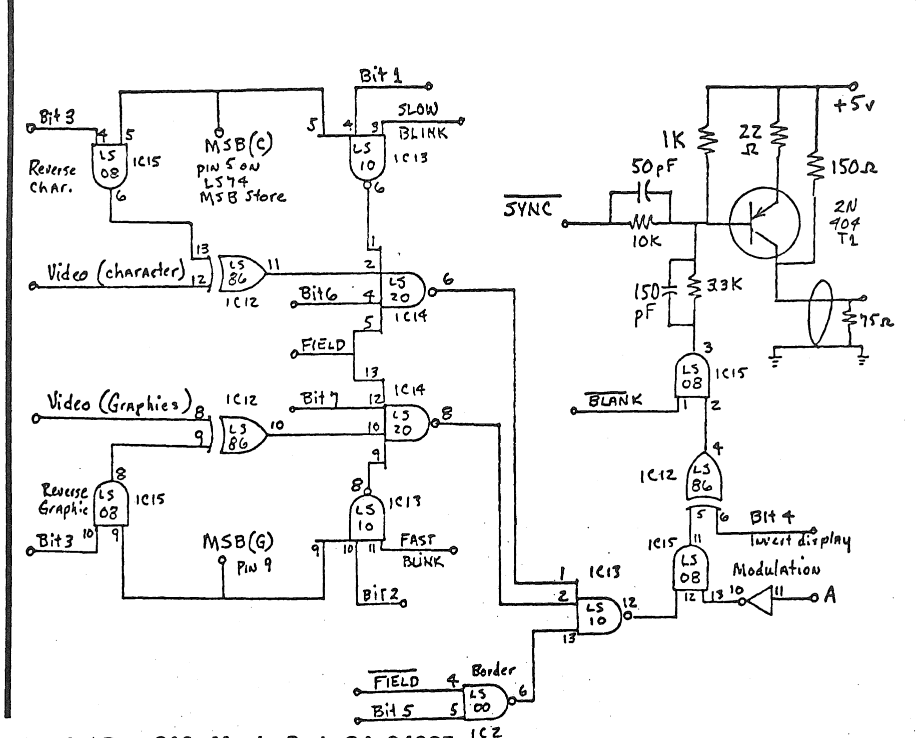
```
This is the game of JOTTO.
I am thinking of a 5 letter word
You try and guess the word by
typing a 5 letter word and I
will tell you how many letters
in your word are in the same
position as in my word.
Use capital letters only.
You get 15 guesses.


What is your word?
```


IO-1 Universal I/O
Address Selectors
Data Selectors
DM81L22
DM81L23
Hinged Bars to MB-Z board
4ea 74LS193N
Bank 3
Bank 2
Bank 1
+5 7805
7805 +5
AS3-1
-2
-3
AS2-1
AS1-1
DS-1
DS-2
DS-3
DS-4
CSR
GSR
CG MCM 6571L
MSBS
74LS04
74LS08
74LS00
74LS193N
MM5320N
IC 1
IC 4
IC 10
IC 9
IC 7
IC 6
IC 15
IC 12
IC 14
IC 3
IC 13
IC 11
IC 2
IC 8
IC 5
2N 404
XTAL
OUT PORT
INPUT PORT IF NEEDED FOR KEYBD
Zener Diodes
MSBS, IC4, IC3 = 74LS74N
GSR, CSR = 74L165N
IC13 = 74LS10N
IC14 = 74LS02N
IC12 = 74L86N
Use 8-10 ea 0.1mfd disc bypass caps on +5 lines - divide IC load between Regs.


8080 Bus
DATA IN
ADDRESS
DATA OUT
Bit 6
Bit 7
Bit 7
Bit 0
AS 3
Memory Bank 3
AS 2
M.B. 2
Driver 8C97
AS 1
M.B. 1
M.B. 0
A10-11
DATA SELECTORS DS1-4
CHARACTER GENERATOR
PARALLEL TO SERIAL
PARALLEL TO SERIAL
MSB STORE
MSB STORE
CLOCK, DIVIDERS, SYNC GENERATOR
OUT PORT LATCH
Bits 0-7
SYNC
BLANK
Video (c)
MSB (c)
Video (G)
MSB (G)
Video Mixer And Control Circuits
Video Out

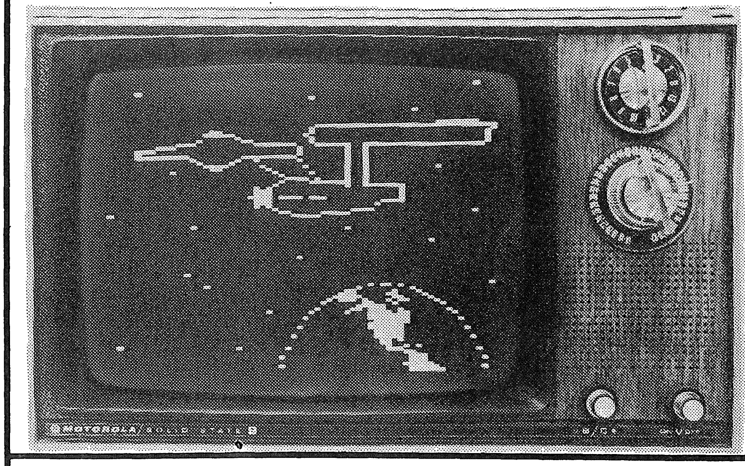

6.136335 mHz
5.6k
5.6k
74 LS04 IC1
PRESET IN 74LS193 OUTPUTS
IC6
LS193
IC7
IC8
IC9
IC10
IC11
A B C D E
F G H I
J K L M
N Ø P Q
R S T U
V W
FAST BLINK
SLOW BLINK
X Y
IC12
LS 86
74LS74 ÷3 IC4
IC2
L00
T U
.001µF
5.6k
5.6k
H. Border
H. Drive
V. Drive
-12v
IC5
5320
SYNC
BLANK
LS74 IC3
FIELD
FIELD
E
IC5
100 PF
5.6K
5.6K

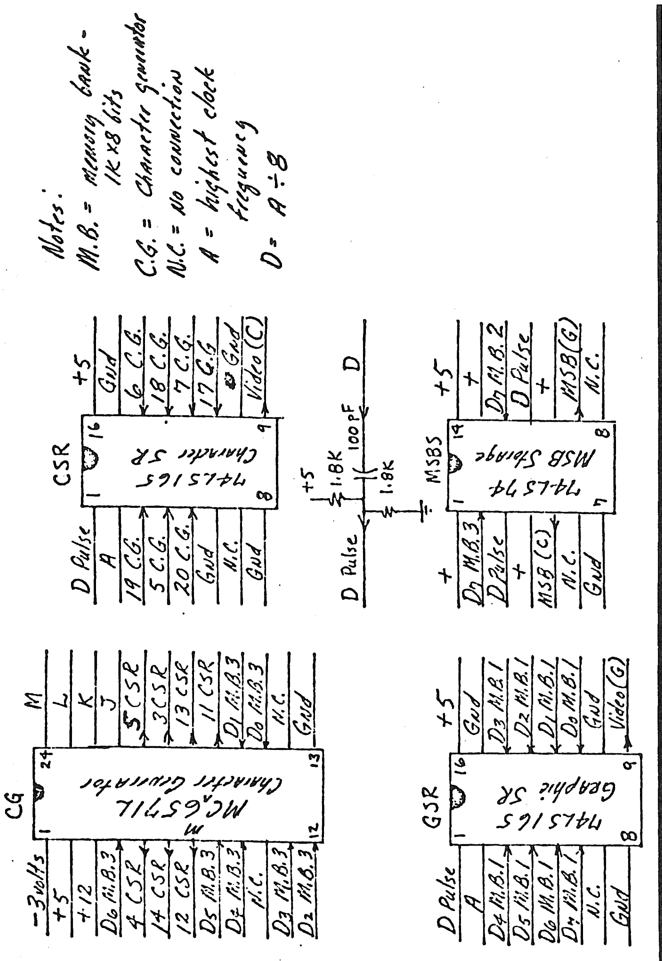

Bit 3
Reverse Char.
LS 08
IC15
MSB(C) PIN 5 ON LS74 MSB Store
Bit 1
SLOW BLINK
LS 10
IC13
Video (character)
LS 86
IC12
Bit 6
LS 20
IC14
FIELD
Video (Graphics)
Bit 7
Reverse Graphic
Bit 3
MSB(G) PIN 9
FAST BLINK
Bit 2
SYNC
1K
50pF
10K
150 pF
33K
22 Ω
150Ω
2N 404 T1
+5v
75Ω
BLANK
Bit 4 Invert display
Modulation
A
FIELD
Bit 5
Border
LS 00
IC2

$98.50 GRAPHICS TERMINAL KIT

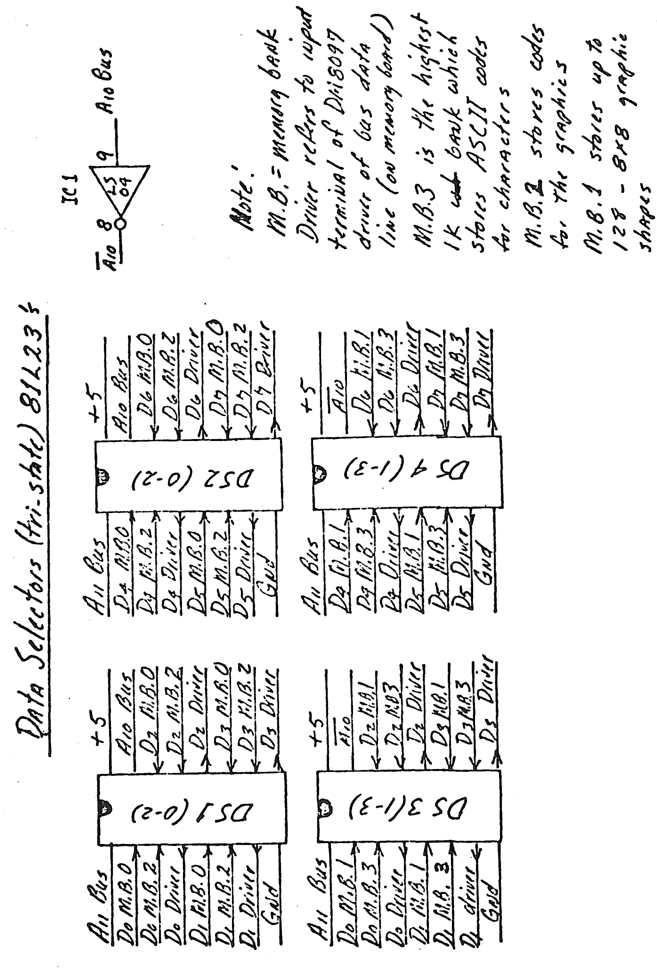
By SWTPC   219 W. Rhapsody   San Antonio, TX 78216

Southwest Technical's GT-61 Graphics Terminal is a low cost graphics unit designed for hobbyists or budget minded commercial applications. The 9 ½" X 13" PC board contains all of the electronics necessary to display an array of cells 64 wide by 96 high on a standard video monitor or modified television set. The graphics terminal contains its own 6144 bit static memory and thus may be driven by any computer system having a TTL compatible 8 bit parallel interface. The unit is available in kit form only and is sold less power supply, chassis, and monitor for $98.50 ppd. in the US. Delivery is 30 days.

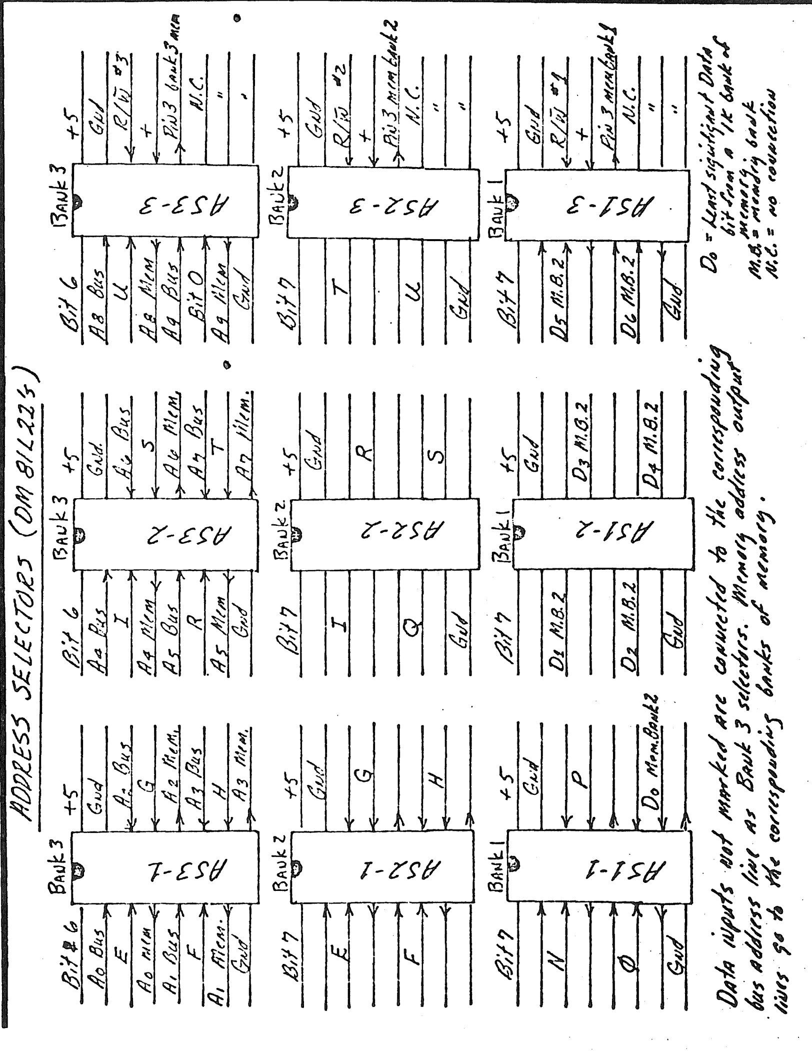




Notes:
M.B. = memory bank - 1K x 8 bits
C.G. = Character generator
N.C. = no connection
A = highest clock frequency
D = A ÷ 8



Note:
M.B. = memory bank
Driver refers to input terminal of DM8097 driver of bus data line (on memory board)
M.B.3 is the highest 1K bank which stores ASCII codes for characters
M.B.2 stores codes for the graphics
M.B.1 stores up to 128 - 8x8 graphic shapes



D0 = Least significant Data bit from a 1K bank of memory.
M.B. = memory bank
N.C. = no connection

Data inputs not marked are connected to the corresponding bus address line as Bank 3 selectors. Memory address output lines go to the corresponding banks of memory.

ERRORS IN & IMPROVEMENTS FOR WHIPPLE'S & ARNOLD'S TINY BASIC EXTENDED (TBX)

Dear Sirs: April 15, 1976

I have noted some errors and possible improvements in Arnold's and Whipple's Tiny BASIC Extended (TBX) [please see *Dr. Dobb's Journal* Vol. 1, Nos. 1 & 2].

A minor reduction could be made at the entry point of the main program by eliminating a jump. The end of the error routine duplicates the initialization, so it could be shortened. These two routines follow (in split octal):

```
INITIALIZATION:                    ERROR:
Address Data Comments              Address Data Comments
000000 061   LXI SP                026275 041   LXI H
     1 377   d1                    026276 002   d1 Entry point of
     2 000   d2                    026277 032   d2 IL program
     3 303   JMP                   026300 061   LXI SP
     4 254   d1                    026301 377   d1
     5 021   d2                    026302 000   d2
021254 041   LXI H                 026303 303   JMP
021255 002   d1 Entry point        026304 257   d1    to IL
021256 032   d2 of IL progrm       026305 021   d2    interpreter
021257 . . . IL interpreter
```

All of the items in the left column could be eliminated, and the entry point could be at the start of the right column, at address 026275. Or, the right column could be replaced by a JMP to address 000000. Or, the two segments could be rearranged as follows:

```
                              . . . . . .
                              (error routine)
                              . . . . . .
Entry point of main program   061    LXI SP
                              377    d1
                              000    d2
This method would             041    LXI H   Entry point of
eliminate 12 bytes.           002    d1      IL Program
                              032    d2
                              . . . . . .
                              IL Interpreter
                              . . . . . .
```

Actually, a lot of extra JMPs and NOPs are to be expected when programming is done in machine language, like TBX was. A primitive assembler, like SPHERE's mini-assembler, which just assembles addresses and some data but not mnemonics, would be all that would be needed to produce a trimmer program.

I should say that I really appreciate the job Arnold and Whipple have done. I'm pointing out a lot of little things, but I think they did a great job.

At a number of places, the character counter advances past spaces. Many bytes could be eliminated by making all of these segments into a subroutine. Such segments are at: 021327, 022324, 023351, 022304, 024100, 027214, 030032, and probably other places.

Subroutine 022147 contains a divide routine. Perhaps this subroutine could be shortened by calling on the other divide subroutine.

Some error jumps, which should be to message number 14 (memory depletion) go instead to error message number 15, which is not defined. This can be corrected by changing addresses 027121, 030350, 030372, and maybe others, from 360 to 355.

The IL Instruction at 033211 is: '266 355 "(".' This means that if the next character isn't "(", address 026355 will be considered the next Interpretive Language (IL) instruction. This will bomb out the program, since 026355 is to be treated as a machine language (ML?) instruction, not IL_x instruction. The instruction at 033211 could be: '233335 "(".' Address 033335 contains a proper instruction, '326352,' which will properly execute the machine language instructions starting at 026352. Incidentally, the address should be 026352, which outputs error message number 13, parentheses error, rather than 026355, which outputs error message number 14, memory depletion. The same problem exists at 032127, 033223, 033241, 033254, 033266, and 033275.

The Random function (RN) should be altered slightly. The random number returned is 16 bits. However, the RN only shifts in 8 new bits each time it is called. Therefore, the upper 8 bits are what the lower 8 bits were the last time RN was used. If address 030210 is changed from 010 to 020, RN will shift in a full 8 bits each time it is called, hopefully making it more random.

When an instruction is being compared to the possibilities, the first word is 'GO,' but the second is not 'to' or 'sub,' the second is compared to '1st,' 'run,' etc., instead of the program immediately indicating unrecognizable statement. This could be fixed by changing the instruction starting at 032057 from '232275 "SUB" ' to '232330 "SUB".' Then 'GO' without 'To' or 'Sub' would go to 'unrecognizable statement' error message.

Thank you for your consideration.

Yours truly,

Charles Skeldon
2320 Co. Rd. I-3
New Brighton MN 55112

1980 CENSUS: HAVE ANY SUGGESTIONS?

The Census Bureau is now actively working on plans for the 1980 census, and important decisions have to be made in the relatively near future.

Although there are many constraints on the census in terms of what and how much information can be collected and tabulated, the Bureau believes that it is very important to obtain and review the recommendations of as wide a range of users and potential users of decennial census data as possible. The Census Bureau is therefore anxious to have the ideas from leaders in mathematics education.

Send suggestions, questions, or comments on the 1980 census to Director, U.S. Bureau of the Census, Washington DC 20233.

Errata/additions to Palo Alto Tiny BASIC

by Lichen Wang

Dear Jim: *23 June 1976*

I have a few miscellaneous items related to the "Palo Alto Tiny BASIC" published in Dr. Dobb's Journal, *Vol. 1, No. 5. First of all, there are a few misprints (my fault). On page 13, right column, second line from the bottom, the minus sign "–" should have been a back arrow "←". The same misprint appeared on page 14, left column, lines 15 and 16.*

Secondly, I forgot to mention that this interpreter actually takes 1.77K bytes. In the list published, I padded it up to 2K bytes, and it can be either in ROM or in RAM. There are 30K bytes unused at the end of the "command table" (Hex 0183-01A0), another 30 unused bytes at the end of the "function table" (Hex 01B3-01D0), and 177 bytes at the end of the I/O routines (Hex 074F-07FF). These unused bytes can be patched to add more commands, and/or more functions, and/or to modify the I/O routines without re-assembly of the whole interpreter. An example follows which adds a video display (VDM by Processor Technology as an alternate output device. When the control-O key is used to turn off the TTY echo and output, the VDM becomes the echo and output device. When the control-O is typed again, echo and output goes back to the TTY, etc. Control-P key is used to clear the VDM screen and text always scrolls up from the bottom of the screen.

The interpreter also needs RAM to store variables, stack, and the Tiny BASIC program. In the published list, 6K of RAM is assumed. You can change this in increments of 256 bytes by changing 9 bytes in the interpreter. These 9 bytes are marked by "@@@@" in the listing.

Last and also least, I have a STARTREK game program coded in Tiny BASIC. It will barely fit in this 6K of RAM. It is probably a very bad example for Tiny BASIC (or any language). In order to squeeze in as much salty stuff as possible, I have abbreviated every command and put as many commands as possible in each line. As a result, the code is almost unreadable. (But it is fun to play!)

Sincerely,

Lichen Wang

NOTE: Wang's StarTrek is being published in the July issue of ***People's Computer Company.***

```
                          ORG   X'071A'
071A C37107        OC2    JMP   VDM3          PATCH FOR VDM OUTPUT
                          ORG   X'074D'
074D C25107               JMP   NZ,VDM1       NOT CONTROL-C
0750 C7                   RSTART              CONTROL-C, RESTART
0751 FE10          VDM1   CMPI  X'10'         IS IT CONTROL-P?
0753 C9                   RET   NZ            NO, RETURN
0754 E5                   PUSH  HL            YES, CLEAR VDM SCREEN
0755 2100CC               LODI  HL,X'CC00'    VDM MEMORY ADDRESS
0758 3EDC                 LODI  A,X'DC'       HI OF ABOVE + 1K
075A 3620          VDM2   LODI  M,' '         BLANK ONE BYTE
075C 23                   INC   HL
075D BC                   CMP   H             VDM MEMORY END?
075E C25A07               JMP   NZ,VDM2       NO, KEEP DOING IT
0761 21C0CF               LODI  HL,X'CFC0'    FINISHED
0764 22BF07               ST    HL,VDMP       SETUP POINTER
0767 AF                   XOR   A             AND VDM REG.
0768 32C107               ST    A,VDMS
076B D3C8                 OUT   X'C8'         VDM PORT #
076D E1                   POP   HL
076E C33207               JMP   CHKIO         GO BACK TO CHKIO
0771 C21F07        VDM3   JMP   NZ,OC3        OCSW ON, GO TO TTY
0774 F1                   POP   AF            OCSW OFF, DO VDM
0775 E5                   PUSH  HL
0776 F5                   PUSH  AF            CHARACTER IN A
0777 2ABF07               LD    HL,VDMP       VDM POINTER IN HL
077A FE0D                 CMPI  @CR           IS IT CR?
077C CA8907               JMP   Z,VDM4        YES, SET UP NEW LINE
077F FE20                 CMPI  X'20'         CHECK OTHER CONTROL CH
0781 DABC07               JMP   C,VDM7        YES, IGNORE
0784 77                   LOD   M,A           NO, PUT IN VDM MEMORY
0785 23                   INC   HL            AND BUMP POINTER
0786 C39307               JMP   VDM5
0789 7D            VDM4   LOD   A,L           COME HERE FOR CR
078A E6C0                 ANDI  X'C0'         GET VDM LINE #
078C C640                 ADDI  X'40'         INCREASE BY 1 LINE
078E 6F                   LOD   L,A
078F 3E00                 LODI  A,0           CARRY INTO H
0791 8C                   ADC   H
0792 67                   LOD   H,A
0793 22BF07        VDM5   ST    HL,VDMP       SAVE POINTER
0796 3E3F                 LODI  A,X'3F'       CHECK LINE OVERFLOW
0798 A5                   AND   L
0799 C2BC07               JMP   NZ,VDM7       NO, RETURN
079C 7C                   LOD   A,H           YES, CHECK END OF 1K
079D FED0                 CMPI  X'D0'
079F C2A707               JMP   NZ,VDM6       OK
07A2 26CC                 LODI  H,X'CC'       WRAP ARROUND
07A4 22BF07               ST    HL,VDMP
07A7 3620          VDM6   LODI  M,' '         BLANK NEXT LINE
07A9 23                   INC   HL
07AA 7D                   LOD   A,L
07AB E63F                 ANDI  X'3F'
07AD C2A707               JMP   NZ,VDM6
07B0 3AC107               LD    A,VDMS        SCROLL UP
07B3 C601                 ADDI  X'01'
07B5 E60F                 ANDI  X'0F'
07B7 32C107               ST    A,VDMS
07BA D3C8                 OUT   X'C8'         VDM PORT #
07BC F1            VDM7   POP   AF
07BD E1                   POP   HL
07BE C9                   RET
07BF C0CF          VDMP   DW    X'CFC0'
07C1 00            VDMS   DB    X'00'
```

ADAPTER MAKES LSI-II's AND 11/03's INTO *REAL* PDP-11's

Able Computer Technology [1538-E East Chestnut St., Santa Ana, CA 92705, (714) 547-6236] is manufacturing a "10001 Univerter". It converts an LSI-11 bus into a DEC Univus, and permits full bidirectional communication between the two. It provides the user with control of all four interrupt levels. It also provides an extended memory map allowing addressing of up to 512K words. The Univerter is a standard quad-width board that can be installed in a PDP-11/03 or an LSI-11 card cage. It is available from stock.

$450 DOT-MATRIX PRINTER FOR 6800's & 8080's 40 Characters/Line, 80 Characters/Second

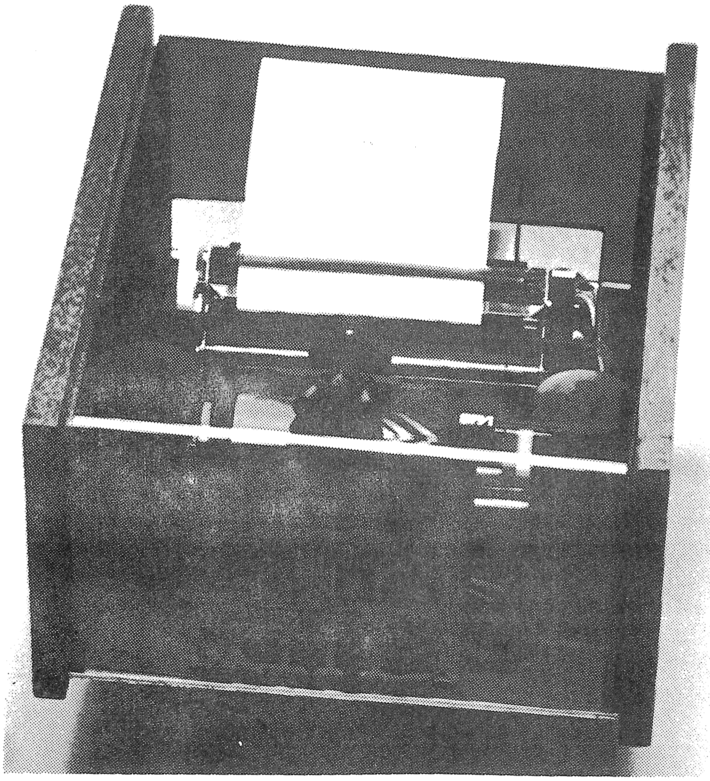
by Electronic Product Associates, Inc. staff

Electronic Product Associates, Inc., 1157 Vega Street, San Diego CA 92110; 714 276-8911, has announced the availability of a new, low-cost, 40-column, dot-matrix impact printer. The printer complete with drive electrincs, character decoding and software driver proms, power supply and attractive hardware and plastic cabinet interfaces directly with the 6800 and 8080 microprocessors. The printer is capable of printing a surprising 80 character per second *bi-directionally*. Single quantity pricing is $450, delivered from stock.

The model 40C utilizes a serially-driven printing element consisting of 7 print solenoids and print wires. The print wires are arranged vertically; the printing element is driven from either direction at constant speed. A synchronous motor driving a spirally grooved drum accomplishes this motion.

Ribbon feed is a simple by-product of printing element motion. Ribbons are inexpensive and easily replaced.

All electronics for driving, decoding and program storage are powered by the self-contained D.C. power supply



MinErrata for MINOL *plus* Tiny TREK

by Erik Mueller
36 Homestead Lane, Roosevelt NJ 08555

June 13, 1976

Here are several errors in the listing of MINOL [please see *Dr. Dobb's Journal*, Vol. 1, No. 5] *which should be corrected:*

Locations:
001 350 should be 242
002 050 should be 273
004 060 should be 107 (o mitted from listing)

Pressing C^c destroys the system (if held down long enough). Fix this by changing the following locations:

```
002 375    303
002 376    111
002 377    hhh
003 000    000
hhh 111    321
hhh 111+1  321
hhh 111+2  321
hhh 111+3  321
hhh 111+4  317
 (etc.)    021
           112
           003
           303
           001
           003
```

hhh 111 is the first address of 11 free locations in user's system.

In my description of the I/O subroutines I meant that the parity bit *(8th bit) must equal one. When I said X^C, I meant control c; X^S means S^C, X^L means L^C.*

The following is an extremely *simplified version of* STAR TREK. *(Text and storage fits in 1.5K.)*

Open Reel *IMSAI/HIT tapes of MINOL 2.1 (along with appropriate read software) are available for $4 from me.*

If I find any more errors, I will write.

Sincerely,
Erik Mueller

ICE-NINE IS ALIVE & WELL IN ILLINOIS

Dear Editor,

Why haven't you listed our club and monthly publication in your fine issues????? Probably because none of our 25 or so members bothered to tell you about us. We are called ICE-NINE INC. A not-for-profit organization formed a year or so ago for mutual computer oriented interests. We have pooled our resources and purchased a Sphere System 40 with floppy discs, line printers, etc. We have our own telephone number for time-share callers and have even set up a radio repeater station (licensed through our amateur radio members) to allow computer use from distances up to 60 miles through amateur tranceivers and remote TTY units.

We are looking for prospective members in the Chicago area and have a huge amount of programs in BASIC and FORTRAN for exchange with other organizations.

C. Cassiouceous
ICE-NINE INC.

Box 291
Western Springs IL 60558

Tiny TREK

```
 1 PR"*TINY TREK*"
 2 D=!/3:W=!/1Ø+9:L=255
 3 X=1:A=Ø
 4 J=1
 5 (12,X-1*8+J+200)=Ø
 6 J=J+1:IF J<8;GOTO5
 7 X=X+1:IF X<8;GOTO4:X=1
 8 J=1
 9 IF !<15Ø;GOTO1Ø:(12,X-1*8+J+2ØØ)=!/155+1:IF(12,X-1*8+J+2ØØ)=2;A=A+1
 1Ø J=J+1:IFJ<8;GOTO9
 11 X=X+1:IFX<8;GOTO8
 12 E=!/38+1:F=!/38+1
 13 (12,E-1*8+F+200)=3:IF 150<!;GOTO16
 14 S=!/38+1:T=!/38+1
 15 (12,S-1*8+T+200)=4
 16 IF W<A;IF W<11;GOTO3
 17 IF S<E;C=E-S:IF T<F;G=F-T:D=D+1
 18 IF E<S;C=S-E:IF F<T;G=T-F
 19 IF C<2;IF G<2;L=255
 2Ø PR"   1234567"
 21 X=1 : K=Ø
 22 J=1
 23 C=(12,X-1*8+J+200)
 24 IF C=Ø;PR" ";
 25 IF C=1;PR".";
 26 IF C=2;PR"K";
 27 IF C=3;PR"E";
 28 IF C=4;PR"B";
 29 IF C=2;K=K+1
 3Ø J=J+1:IF J<8;GOTO23:PR"       ";
 31 IF X=2;PR"SECTOR    ";E;F
 32 IF X=3;PR"STARDATE ";D
 33 IF X=4;PR"ENERGY   ";L
 34 IF X=5;PR"KLINGONS ";W
 35 IF X=6;PR"CONDITION";
 36 IF X=6;IF K=Ø;PR" GREEN"
 37 IFX=6;IF Ø<K;PR" *RED*"
 38 IF X=1;PR:IFX=7;PR
 39 X=X+1:IFX<8;GOTO22:PR
 4Ø IF K=Ø; GOTO 42
 41 H=!/25+1:L=L-H:PRH;"UNIT HIT FROM KLINGONS":GOTO 5Ø
 42 PR"                                             " (Because I have a TVT)
 5Ø IF W=Ø; GOTO 17Ø
 51 IF D=Ø; GOTO 18Ø
 52 IF L<60; GOTO 18Ø
 53 PR"COMMAND";IN A
 6Ø IF A=3;GOTO 15Ø
 7Ø IF A=2;GOTO14Ø
 1ØØ PR"WHAT SECTOR DO YOU WANT TO GO TO?"
 1Ø1 R=1Ø4:GOTO 2Ø1
 1Ø4 IF (12,M-!*8+N+200) # Ø;GOTO12Ø
 1Ø5 (12,E-1*8+F+2ØØ)=Ø:(12,M-1*8+N+200)=3
 1Ø6 E=M:F=N:G=G*3:L=L-G:GOTO 17
 12Ø X=Ø  (Restores position on TVT when incorrect data is entered)
 121 PR:X=X+1:IF X<13;GOTO121:GOTO 1ØØ
 14Ø L=L-6:PR:PR:PR:GOTO 3
 15Ø Pn"WHAT SECTOR TO FIRE AT?"
 151 R=155:GOTO 2Ø1
 155 IF !<30; GOTO 16Ø (Random miss)
 156 IF(12,M-1*8+N+2ØØ)=Ø:W=W-1
 157 IF (12,M-1*8+N+2ØØ)=Ø
 16Ø G=G*4:L=L-G:GOTO 17
 17Ø PR"YOU WIN!!!"
 18Ø PR"YOU LOSE!!"
 2Ø1 IN M,N:IF E<M;C=M-E:IF M<E;C=E-M
 2Ø3 IF F<N;G=N-F:IF N<F;G=F-N
 2Ø4 C=C*C:G=G*G:C=C+G:G=Ø
 2Ø5 G=G+1:IF G*G<C;GOTO 2Ø5:G=G-1:GOTOR
```

1) This game is *not* perfect.
2) It is super-simple.
3) There are three commands:
    1. Move to different sector within quadrant.
    2. Move to different quadrant.
    3. Fire at a specified sector.
4) Energy is refuelled upon *diagonal* docking with a starbase.
5) E = Enterprise
   K = Klingon
   B = Starbase
   . = Star
6) Yes, you can fire phasers and go through stars.
7) Don't get upset if the quadrant you're in doesn't have a starbase (there aren't starbases in every quadrant).
8) Don't get upset if your energy is refuelled even if you aren't docked with a starbase.
9) Don't get upset if anything weird happens.

Button, Button in 8080 machine code

by Ron Santore

Here's the game of BUTTON, BUTTON written in 8080 machine language for computer and terminal. (Altair & TVT or TTY, etc.)

NOTES:

1. Just load the programming instructions in locations 000,000 through 000,377.

2. Then load the text in locations 001,000 through 004,377. Be *sure* that after each paragraph of text, you type the asterisk as I've shown because it's used as a return queue.

3. The program as is takes a little over 1K of memory but it will easily fit into 1K by just shortening the text. You might want to change the text anyway to fit your own (computers') personality.

4. If you have any questions, write or call me (person-to-person): Ron Santore
1957 Huasna Dr.
San Luis Obispo CA 93401
(805) 544-1956

```
000000  061  LXI SP
   001  XXX  your highest memory
   002  XXX
   003  021  LXI D/E
   004  000  instructions
   005  001
   006  315  CALL
   007  347  print subr.
   010  000
   011  315  CALL
   012  103  input subr.
   013  000
   014  016  MVIC
   015  060  zero (ASCII)
   016  315  CALL
   017  210  rnd. subr.
   020  000
   021  107  MOV A to B
   022  021  LXI D/E
   023  020  "whos got the button"
   024  002
   025  315  CALL
   026  347  print
   027  000
   030  014  INR C
   031  171  MOV C to A
   032  062  STA
   033  354  store turn # in text
   034  003
   035  376  CPI A
   036  066  six (ASCII)
   037  312  JZ
   040  330  "you lost"
   041  000
   042  315  CALL
   043  103  input
   044  000
   045  270  CMP A to B
   046  312  JZ
   047  300  "right you are"
   050  000
   051  074  INR A
   052  346  ANI
   053  007
   054  270  CMP A to B
   055  312  JZ
   056  151  "neighbor has it"
   057  000
   060  075  DCR A
   061  075  DCR A
   062  346  ANI
   063  007
   064  270  CMP A to B
```

```
   065  312  JZ
   066  151  "neighbor has it"
   067  000
   070  021  LXI D/E
   071  360  "who me"
   072  002
   073  315  CALL
   074  347  print
   075  000
   076  000  NOP
   077  000  NOP
000100  303  JMP
   101  030
   102  000
000103  333  IN
   104  000  status word
   105  017  RRC
   106  332  JC
   107  103
   110  000
   111  333  IN
   112  001
   113  376  CPI
   114  107  "g" (ASCII)
   115  310  RZ
   116  376  CPI
   117  131  "y" (ASCII)
   120  312  JZ
   121  014
   122  000
   123  376  CPI
   124  116  "n" (ASCII)
   125  312  JZ
   126  367  end subr.
   127  000
   130  376  CPI
   131  070  "8" (ASCII)
   132  372  JM
   133  146
   134  000
   135  021  LXI D/E
   136  072  "no such number"
   137  003
   140  315  CALL
   141  347  print
   142  000
   143  303  JMP
   144  103  input
   145  000
000146  346  ANI
   147  007
   150  311  RET
```

```
000151  021  LXI D/E
   152  240  "neighbor has it"
   153  002
   154  315  CALL
   155  347  print
   156  000
   157  315  CALL
   160  210  rnd subr.
   161  000
   162  376  CPI
   163  003  "3" (Binary)
   164  372  JM
   165  200  pass higher
   166  000
   167  005  DCR B
   170  170  MOV B to A
   171  346  ANI
   172  007
   173  107  MOV A to B
   174  303  JMP
   175  030
   176  000
   177  000  NOP
000200  004  INR B
   201  170  MOV B to A
   202  346  ANI
   203  007
   204  107  MOV A to B
   205  303  JMP
   206  030
   207  000
   210  041  LXI H/L
   211  265
   212  000
   213  026  MVID
   214  010  "8" (Binary)
   215  176  MOV M to A
   216  007  RLC
   217  007  RLC
   220  007  RLC
   221  256  XRA M
   222  027  RAL
   223  027  RAL
   224  055  DCR L
   225  055  DCR L
   226  055  DCR L
   227  176  MOV M to A
   230  027  RAL
   231  167  MOV A to M
   232  054  INR L
   233  176  MOV M to A
   234  027  RAL
   235  167  MOV A to M
   236  054  INR L
   237  176  MOV M to A
   240  027  RAL
   241  167  MOV A to M
   242  054  INR L
   243  176  MOV M to A
   244  027  RAL
   245  167  MOV A to M
   246  025  INR L
   247  302  JNZ
   250  216
   251  000
   252  346  ANI
   253  007
   254  376  CPI
   255  010  "8" (Binary)
   256  370  RM
   257  303  JMP
   260  210
   261  000
   262  XXX  any #
   263  XXX  any #
   264  XXX  any #
   265  XXX  any #
   266  000  NOP
   267  000  NOP
   270  000  NOP
   271  000  NOP
   272  000  NOP
   273  000  NOP
   274  000  NOP
   275  000  NOP
   276  000  NOP
   277  000  NOP
000300  171  MOV C to A
   301  376  CPI
   302  063  "3" (ASCII)
   303  372  JM
   304  317
   305  000
   306  021  LXI D/E
   307  320  "you found it"
   310  003
   311  315  CALL
   312  347  print
   313  000
   314  303  JMP
   315  103  input
   316  000
000317  021  LXI D/E
   320  205  right you are
   321  003
   322  315  CALL
   323  347  print
   324  000
   325  303  JMP
   326  103  input
   327  000
000330  170  MOV B to A
   331  366  ORI
   332  060  prefix for ASCII
   333  062  STA
   334  104  store button in text
   335  004
   336  021  LXI D/E
   337  040  "you lost"
   340  004
   341  315  CALL
```

```
   342  347  print
   343  000
   344  315  CALL
   345  103  input
   346  000
000347  333  IN
   350  000  status word
   351  007  RLC
   352  332  JC
   353  347
   354  000
   355  032  LDAX D/E
   356  376  CPI
   357  052  * (ASCII)
   360  310  RZ
```

```
   361  323  OUT
   362  001
   363  023  INX D/E
   364  303  JMP
   365  347
   366  000
000367  021  LXI D/E
   370  220  "thanks for playing"
   371  004
   372  315  CALL
   373  347  print
   374  000
   375  303  JMP
   376  103  input
   377  000
```

ASCII DATA TO BE STORED IN MEMORY

```
ADDRESS                   TEXT

(001,000)       CR        BUTTON, BUTTON CR LF LF
                EIGHT PEOPLE ARE SITTING IN A CR LF
                CIRCLE, WITH YOU IN THE CENTER. CR LF LF
                ONE OF THEM HAS THE BUTTON AND CR LF
                YOU HAVE TO GUESS WHO. CR LF LF
                THE PERSON WITH THE BUTTON CAN CR LF
                PASS IT, SO BE CAREFUL. CR LF LF
                YOU HAVE FIVE GUESSES. CR LF LF
                WHEN YOU'RE READY, TYPE "G". *

(002,020)       CR LF                          0       CR LF
                        BUTTON, BUTTON     7       1     CR LF
                                          6    ?    2     CR LF
                          WHO'S GOT        5       3     CR LF
                         THE BUTTON?           4       CR LF *

(002,240)       CR
                I DON'T HAVE IT, CR LF
                MY NEIGHBOR DOES. CR LF
                BUT WHOEVER HAS IT PASSES IT! CR LF *

(002,360)       CR
                WHO, ME?? CR LF
                I DON'T HAVE IT! CR LF
                WHOEVER HAS IT, KEEPS IT. CR LF *

(003,072)       CR
                SILLY, CR LF
                THERE'S NO ONE HERE CR LF
                WITH THAT NUMBER....TRY AGAIN: CR LF *

(003,205)       CR
                RIGHT YOU ARE; LUCKY! CR LF
                PLAY AGAIN? (Y OR N) *

(003,320)       CR CR LF LF
                YOU FOUND THE BUTTON IN _ TRIES. CR LF
                ANOTHER GAME? (Y OR N) *

(004,040)       CR
                SORRY. THAT WAS YOUR LAST GUESS. CR LF
                "_" HAD THE BUTTON!  TRY AGAIN?  CR LF *

(004,220)       CR
                THANKS FOR PLAYING..............CR LF
                ANYONE ELSE WANT TO PLAY?? CR LF
                (Y OR N) CR LF *
```

DON'T UNDERESTIMATE BASIC

Dear Editor, June 9, 1976

I think anyone who underestimates BASIC in its more sophisticated forms is making a mistake. It is powerful, it can be well organized, and yet a novice can get going very easily. Most important for micros—the time for an amateur or part-time programmer to get a working program is ½ that of other languages.

C.D. Johnson 2801 SW Patton Lane
Forest Products Engineering Portland OR 97201

BASIC COMPLAINT & MACRO MESSAGE

Dear Sir, 5 May 1976

I am very curious about the motivation for including the article "A Critical Look at BASIC" by Dennis Allison in *Dr. Dobb's Journal* Vol. 1, No. 2. This article is the first one I have encountered in the computer hobbyist press that talks about modular and structured programming. This may be because, as your editorial says, that most other magazines are hardware oriented. In any event, Allison's article confirms what I have long suspected, namely, that BASIC is not the language of choice for state of the art programming. However, the inclusion of Allison's article in a magazine whose *raison d'etre* is to promote a subset of BASIC does seem a bit odd, to say the least.

Allison's article raises some questions that neither *Dr. Dobb's Journal* nor *PCC* seem to answer, namely, if BASIC is bad for you, why encourage people to be BASIC junkies?

Let me note that I am *not* a rabid BASIC hater; just troubled by the difference between what we are supposed to do, and what we actually do.

Those who advocate structured programming seem also to advocate language with lots of control structures. Lots of control structures sounds like a big language to me. Big languages are OK if you have megabytes of core, but obviously aren't very good if you're a hobbyist with 2K. Structured programming seems precluded by the limitations of a minimal hobbyist system. Is the hobbyist with a modest system limited to assembler or a language with not much more than GOTO's and a conditional branch? Or, is there some kind of a happy compromise between Tiny BASIC and, say, PL/I? I would certainly like to see *DDJ* address some of these issues.

A final suggestion. The assembler I use at work doesn't have any macro facilities. The other day, I decided to see what I could do about this. The macro generator GPM described by Wegener in his book, *Programming Languages, Information Structures, and Machine Organization*, looked interesting. I looked up the original article on the language (Strachey, "A General Purpose Macrogenerator," *The Computer Journal*, Oct., 1965, Vol. 8, No. 3, pp. 225-241) and discovered a listing for a GPM processor written in CPL. Strachey says the original implementation of GPM was 250 "orders" long. This is one hell of a lot of macrogenerator per word of core. Thus GPM might be of interest to people with home brew assemblers. Sounds like the sort of thing *DDJ* might be interested in. My implementation was a "quick and dirty" FORTRAN job done on the sly. As you might expect, Strachey's program has bugs in it. Some are real boo-boos.

Yours,
Fred J. Dickey 3420 Granville Rd
Westerville OH 43081

There is a lot wrong with BASIC; it is not the language of choice when the program is going to be long or complex. Unfortunately, there is a substantial group of people who do not understand that; hence, the publication of my "Critical Look At BASIC." I had hoped that it would help our audience (many of whom have only recently encountered any programming language) attain a bit of perspective on what BASIC is and where it belongs in the spectrum of things.

There is a lot right with BASIC, too. For small programs its interactive capabilities outweight the cumbersome control structures. Its "text editor" orientation makes it easy to implement interactively with an interpreter. Given the spectrum of available language models, it is difficult to see how any other language could have been a better model for a super-minimal implementation. Tiny BASIC is about right—and an one is going to write a giant tiny BASIC program (I hope!).

Macro processors are magnificent tools with frightening powers and capability. The problem is how to make sure that a macro, particularly one in Strachey's GPM, does what you think it does. I would hazard a guess that some of the "bugs" you have found in the published version are, in fact, simply unexpected macro expansions which are performed according to the rules. I'd suggest that you look at another MACRO system—the TRAC system. There is a good description in Nelson's *Computer Lib*. The FORTH language and Logical Machine Corporation's ADAM are also macro-like systems, but they defer expansion to run-time. We'd be pleased to publish macro systems implementations should anyone be willing to prepare them.

Incidentally, macro systems can perform many of the same functions as compliers, but the underlying model is quite different. A compiler decomposes the input text into a phrase structure and then assigns meaning based upon that decomposition. A macro processor matches a template and then transforms the text accordingly. Macro systems are inherently more powerful than compilers modeled on context-free languages since they are (inherently) context-sensitive. A.S. Tenenbaum describes using such a system in *IEEE Transacting on Software Engineering*, SE-2,2, June, 1976, p. 121. --Dennis Allison

COMPUTERS FOR STUDENTS' HOME STUDIES

Dear Mr. Warren: 8 May 1976

We are organizing a research project whose aim is to investigate how small "Home Computers" might be used in education—helping students to study at home. To keep up informed about new developments related to home computers, please enter our subscription to *Dr. Dobb's Journal*.

Do you know of other publications related to home computers?

Sincerely,
Jerry Felson, Ph.D.
President 84-13 168 St
Cybernetic Decision Systems, Inc. Jamaica NY 11432

COMPUTERS-IN-EDUCATION BIBLIOGRAPHY

The National Council of Teachers of Mathematics (NCTM) bibliography, ***Computers in Education***, has replaced the old list, ***Computers in the Mathematics Classroom***. This new listing is separated into seven sections, including one on mathematics texts series.

Single copies of this 41-page bibliography are available free on request from the NCTM Headquarters Office, 1906 Association Dr., Reston VA 22091.

SUMMER MEETING OF THE ASSN. FOR DEVELOP— MENT OF COMPUTER-BASED INSTRUCTIONAL SYSTEMS

The 1976 Summer Meeting of the Association for the Development of Computer-Based Instructional Systems (ADCIS) will be sponsored by Control Data Corporation at Minneapolis, Minnesota, August 10-12, 1976. For further information about the conference, contact the General Program Chairperson: Dr. Karen Duncan, Director, Office of Computer Resources, College of Dental Medicine, 80 Barre Street, Charleston, South Carolina 29401, (803) 792-3211.

HAND-HELD CALCULATORS IN CLASSROOMS

The Iowa Council of Teachers of Mathematics (ICTM) has recently published the Monograph-1976, *The Hand-Held Calculator*. The ideas and activities included were suggested by ICTM members from their classroom experiences.

Copies of this monograph are available for $1.50 (ICTM member), or $2 (nonmember) from Ann Robinson, 509 W 20 St., Cedar Falls IA 50613. Make all checks payable to ICTM.

TINY TIME SHARING???

Dear Editor, 6/2/76

I would like to get readers to start thinking about the possibilities of contructing multiple-user or time-shared systems using table-top hardware.

The development which I think makes this possible is the Video Display Module VDM-1 from Processor Technology Corp. (6200 Hollis St., Emeryville CA 94608). I happen to have designed it, in part for the money, but also so that people more skilled in software than I (and that's almost anyone) could put together multi-user systems.

The VDM-1 is a memory module (1024 bytes) with a window (the video monitor screen). It has an upper/lower case character set which includes control characters (128 characters). There is a video inversion cursor which can be set at each character by setting the high-order bit of that character. This effectively doubles the character set to 256. Display format is 64 characters by 16 lines.

Since it is memory, the processor can read from the VDM as well as write to it. This means that information specific to a given user can be stored in that user's VDM, and pulled out for use when desired, modified, and put back in. This can happen in a memory area which is masked from the view of the user by the "window shade." As its name implies, this is a blanked area of the screen which can be "pulled down" from the top to blank a maximum of 15 text lines. The CPU determines the length of the shade through a status byte which it outputs to the VDM through an OUT instruction.

Suppose that Tiny BASIC (or Tiny ALGOL or Tiny FORTRAN or whatever) is set up in the CPU's main memory area. Several users with VDM's could be building programs, the object code of which is stored in the first few lines of their screens. (Here my ignorance of systems software will probably become laughably apparent. It's the vision that counts.) The CPU runs through a schedule in which it pulls out the object code and tables of parameters in a user's in a user's screen, runs the program until a convenient point is reached, stuffs the code and new parameters back under the window shade, and goes on to the next user. One of the parameters would obviously be the location on the screen of the cursor. If the total number of bytes used for this storage were 512 per user, that would still leave 8 lines of 64 characters. These could be configures as two columns of 32 characters, having a total length of 16 lines.

The more ambitious a user got, the lower the window shade would go as the hidden area filled up with stuff. This would provide a "negative feedback" effect which might serve to keep the user reminded of the limited nature of the machine resources. Users of *I*ncredible *B*ig *M*onster machines will throw tantrums at the thought of this, but they will have to be brought into the real world somehow, whether they like it or not.

I have been talking about a multi-user operation, in which several people use the same program. True time-sharing requires (I think) that each time the CPU steps to the next user, it be able to call up the program (meaning Tiny BASIC or Tiny ALGOL) that that user wants. Clearly these programs cannot be kept under the window shade, but, if they are tiny enough, there should be enough RAM available on a full-blown 65K system (providing the power supply holds out).

Incidentally, it might be a tickle to keep object code and parameters on the screen without pulling the window shade down over them. They would appear to flicker, sparkle and otherwise rearrange themselves in operation. This would be a much better show than black screen, and might serve as a debugging aid, together with a chart of the binary equivalent of the character set.

That's about as much as I can offer, except for help in interpreting the VDM-1 manual, which is available for $4 from PTCO. It's a pretty good manual, so I don't think there will be too much call on that score.

Do it!

Lee Felsenstein
LGC Engineering

1807 Delaware St.
Berkeley CA 94703
415 845-4736

IVERSONS INITIATE APL NEWSLETTER

Dear Editor: 5/24/76

APL Press is a new publishing house devoted exclusively to APL. Its first book, to appear this summer, is a high school text on elementary analysis by Ken Iverson, the inventor or APL. Several other titles are planned for publication this year, and further manuscripts are being sought.

A newsletter is also planned, to present brief articles, problems, definitions of functions, reports on conferences, correspondence, and others items of interest to the APL community. The first issue, which is scheduled for July, will include a report by Professor Jenkins on a recent APL Implementors' Workshop, an article on magic cubes by Professor Mauldon, and material on a new form of function definition excerpted from a forthcoming book.

Readers interested in receiving the newsletter and information on other publications, or in submitting material for publication, should write to APL Press, Box 27, Swarthmore PA 19081.

Jean Iverson

[Jean Iverson is in charge of the APL Press. She is "closely associated" with Ken Iverson. —JCW]

A SOFTWARE EXCHANGE FOR 6800's

Dear Sirs: 5-15-76

I am sponsoring a SOFTWARE EXCHANGE for those interested. Anyone interested in receiving software for any of the microcomputers, send your name, address, and any software you have available. I have some software for the 6800 for immediate distribution. When I receive software from other individuals, I will distribute the material to those interested. Please include $3 to cover the cost of mailing and photocopying. You need not submit software to benefit.

Very truly yours,

Howard Berenbon

2681 Peterboro
W. Bloomfield MI 48033
313 851-7966

We would be happy to save you the cost of photocopying listings and documentation by publishing your 6800 programs in *Dr. Dobb's Journal*. Also, if you don't wnat to spend the time and energy running your software exchange operation, you could submit your programs to Community Computer Center for their non-profit Program Repository & Tape Duplication Facility (please see *Dr. Dobb's Journal* Vol. 1, No. 3).

IMS ASSOCIATES, Inc. recently moved into new facilities which more than quadruple the company's manufacturing space. The company's new address in San Leandro, California, is 14860 Wicks Blvd, 94577; (415) 483-2093. The rapid growth of IMSAI has been attributed to the demand for the new IMSAI 8080 Microcomputer which was introduced earlier this year.

FCC PETITION ON ANSCII TRANSMISSIONS BY HAMS

by Bruce J. Brown, WB4YTU April 19, 1976
4801 Kenmore Ave., no. 1022
Alexandria VA 22304
703 370-1431, home; 202 697-9654, work

This is a petition for rulemaking in the matter of revisions of Federal Communications Commission Rules, Sections 97.69 and 97.117 to permit use of the American National Standard Code for Information Interchange (ANSCII), formerly ASCII.

The American National Standard Code for Information Interchange (ANSCII), formerly ASCII, was developed by the American National Standards Institute (ANSI, formerly American Standards Association 'ASI') as the standard code for information interchange in the United States.[1]

The 7-bit-plus-parity ANSCII code provides 128 possible characters (Figure 1) versus the 58 characters of the Baudot code. In addition to figures, numbers, and punctuation, the code set has provisions for special symbols and control characters which is vital to automated data exchange and computer control.

Its purpose is to establish uniformity and compatibility in the interchange of information among domestic and foreign manufacturers of data processing and communications systems.

In March 1968, President Johnson approved a recommendation by the Secretary of Commerce that ASCII be adopted as a federal standard.[2]

Sections 97.69 and 97.117 are ambiguous and contradictory with regards to codes presently allowed. 97.69(a) states "A single channel five-unit (start-stop) teleprinter code shall be used . . ."; however, Section 97.117 states "The transmission by radio of messages in codes or ciphers . . . is prohibited." These sections are in clear conflict. Furthermore, Section 97.69(a) also states "In general, this code shall conform as nearly as possible to the teleprinter code or codes in common commercial usage in the United States."—which is ANSCII!

There are several arguments to support the use of ANSCII by amateur radio operators.

a. Large quantites of surplus ASCII terminal equipment are available at very low cost on the surplus market. Inexpensive Baudot devices are becoming increasingly difficult to find.

b. Government and industry have only recently begun to explore the use of recently developed microprocessor circuits to solve complex teleprocessing problems. Hobbyists, many who are amateur radio experimenters, have also shown considerable interest in these devices as evidenced by the high-volume microprocessor sales to non-commerical buyers, and the emergence of numerous amateur computer journals. Hams, using microprocessors in concert with presently allocated communications channels, have the opportunity to make serious contributions to the infant teleprocessing field while greatly enhancing current amateur modus operandi. It is not unlikely that hams will some day use microprocessors in communications networks (e.g., packet switching) to permit faster and more reliable traffic handling for emergency and routine messages. Fruition of many of these concepts, however, is directly dependent upon the approval by the FCC of a coding scheme with a large-character set, such as ANSCII, for compatibility with microprocessors and automatic communications systems. Failure to approve such a code will greatly stifle the advancement of non-commercial communications and would be in direct conflict with the purpose from the amateur radio service as expressed in Section 97.1(b) and (c).

c. ANSCII, by virtue of its diversified character set, is highly compatible with amateur telemetry systems; e.g., remotely monitoring the status of repeater control circuits.[3]

Using asynchronous ANSCII transmission with one start, two stop, one parity, and seven data bits per character, speeds of 10, 30, and 60 characters per second will equate to rates of 110, 330, and 660 bits per second (bps), respectively.

Through simple Fourier analysis to the 5th harmonic, it can be shown that the signaling bandwidth for data at speeds of 110, 330, and 660 bps is 220, 660, and 1320 hertz, respectively. Furthermore, it can be shown that the AFSK bandwidth for a 660 bps signal is less than that required for SSTV transmission.

Based upon the technical and operational benefits that the use of ANSCII could provide, and considering that no detrimental effect to the amateur community would result, it is requested that applicable sections to Part 97 be revised to permit the use of ANSCII.

[1] Data Communications Systems, Control Data Corporation, April, 1974, page 47.

[2] Introduction to Computer Data Communications, Honeywell Corporation, July, 1973, pages 2-19.

[3] QST, March 1976, page 73.

A CLUB SURVEY FOR A CLUB CLUB

Dear Editor,

I am doing a survey of hobbyist computer clubs. It should be interesting to find out how many hobbyist club members there are, what kinds of things they're doing, etc. Hopefully the tabulated results can be printed in *DDJ* after I've compiled them. One of the reasons for the survey is to evaluate interest in an organization of hobbyist clubs (tentatively called 'Your Club of Clubs' or 'The Metaclub'). Any club *not* on the following list should get in touch with me for more information.

Amateur Computer Club of N.J., Atlanta Area Microcomputer Hobbyist Club, Bay Area Microprocessor Users Group, Bit Users Association, Cache (Chicago area), Cleveland Digital Group, The Computer Hobbyist Group (N. Texas), Denver Amateur Computer Society, El Paso Computer Group, Homebrew Computer Club, LLLRA Hobbyist Computer Group, Long Island Computer Association, Miami Area Computer Club, CPU (Monterey), Northwest Computer Club (Seattle), Nashau Area Computer Club, New York City Micro Hobbyist Group, Pittsburgh Area Computer Club, Santa Barabra Nameless Computer Club, Southern California Computer Society, Tallahassee Amateur Computer Society.

I also invite comments and questions from anyone interested.

Sincerely,
Dave Caulkins

437 Mundel
Los Altos CA 94022
415 948-5753

WESTERN DATA'S 6502-BASED DATA HANDLER

Complete Kit for $169.95,
Plug-Compatible to Altair Peripherals

by Western Data Systems staff

The Data Handler is Western Data Systems new product. It's a microcomputer using the MOS Technology 6502 microprocessor with the latest state of the art technology producing a high performance microcomputer at a low price.

The high speed operating capabilities of the Data Handler are enabled by the use of an easy-to-use full-function, hardware-controlled, front panel. A large ground plane area (to minimize noise at high operating speed) is on the P.C.B. and 2102-type RAMS.

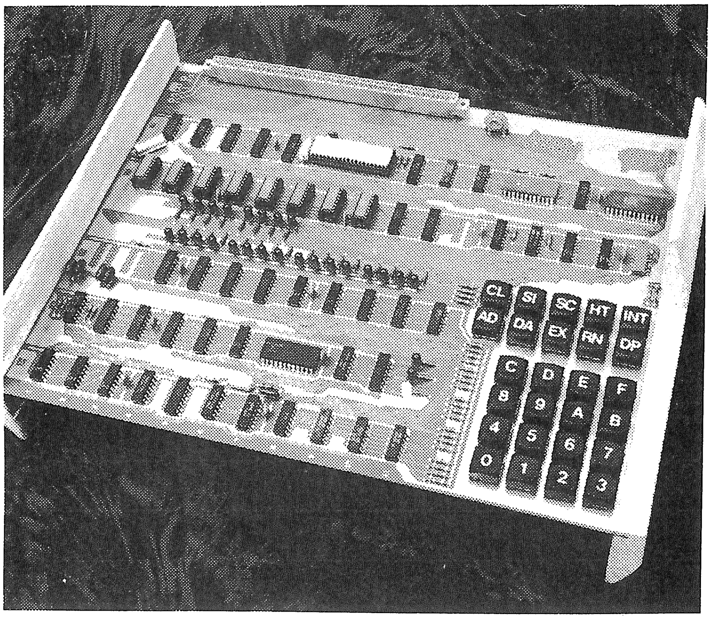
Slower accessing memories (EPROMS and ROMS) may be used, although this will reduce the cycle speed to within the limits of other microcomputer kits. The single 13.75" x 10.5" P.C.B. can directly address 65K of memory and contains 1K bytes of static RAM on the board with complete address decoding.

It also consists of all circuitry needed to be a stand-alone microcomputer for even such high-speed devices as disk peripherals.

The Data Handler is designed with identical drive capabilities around the 8800 Altair, 100-pin, tri-state bus. It's plug-in compatible with the long list of Altair peripherals. Expandability can be accomplished in a manner identical to the 8800 Altair by using the mother board.

The Data Handler also has dual interrupt lines (one maskable), slow-down circuitry for slow memories, DMA (direct memory access) circuitry, and DMA acknowledge control. One 8-bit parallel input port, one 8-bit parallel output port, separate IO address control, and memory control lines. Single voltage (+5 volts) and cycle times to 250ns. It has full front panel control with the use of keyboard switches to provide the following hardware:

Single-cycle operation.
Single-instruction operation.
Memory examine (left incremental).
Memory deposit (left incremental).
Initialization.
Halt.
Run.
Hex data and address entry.



For an introductory offer the Data Handler Bare Bones Kit is being offered for $79.95, which includes the Data Handler P.C.B., 26 keyboard switches, P.C. B. stand, and complete documentation.

The complete kit costs $169.95. This includes the Data Handler P.C.B., 26 keyboard switches, P.C.B. stand, complete set of I.C.'s, 1K static RAM, 500ns memory, resistors, capacitors, L.E.D.'s, 1 mhz 6502, and complete documentation. This microcomputer is ideal for the hobbyist and industrial user alike.

For complete information on ordering, write to:
Western Data Systems
3650 Charles St, No. Z
Santa Clara CA 95050
atn: Cindy & Mike Indihar

Office: 408-984-7804
Home: 408-378-3569

The Introductory Offer expires August 31, 1976.

RCA COSMAC & μSCOPE

Dear Bob, 4/12/76

RCA has formally announced the 1802 chip for COSMAC, and it looks even better than the 1801. It seems strange that so few hobbyists are using COSMAC, since it was originally intended for the personal computer market (partly) and has a remarkably adaptable instruction set. Now that the new, improved version is available maybe some enterprising OEM will jump into the hobby market with a COSMAC-based machine. The RCA COSMAC Microkit (not to be confused with the RCA COSMAC Microtutor) is a beautifully engineered computer, but probably too expensive for most hobbyists. I don't know what the price tag is, but it doesn't look cheap (is it true that the jewels in the panel lamps are synthetic rubies?).

In the March-April, 1976 issue of *PCC* I predicted that the 1980 hobbyist would have a breadbox-size computer containing an integral ASCII keyboard, CRT display, tape cassette, hardcopy printer, and floppy disc. Well, it isn't quite 1980 but the newly-announced μScope 8000 (see the April 1 issue of *Electronics*) is a breadbox-size computer containing an integral ASCII keyboard, CRT display, tape cassette, harccopy printer, and a price of $6995. No floppy disc, but it does have a novel incremental assembler.

Tempus Digits,
Jim Day

LED REPLACEMENTS FOR BURN-OUTABLE PDP-8/E LAMPS

A conversion kit is available to enable replacement of standard incandescent lamps used in the PDP8/e minicomputer with light-emitting diodes, to eliminate the problem of burned-out bulbs. The kit is complete with a set of direct-replacement LED's and instructions for modification of the Front Panel Control Board circuitry. $39.95. Delivery, stock to 30 days. Scientific Test Systems, Box 741, Wallingford CT 06492; 203 265-5028

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

TV DAZZLER

SOFTWARE CONTEST

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

FIRST PRIZE:	$500	certificate for hardware from CROMEMCO
SECOND PRIZE:	$250	certificate for hardware from CROMEMCO

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

RULES:

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

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

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

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

ENTRIES MUST BE RECEIVED BY SEPT. 30, 1976