dr. dobb's journal of

$1.50

COMPUTER Calisthenics & Orthodontia

Running Light Without Overbyte

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

A REFERENCE JOURNAL FOR USERS OF HOME COMPUTERS

In This Issue. . .

DON'T KEEP IT A SECRET!

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

DR DOBB'S JOURNAL OF
COMPUTER CALISTHENICS & ORTHODONTIA

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

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

Editor
Jim C. Warren, Jr.

Contributing Editor
F. J. Greeb

Watchdogs
Bob Albrecht
Dennis Allison

Underdog
Rosehips Malloy

Circulation & Subscriptions
Mary Jo McPhee

Bulk Sales
Dan Rosset

Printer
Nowels Publications, Menlo Park 94025

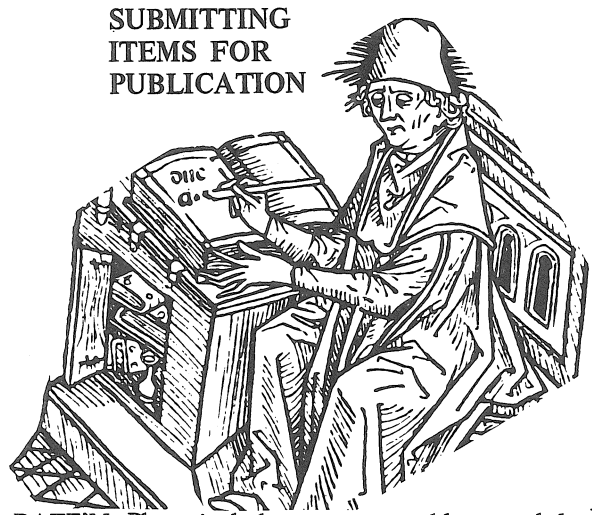
POSTMASTER: *Please send Form 3579 to: Box 310, Menlo Park CA 94025. Return postage guaranteed. Application to mail at second-class postage rates is pending at Menlo Park CA. Published 10 times per year; monthly, excluding July & December.*

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

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

Discounts available for bulk orders.

SUBMITTING ITEMS FOR PUBLICATION



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

TYPE'M—If at all possible, items should be typewritten, double-spaced, on standard, 8½ x 11 inch, white paper. If we can't read it; we can't publish it. Remember that we will be retyping all natural language (as opposed to computer languages) communications that we publish.

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

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

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

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

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

LETTERS FOR PUBLICATION—We are always interested in hearing your praise, complaints, opinions, daydreams, etc. In letters of opinion for publication, however, please back up any opinions that you present with as much factual information as possible.

We are quite interested in publishing well-founded, responsible evaluations and critiques of anything concerning hobbyist hardware or software, home computers, or computers and people.

We may withhold your name from a published letter if you so request. We will not publish correspondence, however, which is sent to us anonymously.

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

ADVERTISING—Advertising from manufacturers and vendors may be accepted by us. However, we reserve the right to refuse any advertising from companies which we feel fall short of our rather picky standards for ethical behavior and responsiveness to consumers. Also, any such commercial advertiser is herewith informed that we will not hesitate to publish harsh criticisms of their products or services, if we feel such criticisms are valid.

History repeats itself . . . I hope

Last December, I had the very good fortune to attend a lecture on the history of the electronic digital computer given by Professor Henry S. Tropp. Not too long ago, Dr. Tropp spent several years directing a research project for the Smithsonian Institute concerning the history of computers in the period, 1935 to 1955. Much of his work consisted of traveling around interviewing many of the "old Timers" in the fast-moving world of digital computers. Because the field is as new as it is—the first "real" computer was invented in the 1940's—many of the original researchers in computer science and technology are still alive. Dr. Tropp often spent several days talking with them.

His December lecture was a rambling collage of fascinating anecdotes and facts about the people and the technology, and insights into the intellectual character of the times. Some of his observations come to my mind, repeatedly, as I interact with computer hobbyists:

Dr. Tropp noted that, in those first years, there was a tremendous amount of intellectual and technical ferment in the area of computers. A number of prodigious steps were taken in a relatively short time, outlining and developing the majority of the concepts, theories, and principles that remain the cornerstones of computer science and computer engineering. A number of intellectual giants were involved: John Von Neumann, Alan Turing, George Stibitz, Howard Aiken, John Mauchly, and many others. As often as not, however, it appears that they had little idea of where they were going, and almost stumbled upon their great discoveries. The research community involved in these efforts was relatively small; most of the researchers knew one another, and there was often close communication among them. It seems like each new discovery by any one person quickly triggered renewed excitement and activity on the part of many others, even though they were scattered all over the northeastern United States, and some were in England. Each new discovery or development was quickly shared, and often served as a foundation element for someone else's discovery. This continued until the early 1950's. Then, this great rush of creation and advancement seemed to slow to a crawl. Some of the researchers went off to work for industry. A number of them moved to the West Coast to apply their research to the aircraft industry. Computers came out of the experimental labs, and became a profitable area of activity for business and industry. Particularly important: It appears that the easy communication and exchange of ideas that was so evident in the '40's was greatly curtailed in the '50's.

I see several parallels between what happened in those early days, and what is now occuring in the hobbyist community:

I see a great deal of excitement and obvious pleasure in the hobbyists, as they learn about these funny machines and discover their phenomenal capabilities. It seems to me that those early researchers must have felt much of the same excitement and intellectual stimulation, for computers were as new and novel to them, then, as they now are to many of the hobbyists.

I hear of hobbyists who are spending most of their time in a great flurry of activity and experimentation. In the same way, those first computer scientists often worked night and day on their research, sometimes moving bunks into their labs to facilitate their continuous efforts.

I note the extensive development of very inexpensive hardware within the hobbyist community, often developed in a basement shop. Similarly, the computer researchers of the '40's often worked on a shoestring budget in antiquated facilities.

Perhaps most significantly, I note the great willingness to share ideas, developments, facilities, and solutions to problems among the hobbyists. This obviously compares with the easy and open communication that was perhaps invaluable to those early researchers.

It is this open sharing that particularly delights me, and with which I am particularly concerned. I hope that it continues. We must all do whatever we can to encourage it. The sharing of ideas is useful in that it allows us to stand on one another's shoulders, instead of standing on one another's feet. But, there is something else being shared that is of at least equal value: the enthusiasm and intellectual excitement. There is no doubt in my mind that the sharing of such enthusiasm, as well as information, was a significant factor in the prodigious creativity of those original researchers. When one of them was frustrated and "down," probably someone came flying through the door, "wired" over some new discovery, and ricocheting off the walls . . . and the depressed co-worker wasn't "down" for very long. I believe the same holds true for the computer hobbyists. So . . . continue to share your ideas, and continue to share your excitement.

–Jim C. Warren, Jr., Editor

SCANNING THE INDUSTRY PERIODICALS

FROM THE MAY 24TH ISSUE OF *ELECTRONIC NEWS* (Fairchild Publications, Inc., 7 E 12 St., NYC 10003; (212) 741-4230)

DROP IN DEC ORDERS CAUSES LSI/11-CHIPMAKER MAJOR FINANCIAL PROBLEMS

Western Digital Corp., based in Newport Beach, Cal., manufactures the 16-bit microprocessor that is the basis for Digital Equipment Corporation's LSI/11. Western's attorney recently told its creditors that the company had a "cash flow position where it could not operate," due to drops in DEC orders (e.g., from $400K in March to less than $100K in April. A vice-president from DEC indicated that, although their LSI/11 sales had been going as expected, they could not afford to continue to buy in the same quantities they had in the past. Evidently, DEC had been purchasing more than they were using; they indicated that their inventory was filled to capacity. The creditors' committee recommended giving Western 30 days to rectify its financial position.

Editor's Note: If I remember correctly, Western agreed to sell their chip only to DEC. If DEC really wants to keep Western afloat, as appears to be the case, then DEC could try releasing Western from that exclusive sales agreement. My suspicion is that, if Western were free to peddle their PDP-11 look-alike microprocessor to everyone who wished to buy it, they could cure their financial problems rather quickly.

NATIONAL'S SC/MP KIT FOR $99

National Semiconductor has a microprocessor kit available for $99. This includes their SC/MP m-p (46 instruction types, single- and double-byte instructions, built-in serial I/O ports, bi-directional 8-bit bus, parallel data port, and latched, 12-bit address port), a preprogrammed 512-byte KITBUG ROM (monitor and debugger), 256 bytes of RAM, TTY interface (including buffer and driver for 20mA current-loop), voltage regulator, data buffer crystal (1.0 MHz), complete literature and schematics, and all the passive components and a PC board required to build a wee microcomputer. Contact Semiconductor Concepts: 145 Oser Ave., Hauppauge NY 11787, or 21201 Oxnard St., Woodland Hills CA 91364.

Any computer stores carrying SC/MP kit?

SEMICONDUCTOR SHIPMENTS EXPECTED TO NEAR $5 BILLION MARK, THIS YEAR

Western Electronics Manufacturers' Association (WEMA), has projected that worldwide semiconductor shipments will hit $4.98 billion in 1976. This is 24% above the $4.02 billion shipped in 1975. This report was a joint effort of all the major semiconductor manufacturers except Texas Instruments.

Other preductions contained in this report included: MOS circuits will byte into the bipolar circuit market, ECL families will see a slowing growth, and I-squared-L will see, at best, a questionable growth. P- and N-channel MOS is expected to increase from $728 million (in 1975) to $959 million. C/MOS is projected to jump from $96 million (1975) to $153 million.

MOTOROLA & AMD AGREE ON 2901 SECOND-SOURCE

Motorola Semiconductor and Advanced Micro Devices recently signed a licensing agreement for Motorola to second-source the bipolar, bit-sliced, 2901 microprocessor family developed by AMD. This makes Motorola the third second-source for 2901's. Others include Ratheon Semiconductor, and the Sescosem Division of the Paris-based Thomason-CSF.

FLOATING POINT SOFTWARE IN P/ROM's FOR 8080's

Recognition Systems in Van Nuys, Cal., is offering a floating point package in P/ROM's for $495. A number of routines are included: floating point multiply, divide, add, subtract, fixed-point to floating-point conversion, square root, and floating-point to binary-coded decimal conversion.

MONOLITHIC MEMORIES OFFERS MICROPROCESSOR-BASED NOVA COMPETITOR

Monolithic Memories, Inc. of Sunnyvale, Cal., recently announced a 16-bit microcomputer that is said to be both software and I/O compatible with Data General's NOVA 2 and 3. MMI expects to be producing about 50/month by mid-July. The systems will be priced at $2500 in quantities of 50 or more, including 32K words of memory. The systems use four of MMI's 6701 four-bit bipolar microprocessors. The 32K word memories use of 128 of MMI's 2180 4K RAM's.

There is also a rumor of a NOVA-on-a-chip in the foreseeable future.

8K PDP-8/E MEMORY FOR $650

WE Computer Extension Systems of Houston has announced a plug-in memory board for the PDP-8 Omnibus. The memory uses NMOS RAM's. A 4K version costs $400; the 8K version is $650. The company states that they come with an unconditional one-year warranty.

PASCAL COMPILER FOR PDP-11 EXTENDED FOR REAL-TIME FUNCTIONS

A compiler for PASCAL on the PDP-11 is being offered by Electro Scientific Industries of Portland, Ore., for $1500. PASCAL is an excellent, cleanly-designed, block-structured, high-level language, created about five years ago by Niklaus Wirth. ESI has extended the language to include the necessary constructs for real-time data acquisition and process control. The system runs under DEC's RT-11 operating system and requires 16K words.

First word on a floppy-disc operating system

Command language & facilities similar to DECSYSTEM-10

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

We have the first tidbits of information on the floppy-disc operating system to which we have alluded in past issues:

The system, called "CP/M," runs on an 8080. It is available from Digital Research, Box 579, Pacific Grove CA 93950; (408) 373-3403. Its user interface is patterned after that of the DECSYSTEM-10. The file commands include RENAME, TYPE, ERASE, DIRECTORY, LOAD, and auto-load/execute facility (type the name of an object file; it will be loaded and begin execution). File-names follow the DEC standard of a 1-8 character name with a 1-3 character suffix. An Editor is included that has somewhat the flavor of TECO. There is a PIP facility that allows easy transfer of files to and from any available device, e.g., terminal, paper-tape I/O, cassettes, floppy discs on any drive, etc.. Note: PIP is DECeze for *P*eripheral *I*nterchange *P*rogram. Other systems software is likely to be included.

Internally, the system allows for dynamic file allocation of files ranging from 0 to 250K bytes in length. It is initially set up to allow up to 63 different user-defined file-names (CP/M system files do not impinge on this file-name space), and can be simply modified to allow up to 255 such file-names. The system is written in PL/M. At an absolute and rather undesirable minimum, it will run on 8080's with only 8K bytes of memory. To be really usable, it requires 12K or, preferably, 16K bytes. The system includes an automatic bootstrap facility via a RESET of the controller, or allows a software bootstrap, if the controller doesn't have the RESET facility. The system is sufficiently modular that its designer feels he can easily modify it to operate on most floppy-disc drives and with most floppy-disc controllers.

This system *already exists and has been in use for over a year.* It was originally designed and implemented by Dr. Gary Kildall, a Computer Science Professor at the Naval Postgraduate School in Monterey, and a well-known, independent consultant in the area of mocroprocessor systems software. Gary is also the designer and implementor of PL/M, the "industry standard" high-level language for microprocessors. PL/M was produced for Intel for their 8000-family micros. Gary recently completed PLuS, a language for the Signetics 2650 microprocessor that is upwards-compatible with PL/M. Incidentally, since CP/M is written in PL/M, and PLuS is--for the most part--a superset of PL/M, Gary feels that it will be relatively simple for him to make this operating system also available for 2650-based systems.

It is expected that Digital Research will offer all of the parts for an inexpensive floppy-disc system, ready to build (kit), and/or plug-in (assembled) to an IMSAI or Altair microcomputer. These "parts" consist of: CP/M, the operating system; a floppy-disc controller; and floppy-disc drive(s). Pricing is still tentative, however these are the *conservative* estimates:

CP/M User's Manual + Editor User's Manual + CP/M Interface Manual $15
Extensive systems documentation package $35
Formatted, verified, "loaded" disc $20
Note: a "raw" disc costs about $8
Floppy-disc controller ..... $100-$350
Floppy-disc drives ..... $550-$650

We expect to carry more information, including firm prices, and one or several articles by Dr. Kildall in near-future issues.

From our experience, this is the hottest deal going! It's cheap, as far as floppy-disc systems for micros go. The software is well-designed, based on a well-known and easy-to-use operating system that has been around for a DECade. Additionally—major points worth considering—it has been in use for some time, it has been used by a number of people, and it is fairly completely debugged. We know Gary personally, know 'where his head's at," and know that he backs his products and is responsive to his customers. Incidentally, he has an excellent and ongoing working relationship with Digital Research.

You should very seriously consider obtaining a floppy disc subsystem—hardware and software— for your home computer (either Gary's or someone else's). A short-access mass-storage facility increases your capabilities by several orders of magnitude, particularly when it is backed by comprehensive, well-designed, debugged systems software. Note that the IBM 650 (the Model T of computers, and the one that moved computers out of the laboratory and into mass production and widespread usage) had a *main* memory that was a 2000-word rotating drum with access measured in milliseconds.

[The Dragon sez we should tell you that the 650 had 2000 *ten-digit* words. Think of it as about 10,000 bytes.]

A PLEA FOR *EXPLICIT* DIRECTIONS

Dear Jim Warren Jr., 4-27-76

My problem is interfacing (hooking all the components together so they work as a system). For instance, I can find *no where* instructions on *exactly* how to hook my SWTC keyboard and TVT-II up to my Altair through my 3P + S and 22" video monitor. I don't mean a few general instructions. I mean a wiring diagram showing *exactly* which lines go where so all the strobes, waits, readys, etc. work together. The same is true of my suding cassette interface. Then, if I order a tape from John Arnold what do I have to do to get it to work. I have no trouble building the parts and getting them to work, but nobody will furnish me with comprehensive instructions on putting them together into a *system.*

Sincerely,
John Greiner, Jr. 20002 S 57th
Temple TX 76501

SAN FRANCISCO BAY AREA STORE: COMPUTERS & STUFF of San Lorenzo

Computers & Stuff, a new retail computer outlet, has opened at 664 Via Alamo, San Lorenzo CA 94580; (415) 278-4720.
Its hours are: Wed. - Fri., 4 - 8 p.m.
Sat. - Sun., 1 - 7 p.m.

Hardware & software for speech synthesis

by Lloyd Rice Computalker Consultants
821 Pacific St., No. 4 Santa Monica CA 90405

The process of generating voice output with a computer can be broken down into several steps. We will examine the operations at each step to determine the flow of information into and out of the step. This examination will give us the background needed to decide which parts of the overall process should be wired into a hardware device, and which parts should be kept as software to retain flexibility and control over the process. Perhaps the easiest way to carry out such an examination is by following an example phrase thru the system as it is transformed into a speech signal and sent to the loudspeaker.

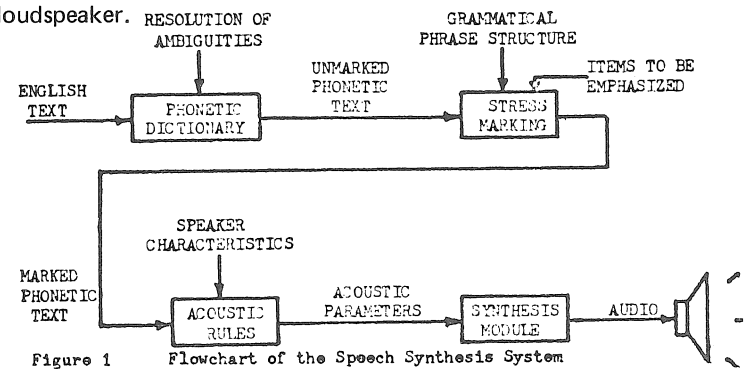


Figure 1 Flowchart of the Speech Synthesis System

Figure 1 shows a flowchart of the speech output system to be described. We will see that the kinds of external information needed for the first 2 steps is quite difficult to obtain and can require large amounts of processing, whereas the information needed in the third step is easily determined, and in most applications can be set as constants in the system. Finally, the acoustic parameters contain all the information necessary to control the last step, the actual synthesizer, to produce audio output. As a result of these observations, we will see that is most cases, one should specify the material to be synthesized in the form of marked phonetic text rather than raw English text. In order to present a more complete description, however, we will begin with the first step shown in Figure 1, input of English text.

Beginning with the sample text, "This is computer speech.", we first consult a phonetic dictionary, which performs a direct translation to phonetic text. A phonetic coding scheme suitable for this purpose which is compatible with the ASCII character set and Teletype output was developed by the Advanced Research Projects Agency (ARPA) as a part of a recent speech recognition study. That phonetic code, known as ARPABET, is listed in Table 1. The output of the phonetic dictionary in our example would be, "DHIHS/IHZ/KAHMPYUWTER/SPIYCH.#".

The main problem which arises at this stage is due to homographs, words that are spelled the same but pronounced differently. Two different types of homographs, however, present quite different problems. The first type consists mainly of short words such as "bow," pronounced either as in "tied a bow" and "bow and arrow," or as in "off the starboard bow." In these cases, the pronounciation can usually be resolved by examining the surrounding context. The other type of ambiguity is a lesser noticed but very widespread phenomenon in English: the situation where a word has a different stress pattern depending on whether it is used as a noun or a verb. As an example of this, notice the difference in "It was a dull subject," and "They were going to subject him to cruel punishment." It is not evident from the spelling which usage is intended, and requires that a fairly complete grammatical analysis be carried out to make that decision. One advantage at this point is that good use can be made of the recovered grammatical structure in the next step, where a more elaborate assignment of stress is performed.

The second step in the synthesis process deals with the assignment of sentence stress levels to the phonetic text string. To clarify that operation we will first have a closer look at the nature of the linguistic feature known as stress. The stress will be coded as a numerical value attached to a vowel in the phonetic string. That value will be realized later by the synthesizer in three different ways: as an increase in the pitch frequency, as a lengthening of the vowel duration, and to some extent as an increase in amplitude. The primary or highest level of stress is marked as a "1" following the vowel symbol. Secondary stress, marked with a "2," has less extreme acoustic effects than primary stress. As many as 3 or 4 distinct levels of stress may be marked in a sentence.

With regard to its communicative value, stress serves two quite distinct functions. The sentence stress pattern, together with timing and intonation, serves to communicate the grammatical structure to the listener. One can think of the grammatical structure as being transformed into a stress and intonation pattern by the speaker which is then decoded back into the phrase structure by the listener. Using the term "grammar," I am here including several kinds of information about words, such as the noun-verb distinction discussed above as well as syntactic information about the phrase and clause structure. The second function of the stress pattern is to indicate which item or items in a sentence are to be given special emphasis. The meaning of a sentence can be shifted around by emphasizing different items. The sources of information needed for marking these two components of the stress pattern are quite different and must be considered separately. Our example, with the stress pattern marked, would be something like, "DHIH3S/IHZ/KAHMPYUW1TER/SPIY2CH.#". Notice also that the word and utterance boundary markers have been kept explicitly in the text string.

The purpose of the portion of the system described thus far is imply to generate strings of phonetic text with marked stress patterns which are to be synthesized by the 2 steps in the bottom row of Figure 1. Marked phonetic text strings can be obtained in other ways, of course. In the case of predetermined phrases, marked phonetic strings can be stored instead of raw English text, making the synthesis task much simpler. On the other hand, consider synthesis of speech from an information network of some kind. The grammatical information could come from a phrase structure grammar which is being driven by relationships in the network. Items in the network would be coded as phonetic strings, or in essence, references to the phonetic dictionary described above. There are many significant problems remaining with this approach, but it is perhaps one of the more exciting applications of synthetic speech. The third box in the flowchart in Figure 1 is the acoustic rules section. In order to describe what the acoustic rules are and what they do, we must first look at the acoustic structure of speech. The speech code must be broken down into components so it can be synthesized by controlling, in real time, a limited number of parameter values. To a good

approximation, speech can be represented by the model shown in Figure 2.

This model requires 9 parameter control values consisting of 5 frequency controls and 4 amplitude controls. The box labeled "pulse source" is a controllable frequency oscillator which is adjusted dynamically to determine the voice pitch. The boxes labeled "resonator" are tunable, single-pole, bandpass resonators which determine the frequency or spectral shape of the speech signal in different ways. The data bus symbol used to represent the control inputs indicates that each parameter can be controlled by at most 8 bits from the computer's output bus. The data rates needed to control the

Table 1

COMPUTER PHONETIC REPRESENTATIONS

NOTE: Spaces are ignored except within escapes.

Phoneme	Computer Representation 1-Character	Computer Representation 2-Characters	Example	Phoneme	Computer Representation 1-Character	Computer Representation 2-Characters	Example
i	i	IY	beat	p	p	P	pet
I	I	IH	bit	t	t	T	ten
e	e	EY	bait	k	k	K	kit
ɛ	E	EH	bet	b	b	B	bet
æ	@	AE	bat	d	d	D	debt
ɑ	a	AA	Bob	g	g	G	get
ʌ	A	AH	but	h	h	HH	hat
ɔ	o	AO	bought	f	f	F	fat
o	o	OW	boat	θ	T	TH	thing
U	U	UH	book	s	s	S	sat
u	u	UW	boot	š or ʃ	S	SH	shut
ə	x	AX	about	v	v	V	vat
ɨ	X	IX	roses	ð	D	DH	that
ɝ	R	ER	bird	z	z	Z	zoo
aU or aw	W	AW	down	ž or ʒ	Z	ZH	azure
aI or ay	Y	AY	buy	č	C	CH	church
ɔI or ɔy	O	OY	boy	ǰ	j	JH	judge
y	y	Y	you	ʍ	H	WH	which
w	w	W	wit	syl l,ḷ	L	EL	battle
r	r	R	rent	syl m,ṃ	M	EM	bottom
l	l	L	let	syl n,ṇ	N	EN	button
m	m	M	met	flapped t,ɾ	F	DX	batter
n	n	N	net	glottal stop,ʔ	Q	Q	
ŋ	G	NX	sing	silence	-	-	
				non-speech segment	!	!	laugh, etc.

AUXILIARY SYMBOLS (1- AND 2-CHARACTER CODES ARE IDENTICAL)

Symbol	Meaning	Symbol	Meaning
+	Morpheme boundary	:3 or .	Fall-rise or non-term juncture
/	Word boundary	* **	Comment (anything except * or **)
#	Utterance boundary	' '	Apos-surround special symbol in comment
:	Tone group boundary		
:1 or .	Falling or decl. juncture	( )	Phoneme class information
:2 or ?	Rising or inter. juncture	< >	Phonetic or allphonic escape

STRESS REPRESENTATIONS (IF PRESENT, MUST IMMEDIATELY FOLLOW THE VOWEL)

Value	Stress Assignment	Value	Stress Assignment
o	No stress	3	Tertiary stress
1	Primary stress	.	(Etc.)
2	Secondary stress	:	

parameters are quite low, the highest rate needed for any parameter being less than 100 new settings per second.

I will not go into detail here describing the actual parameter values needed to represent particular speech sounds. An article to appear in the August, 1976 issue of *Byte Magazine* goes into some detail on the nature of the different kinds of speech sounds and how they can be generated by controlling the parameter values in such a model. Sucn information would, of course, be necessary to write a software implementation of the acoustic rules. For our present purposes, we consider the 9 control values as outlined above to represent an acoustic parameter model of speech. We can now turn to a discussion of the acoustic rules and the tasks they must perform to generate controls for this model.

Each phoneme, as encoded in the phonetic text string, is a symbol representing one or more acoustic speech segments, each such segment being produced by a particular pattern of values on the control parameters. Each pattern, or configuration of control values, must be held for a specific length of time before changing to the next pattern. As a first approximation then, the rules would consist of a series of table look-ups to convert each phoneme into a sequence of parameter patterns, along with the duration each pattern is to be held.

Now comes the catch! This first approximation makes rather poor speech. The problem is that the transitions between parameter values are often more important than the actual values at any given time. The flow of parameter values must be more carefully orchestrated. Actually, the only tough problem here is that correct transitions *between* phonemes are just as important as having the correct temporal structure *within* a phoneme. This means that phonemes cannot be coded as independent sets of parameter time functions which are merely joined together sequentially, but that some interaction must take place between phoneme patterns before they are sent out to the synthesizer module. Briefly, the different phonemes of a language can be classified according to the effects of boundary interactions. The transition of each parameter value across a given phoneme boundary can then be determined from the boundary characteristics of each of the neighborhing phonemes. Such boundary behavior information can be stored in phoneme look-up tables.

In addition to assigning initial parameter values and mapping the transitions across boundaries, the acoustic rules must also assign and modify durations. For example, a stressed vowel is given a longer duration than the same vowel in an unstressed position.

A third function the acoustic rules must perform—probably the most important for natural sounding speech—is to assign the time pattern of values to the pitch frequency parameter. First, an archetypal intonation pattern is chosen on the basis of punctuation (retained in the phonetic text just for this purpose). A period selects a falling pitch, a comma signals a level pause, and a question mark indicates a rising pattern. Other diacritical marks could be defined in the phonetic string to generate more complex pitch patterns such as singing. The selected archetypal pitch pattern is then modified locally by specific phonemes. Such local modification of the pitch pattern is one of the effects of a stress level marked on a vowel. Also, some consonants affect the pitch value slightly.

To complete the synthesis process, the acoustic parameters generated by the acoustic rules are output, in real time, to a synthesizer module such as sketched in Figure 2 and

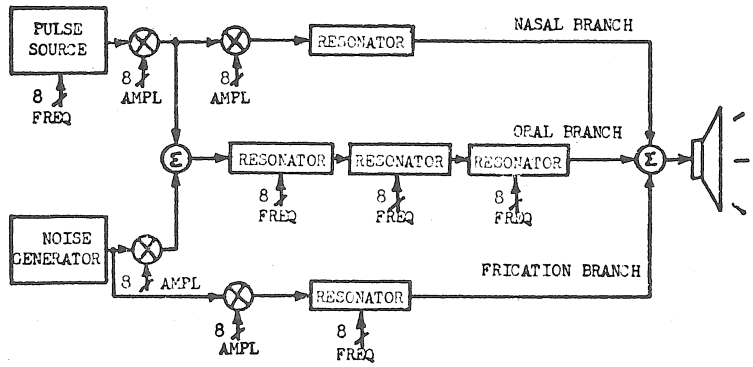


Figure 2 A Model of the Structure of Speech

described in the forthcoming *Byte* article. The synthesizer constructs an audio frequency signal as specified by the control parameters. The audio signal is then sent to a loudspeaker as the speech output.

It would be impractical to consider simulating the synthesizer module in software because of the speed needed to generate speech in real time. That task is much more appropriately handled by analog hardware. Such a hardware synthesizer module is currently being developed by Computalker, 821 Pacific St., No. 4, Santa Monica CA 90405. The Computalker synthesizer module would be driven by the microcomputer output data bus as described above. The software interface consists either of a direct, manually-controlled parameter pattern generator or an implementation of the acoustic rules. Software for the acoustic rules will also be developed by Computalker as the hardware becomes available.

I believe it is important to consider at this point some of the trade-offs involved in implementing the acoustic rules in software rather than hardware. A synthesizer system such as the Votrax VS-6 contains a hardware implementation of the basic acoustic rules. As a result, the language available for coding the phonetic text is fixed and cannot be extended. In addition, the phoneme table values are fixed so that each phoneme has a set phonetic quality. By implementing these rules in software you could retain the flexibility over pitch patterns and speech rate and also have control over the phonetic qualities which determine the language and dialect. The acoustic rules determine a number of qualities in the resulting speech which are characteristic of a particular speaker, such as the sex and age, and other qualities which vary from occasion to occasion, such as voice quality, speaking rate, distinctness of articulation, etc. Because of time constraints, a software version of the acoustic rules may not have time to handle all these possibilities as on-line variables. Of course, it is cheaper to produce a synthesizer module if a hardware acoustic rules system is not included.

How could speech output from a microcomputer be used? Several applications come to mind for the hobbyist environment, such as responses in games, voice readout of measurement data, system status warnings, etc., etc. Other applications might include telephone answering and intrusion warnings. What about generating audio tape labels automatically? Each of these applications makes its own demands for quality, naturalness and range of vocabulary needed. I would very much like to hear of your interest in computer speech output. What applications do you have in mind? What problems do you foresee? A note to the above address will assure that you receive further information as it becomes available.

MINOL—Tiny BASIC with Strings in 1.75K Bytes

AN OUTSTANDING JOB DONE BY A HIGH SCHOOL JUNIOR

Dear Mr. Warren: **May 1, 1976**

I have a Tiny BASIC program running on my Altair that I think you might be interested in. I call it MINOL (mine-all). It fits in 1.75K memory. Unlike the other Tiny BASIC's, MINOL has a string-handling capability, but only single-byte, integer arithmetic and left-to-right expression evaluation.

Additions to TB include CALL machine-language subroutines, multiple statements on a line (like TBX), and optional "LET" in variable assignments. Memory locations of the form (H,L) can be used interchangably with variables, permitting DIM-like operations.

Sincerely,

Erik T. Mueller **36 Homestead Lane**
Roosevelt NJ 08555

MINOL is an abbreviated form of BASIC with additional features. It has twelve statements: LET, PR, IN, GOTO, IF, CALL, END, NEW, RUN, CLEAR, LIST, and OS.

Variables: A letter from A to Z, or a memory location of the form (H,L), where H is the high address (decimal), and L is the low address. H and L may be expressions.

Number: An integer from 0 to 255.

Expression: A series of terms separated by arithmetic operators.

Terms: Numbers, variables, schars, random.

Schar: A single character enclosed in single quotes. Gives the ASCII value of the character.

Random: "!" (exclamation point) gives a random number between 0 and 255. (Subroutine by Jim Parker.)

Arithmetic Operators: + – * /

Relational Operators (not permitted in expressions): = # <("less than")

Arithmetic Evaluation: All expressions are evaluated from left to right (no precedence of operations).

Statements: A statement consists of one or more substatements separated by ":" (colon), and terminated by CR. Lines up to 72 characters. Line numbers from 1 to 254. All statements may be used with or without a line number. Statements without a line number are executed immediately. Statements with line numbers are edited into the existing program.

Substatements: [**LET** |ϕ] <var> = <expr> Assigns the value of a variable. The "LET" can be left out if desired.

```
Ex:  LET S = 0
     LET (24,0) = P-59
     A=B+C*J-198
     (25,5)=A*7/B
```

PR <var-list> [; |ϕ]

<var-list> : Literals, strings, or expressions separated by commas.

Literal: Characters to be printed enclosed in double quotes.

Strings: $(H,L): A series of memory locations starting at H,L which contain characters previously entered.

Expressions: Simple variable or expression.

```
Ex:  PR"YOU SAY YOUR NAME IS",$(10,0)
     PRA,B,(6,0),
     PR 56+!/A,B
     PR
```

A semicolon at the end of a PR suppresses CRLF. A blank PR produces a CRLF.

PR Format: Numerical values are printed with one leading and trailing space and with all leading zeros suppressed. All strings and literals are printed without leading and trailing spaces. No zone spacing.

GOTO<expr>

Transfers control to the specified statement. GOTO 0 transfers control to beginning of unnumbered statement.

```
Ex:  GOTO A*10
     GOTO 78
```

IF<expr> <relop> <expr>;<statement>

Executes the statement following the ";" (semi-colon) if the specified relation is true. If it is untrue, control is transferred to the next statement on the line (if present).

```
Ex:  IF X=5 ; GOTO 20
     IF A='Y' ;PR"SURE, WHY NOT?"
     IF A+B*C # !;GOTO 20 : PRA+B*C
     IF Y # 6; S=!
```

IN [<var>|<str>] [,[<var>|<str>]]*

This statement permits two types of data to be entered from the terminal: a) Numeric data; and
b) Alphanumeric data; either a single letter, or a string of n characters.

Using a <var>: The input data is tested. If it is numeric, the number is deposited into the variable. If the data is not a number, the ASCII value of the first character typed is deposited.

Using a <str>: (of the form $(H,L) The inputted characters are deposited into memory sequentially starting at location H,L. 255 is placed in memory after the last character before CR. All spaces inputted are ignored unless enclosed by quotes. Note that (H,L) refers to a single location, but $(H,L) refers to a series of locations beginning at H,L. (H,L) can be used in expressions as a variable, but $(H,L) can only be used in I/O statements (IN, PR).

CALL (H,L)

Calls users subroutine starting at location H,L decimal.

END: Terminates program.

NEW: Deletes all lines of a program.

CLEAR: Sets all variables (A-Z) equal to zero.

RUN: Starts execution of program at lowest numbered statement.

LIST: Lists program in memory.

OS: Transfers control to user's operating system.

Line editing and correction:

Typing X^S deletes the last character typed.

X^L deletes an entire line.

X^C stops executing program.

Prints: BREAK AT LL (LL is the line that was to be executed before the interrupt occurred.)

To delete a line, type the line number followed by CR.

To change a line, type in the line with changes. The new line will replace the old one.

ERROR MESSAGES !ERR L AT XX

1. Label does not exist

2. Input is over 72 characters.
3. Unrecognizable statement type.
4. Illegal variable.
5. Syntax error.
6. Out of memory.

EM MINOL 2.1 SYNTAX Apr. 1976

```
<line>          ::=<number> <statement> cr |<statement>cr
<statement>     ::=<substatement>* : <substatement>
<substatement>  ::=[LET |ϕ] <var> = <expr>
                   PR <expr-list> [;|ϕ]
                   IN <var-str-list>
                   IF <expr><relop><expr> ; <statement>
                   GOTO <expr>
                   CALL <memloc>
                   END
                   RUN
                   LIST
                   NEW
                   CLEAR
                   OS
<number>        ::=<digit>*2 <digit>
<digit>         ::=0|1|. . |8|9
<var>           ::=A|B|. . |Y|Z|<memloc>
<relop>         ::=#|=|<
<expr-list>     ::=[<literal>|<expr>|<str>] [,[<literal>|
                   <expr>|<str>] ]*
<var-str-list>  ::=[<var>|<str>] [;[<var>|<str>] ]*
<expr>          ::=[<term><aroper>] * <term>
<term>          ::=<var>|<number>| '<schar>' |!
<literal>       ::="<char>*"
<schar>         ::=<char>
<str>           ::=$ <memloc>
<memloc>        ::=(<highadr>,<lowadr>)
<highadr>       ::=<number>
<lowadr>        ::=<number>
<char>          ::=any character except " and cr
```

Notes: <>encloses an element of MINOL
ϕ is the empty set
* repeat limited by length of line
*2 repeat from 0 to 2 times

MINOL

Memory Allocation:
(All locations are split octal)

000 000 - 000 115 I/O Routines, etc.

System Reset:	000	000	061 LXI SP
		001	377
		002	017
		003	317 RST CRLF
		004	303 JMP, MINOL
		005	116
		006	000
CRLF:		010	A subroutine to output a CR followed by a LF.
INPUT:		020	Moves a character from input device to the A register. Parity equals 1. Must output an echo check of the inputted character.
OUTPUT:		040	Outputs character in the A register. Parity equals 1. No registers may be altered.
Must CALL, INT		315	
		363	
		002	

This checks for keyboard interrupt (X^C).

000 116 - 006 377 MINOL Interpreter

000 253 L		Highest memory location available
261 H		for MINOL programs.
001 142 L		Address of user's operating system,
143 H		or monitor.

All input text is stored at 006 210 - 006 320
Free speace is left for short strings at 006 333 - 006 377
Variables (A-Z) are stored at 005 007 - 005 042

007 000 + Program storage

Executing MINOL:
To start MINOL and initialize program area, EXAMINE 002 350, RUN.
To start without initialization, EXAMINE 000000, RUN.

Dear Jim: **May 24, 1976**

I am enclosing the listing of MINOL—manually typed!

There are several features of my program, both positive and negative, that I might point out.

On the plus side, MINOL uses only 1.75K of memory, including the input-output subroutines (although since writing it I see how I can make it even smaller.) Memory locations of the form (H,L) can be used similarly to one- or two-dimensional DIMs in higher BASIC's. Simple input or output strings are possible by specifying a series of memory locations—of the form $(H,L) where H,L is the first location where characters are to be deposited. I am enclosing three programs to illustrate these features. On the negative side, the program is not designed for arithmetic functions, having no grouping of operations, and being limited to a value of 255. The relational operators are restricted to =, #, and <, although > ("greater than") can be done by reversing the logical expressions. Fewer error messages are provided than usual. MINOL is *written completely in machine language* without using IL.

When I can supply MINOL on a cassette I'll let you know. You might like to know that I am in my third year of high school.

Yours truly,

Erik T. Mueller
Britton House
Roosevelt NJ 08555

Additions/changes since the May 1st letter:

Spaces are ignored:
a. During line/statement entry unless enclosed by quotes.
b. When inputting variables.
c. When inputting strings if the L address is zero.

Spaces are accepted:
a. When inputting strings if the L address is non-zero.
b. When enclosed by quotes.

Instead of GOSUB/RET statements, use the following substitute statements to perform the same function:

First initialize the GOSUB stack pointer Y,Z:
2 Y=14:Z=255 (Y and Z are the H,L address of some free space in memory.)

Instead of a GOSUB statement, substitute the

following: LET(Y,Z)=<Return label> :Z=Z-1:GOTO <subroutine label>

Instead of a RET, substitute: Z=Z+1:GOTO(Y,Z)

Free space is left for very short user's strings from 006366 to 006377.

On a directly-executed IN statement, although the data will be correctly stored, an error message may appear after its execution.

The monitor gives a "]" as a prompt. The IN statement gives a "?" unless a sense switch is up.

Three programs in MINOL:

```
]LIST
1Ø PR"GIVE ME A SENTENCE":IN$(14,1)
2Ø PR"STRING TO SEARCH FOR?":IN$(14,101)
21 A=Ø
22 A=A+1:IF(14,A)#255;GOTO22
23 B=Ø
24 B=B+1:IF(14,100+B)#255;GOTO24
3Ø C=1:D=1:S=Ø
4Ø IF(14,D+1ØØ)#(14,C);GOTO7Ø
5Ø D=D+1:C=C+1:IFD<B;GOTO4Ø
6Ø LETS=S+1
65 C=C-1
7Ø LETD=1
8Ø C=C+1:IFC<A;GOTO4Ø
9Ø PR"'";$(14,101);"' OCCURS";S;
96 IFS=1;GOTO1ØØ
97 PR"TIMES IN '";$(14,1);"'":END
100 PR"TIME IN '";$(14,1);"'":END
]RUN
GIVE ME A SENTENCE
? THE BLUE BIRD IN THE BLUE SKY
STRING TO SEARCH FOR?
? BLUE
'BLUE' OCCURS 2 TIMES IN 'THE BLUE BIRD IN THE BLUE SKY'
```

```
]LIST
1Ø "***NUMBER-A NUMBER GUESSING GAME (NUMØ5)
2Ø PR:PR"WHAT IS YOUR NAME";:IN$(14,1)
3Ø X=!:S=Ø:PR"HI,";$(14,1);". WELCOME TO THE GAME OF NUMBER"
4Ø PR"I'M THINKING OF A NUMBER FROM ø TO 255"
5Ø PR"GUESS MY NUMBER!!"
6Ø PR:PR"YOUR GUESS";:ING:S=S+1
65 IFG=X;GOTO90
7Ø IFG<X;PR"TOO SMALL. TRY A BIGGER NUMBER."
8Ø IFX<G;PR"TOO BIG. TRY A SMALLER NUMBER."
85 GOTO6Ø
9Ø PR"THAT'S RIGHT,";$(14,1);"!! YOU GOT IT IN";S;"GUESSES"
1ØØ PR"PLAY AGAIN";:INA:IFA='Y';GOTO3Ø
11Ø PR"OK.....HOPE YOU HAD FUN.":END
```

```
1Ø PR"NAME";:IN$(14,1)
2Ø IF(14,1)='J';IF(14,2)='I';IF(14,3)='M';PR"IT'S JIM!"
3Ø IF(14,1)#'J';PR"IT'S NOT JIM.":GOTO1Ø

]RUN
NAME?ERIK
IT'S NOT JIM.
NAME?JIM
IT'S JIM!
NAME?X^C
BREAK AT 1Ø
]
```

MINOL 2.1 Erik T. Mueller May 1976

TAG	ADDRESS	I1	I2	I3	MNEMONIC	COMMENTS
MINOL	000116	076	335		MVI A "]"	Output prompt
	000120	347			RST OUT	
	000121	315	052	004	CAL,INPTXT	Get input line
	000124	041	210	006	LXI HL:TXT	Point to input text with
	000127	176			MOV A,M	HL
	000130	315	062	005	CAL,CHEKN	Check for label
	000133	322	000	001	JNC,DIRECT	If no label, go execute command
FND	000136	043			INX HL	Point to first non-numeric
	000137	176			MOV A,N	character
	000140	315	062	005	CAL,CHEKN	
	000143	332	136	000	JC,FND	
	000146	315	256	004	CAL,MKBIN	Convert ASCII label to binary
	000151	021	000	007	LXI DE:PROG	This section edits (inserts,
ZIP	000154	032			LDAX DE	deletes, changes) lines of
	000155	376	215		CPI "CR"	the program
	000157	023			INX DE	
	000160	302	154	000	JNZ,ZIP	
	000163	032			LDAX DE	Look at line number
	000164	376	377		CPI 377	
	000166	312	177	000	JZ,INSRT	
	000171	067			STC	
	000172	077			CMC	
	000173	270			CMP B	
	000174	332	154	000	JC,ZIP	Point to line number greater
INSRT	000177	176			MOV A,M	than or equal to entered
	000200	376	215		CPI "CR"	label
	000202	312	345	000	JZ,EKIL	If label alone, delete line
	000205	016	002		MVI C 002	
IHR	000207	043			INX HL	Count length of line and add 2
	000210	176			MOV A,M	
	000211	014			INR C	
	000212	376	215		CPI "CR"	
	000214	302	207	000	JNZ,IHR	
OHIO	000217	032			LDAX DE	If line entered already exists,
	000220	270			CMP B	first delete the old one,
	000221	302	231	000	JNZ,IBYH	then insert the new one
PZIY	000224	325			PUSH DE	
	000225	315	351	000	CAL,KILLINE	Delete old line
	000230	321			POP DE	
IBYH	000231	142			MOV H,D	HL points to first location
	000232	153			MOV L,E	where new line will be placed
	000233	325			PUSH DE	Save position in stack
EHR	000234	023			INX DE	
	000235	032			LDAX DE	
	000236	376	377		CPI 377	
	000240	302	234	000	JNZ,EHR	Continue until DE points to
	000243	171			MOV A,C	end of file
	000244	102			MOV B,D	Length of new line in A
	000245	113			MOV C,E	
HBY	000246	023			INX DE	
	000247	043			INX HL	
	000250	365			PUSH PSW	
	000251	173			MOV A,E	
	000252	376	xxx		CPI xxx	xxx = Low address: limit of
	000254	302	265	000	JNZ,HII	program memory
	000257	172			MOV A,D	
	000260	376	xxx		CPI xxx	xxx = High address:memory limit
	000262	312	224	004	JZ,ERR6	Out of memory error
HII	000265	361			POP PSW	
	000266	075			DEC A	
	000267	302	246	000	JNZ,HBY	Increment until DE points to
UPDT	000272	012			LDAX BC	new end-of-file position,
	000273	022			STAX DE	and HL points to where file
	000274	173			MOV A,E	updating begins
	000275	275			CMP L	BC points to end of file
	000276	302	306	000	JNZ,NHR	
	000301	172			MOV A,D	
	000302	274			CMP H	
	000303	312	313	000	JZ,NET	
NHR	000306	013			DCX BC	
	000307	033			DCX DE	
	000310	303	272	000	JMP,UPDT	Relocate file leaving space for
NET	000313	321			POP DE	new line. Retrieve pointer
STR	000314	041	210	006	LXI HL:TXT	
IFD	000317	043			INX HL	
	000320	176			MOV A,M	
	000321	315	062	005	CAL,CHEKN	
	000324	332	317	000	JC,IFD	Point to first non-numeric
	000327	072	120	003	LDA:BIN	character.Put line no. in A
	000332	022			STAX DE	Store line no. in file
NTAT	000333	023			INX DE	Store line text in file
	000334	176			MOV A,M	
	000335	022			STAX DE	
	000336	043			INX HL	
	000337	376	215		CPI "CR"	
	000341	302	333	000	JNZ,NTAT	
	000344	307			RST RESET	Go back to monitor section
EKIL	000345	315	351	000	CAL,KILLINE	Delete line
	000350	307			RST RESET	
KILLINE	000351	032			LDAX DE	Delete a line
	000352	270			CMP B	
	000353	300			RNZ	If deleting line that does not
	000354	142			MOV H,D	exist, return
	000355	153			MOV L,E	
BBL	000356	043			INX HL	
	000357	176			MOV A,M	
	000360	376	215		CPI "CR"	
	000362	302	356	000	JNZ,BBL	Point to next line
ARK	000365	043			INX HL	Relocate file, deleting line
	000366	176			MOV A,M	
	000367	022			STAX DE	
	000370	376	377		CPI 377	
	000372	310			RZ	
	000373	023			INX DE	
	000374	303	365	000	JMP,ARK	
	000377	000			NOP	
DIRECT	001000	317			RST CRLF	Direct execution of a statement
	001001	257			XRA A	
	001002	062	121	003	STA:LNE	Set LNE (current line no.)=0
	001005	303	041	001	JMP,EXEC	Execute statement
RUN	001010	041	000	007	LXI HL:PROG	RUN Statement: Start from be-
LPUB	001013	176			MOV A,M	ginning of program. Get next
	001014	376	272		CPI ":"	statement.
	001016	043			INX HL	
	001017	312	041	001	JZ,EXEC	If not a new line, go execute
	001022	376	215		CPI "CR"	statement
	001024	302	013	001	JNZ,LPUB	
BIB	001027	176			MOV A,M	
	001030	376	377		CPI 377	
	001032	312	000	000	JZ,RESET	If statement no.=377 (end of
	001035	062	121	003	STA:LNE	program), go back to monitor
	001040	043			INX HL	If not 377, store current line
EXEC	001041	315	363	002	CAL,INT	no. at LNE
	001044	043			INX HL	Check for keyboard program
	001045	176			MOV A,M	interrupt
	001046	376	275		CPI "="	Check for "=" in second column
	001050	312	221	001	JZ,LET	(variable assignment)
	001053	053			DCX HL	
	001054	176			MOV A,M	
	001055	376	250		CPI "("	If "(" in first column (memory
	001057	312	221	001	JZ,LET	location assignment)go to LET
	001062	376	303		CPI "C"	Check for "C"
	001064	302	106	001	JNZ,GSM	If not, go on
	001067	043			INX HL	
	001070	176			MOV A,M	
	001071	376	301		CPI "A"	
	001073	312	323	002	JZ,CALL	CALL Statement
	001076	376	314		CPI "L"	
	001100	312	304	002	JZ,CLR	CLEAR Statement
	001103	303	205	004	JMP,ERR3	If neither, report error

```
        001106  376 305        CPI "E"        Check for "E" as in END.
        001110  312 000 000    JZ,RESET
        001113  376 307        CPI "G"
        001115  312 007 003    JZ,GOTO
        001120  376 242        CPI '"'        Check for " indicating
        001122  312 013 001    JZ,LPUB          REM statement
        001125  376 316        CPI "N"
        001127  312 350 002    JZ,NEW
        001132  376 320        CPI "P"
        001134  312 327 001    JZ,PR
        001137  376 317        CPI "O"
        001141  312 LLL hhh    JZ,OS          Address of user's monitor
        001144  376 322        CPI "R"
        001146  312 010 001    JZ,RUN
        001151  376 311        CPI "I"
        001153  302 175 001    JNZ,LS
        001156  043            INX HL
        001157  176            MOV A,M
        001160  376 316        CPI "N"
        001162  312 116 002    JZ,IN
        001165  376 306        CPI "F"
        001167  312 324 005    JZ,IF
        001172  303 205 004    JMP,ERR3
LS      001175  376 314        CPI "L"
        001177  302 205 004    JNZ,ERR3
        001202  043            INX HL
        001203  176            MOV A,M
        001204  376 305        CPI "E"
        001206  312 221 001    JZ,LET
        001211  376 311        CPI "I"
        001213  312 132 005    JZ,LIST
        001216  303 205 004    JMP,ERR3
LET     001221  176            MOV A,M        LET Statement executer
        001222  315 043 005    CAL,TERM       Find "="
        001225  332 217 004    JC,ERR5        Report error if no "=" before
        001230  376 275        CPI "="          CR or ":"
        001232  043            INX HL
        001233  302 221 001    JNZ,LET
        001236  303 321 006    JMP,FIX        Transfer expression text in
        001241  000 000        NOP              program text, to expression
        001243  043            INX HL           buffer
        001244  023            INX DE
        001245  322 324 006    JNC,MREN
        001250  315 162 003    CAL,EXPR
SERCH   001253  176            MOV A,M        Go back before "="
        001254  376 275        CPI "="
        001256  053            DCX HL
        001257  302 253 001    JNZ,SERCH
        001262  176            MOV A,M
        001263  315 371 004    CAL,CHEKLTR
        001266  322 301 001    JNC,INLET      If not variable, get memory
        001271  315 343 004    CAL,GETADR       address
        001274  171            MOV A,C
        001275  022            STAX DE        Store in variable
        001276  303 013 001    JMP,LPUB       Next statement
INLET   001301  376 251        CPI ")"
        001303  302 212 004    JNZ,ERR4
JHR     001306  053            DCX HL
        001307  176            MOV A,M
        001310  376 250        CPI "("
        001312  302 306 001    JNZ,JHR
        001315  171            MOV A,C
        001316  365            PUSH PSW
        001317  315 062 006    CAL,VAL        Get memory location in BC
        001322  361            POP PSW
        001323  002            STAX BC
        001324  303 013 001    JMP,LPUB
PR      001327  043            INX HL         PR Statement executer
        001330  043            INX HL         Skip assumed characters
        001331  176            MOV A,M
        001332  315 043 005    CAL,TERM       If blank print, go to CR
        001335  332 016 002    JC,DCR
NXTE    001340  376 242        CPI '"'        Check for literal
        001342  302 022 002    JNZ,VAR        If not, go on
HR      001345  043            INX HL         Print text until " found
        001346  176            MOV A,M
        001347  376 342        CPI '"'
        001351  312 366 001    JZ,MRENO
        001354  315 043 005    CAL,TERM       If terminator before closing
        001357  332 217 004    JC,ERR5          quotes, print error
        001362  347            RST OUT
        001363  303 345 001    JMP,HR
MRENO   001366  043            INX HL
        001367  176            MOV A,M
        001370  315 043 005    CALL,TERM      If end of statement without
        001373  332 016 002    JC,DCR           semicolon ";",go do CR
        001376  376 273        CPI ";"
        002000  302 217 004    JNZ,ERR5
        002003  043            INX HL
        002004  176            MOV A,M
        002005  315 043 005    CAL,TERM       If term after semicolon do not
        002010  332 017 002    JC,NCR           print CR
        002013  303 340 001    JMP,NXTE       Get next thing to print
DCR     002016  317            RST CRLF
NCR     002017  303 013 001    JMP,LPUB
VAR     002022  376 244        CPI "$"
        002024  312 076 002    JZ,STR         Check if string
        002027  021 151 006    LXI DE:EXP
        002032  076 240        MVI A "SP"     Output leading space
        002034  347            RST OUT
ER      002035  176            MOV A,M        Transfer expression text from
        002036  022            STAX DE          program text to expression
        002037  043            INX HL           buffer
        002040  023            INX DE
        002041  315 043 005    CAL,TERM
        002044  332 054 002    JC,HR
        002047  376 274        CPI ";"
        002051  302 035 002    JNZ,ER
HR      002054  053            DCX HL
        002055  033            DCX DE
        002056  076 215        MVI A "CR"
        002060  022            STAX DE
        002061  315 162 003    CAL,EXPR
        002064  101            MOV B,C
        002065  315 174 005    CAL,PBINBCD    Print expression's value
        002070  076 240        MVI A "SP"
        002072  347            RST OUT
        002073  303 367 001    JMP,MRENO+1
STR     002076  043            INX HL
        002077  315 062 006    CAL,VAL        Get start address of string
MRE     002102  012            LDAX BC          in BC. Print string
        002103  347            RST OUT
        002104  376            CPI
        002105  377            377
        002106  003            INX BC
        002107  302 102 002    JNZ,MRE
        002112  043            INX HL
        002113  303 367 001    JMP,MREN
IN      002116  043            INX HL         Input statement
        002117  333            INP
        002120  377            377
        002121  376            CPI            If sense switches down,print"?"
        002122  000            000
        002123  302 134 002    JNZ,EAHR
        002126  076 277        MVIA "?"
        002130  347            RST OUT
        002131  076 240        MVIA "SP"
        002133  347            RST OUT
EAHR    002134  176            MOV A,M
        002135  315 371 004    CAL,CHECKLET   Check for variable
        002140  332 164 002    JC,LVB
        002143  376 244        CPI "$"        Check for input string
        002145  312 247 002    JZ,STRIN
        002150  376 250        CPI "("        Check for single memory
```

```
        002152  302 212 004  JNZ,ERR4        location
        002155  315 150 003  CAL,VALDE       Get location in DE
        002160  345          PUSH HL
        002161  303 170 002  JMP,HS
LVB     002164  345          PUSH HL
        002165  315 343 004  CAL,GETADR      Get address of letter variable.
HS      002170  325          PUSH DE
IHERE   002171  315 052 004  CAL,INPTXT      Input a line
        002174  317          RST CRLF
        002175  041 210 006  LXI HL:TXT
        002200  176          MOV A,M
        002201  315 062 005  CAL,CHEKN       Check for number
        002204  322 224 002  JNC,LETR
FD      002207  043          INX HL
        002210  176          MOV A,M
        002211  315 062 005  CAL,CHEKN
        002214  332 207 002  JC,FD           Point to first non-numeric
        002217  305          PUSH BC           character
        002220  315 256 004  CAL,MKBIN       Convert ASCII input data to
        002223  301          POP BC            binary
LETR    002224  321          POP DE
        002225  022          STAX DE         Put A in variable
CHK     002226  341          POP HL
        002227  043          INX HL
        002230  176          MOV A,M
        002231  376 254      CPI ","         Check for more input variables
        002233  312 116 002  JZ,IN
        002236  315 043 005  CAL,TERM
        002241  332 013 001  JC,LPUB
        002244  303 217 004  JMP,ERR5
STRIN   002247  043          INX HL          Input string
        002250  315 062 006  CAL,VAL         Get first memory location in BC
        002253  345          PUSH HL
        002254  315 360 006  CAL,DMS         Input a line
        002257  317          RST CRLF
        002260  041 210 006  LXI HL:TXT
LD      002263  176          MOV A,M         Store text beginning at spec-
        002264  376 215      CPI "CR"          ified location
        002266  312 276 002  JZ,TE
        002271  002          STAX BC
        002272  003          INX BC
        002273  303 354 006  JMP,CIN
JE      002276  076 377      MVI A 377       Store 377 at end of string
        002300  002          STAX BC
        002301  303 226 002  JMP,CHK
CLR     002304  021 007 005  LXI DE          CLEAR executer.Var storage
LCR     002307  257          XRA A
        002310  022          STAX DE
        002311  023          INX DE
        002312  173          MOV A,E
        002313  376 042      CPI 042         Last variable location
        002315  302 307 002  JNZ,LCR
        002320  303 013 001  JMP,LPUB
CALL    002323  000          NOP             Skip assumed characters
        002324  043          INX HL
        002325  043          INX HL
        002326  043          INX HL
        002327  315 062 006  CAL,VAL         Get address in BE
        002332  345          PUSH HL
        002333  325          PUSH DE
        002334  021 343 002  LXI DE:RET      Load DE with return address
        002337  325          PUSH DE         Push return address into stack
        002340  140          MOV H,B
        002341  151          MOV L,C
        002342  351          PCHL            Jump to user's subroutine
RET     002343  321          POP DE
        002344  341          POP HL
        002345  303          JMP,LPUB
NEW     002350  021 000 007  LXI DE:PROG     New
        002353  076 215      MVI A "CR"      Initialize Program area
        002355  022          STAX DE
        002356  023          INX DE
```

```
        002357  076 377      MVI A
        002361  022          STAX DE
        002362  307          RST RESET
INT     002363  365          PUSH PSW        This subroutine checks for
        002364  333 013      INP 013          X^C from keyboard
        002366  376 203      CPI "X^C"
        002370  312 375 002  JZ,BREAK
        002373  361          POP PSW
        002374  311          RET
BREAK   002375  317          RST CRLF
        002376  021 112 003  LXI DE          "BREAK"
        003001  315 360 004  CAL,PRINTXT
        003004  303 240 004  JMP,AT+
GOTO    003007  043          INX HL          GOTO executer
        003010  043          INX HL          Skip assumed characters
        003011  043          INX HL
        003012  043          INX HL
        003013  021 151 006  LXI DE:EXPR     Calculate value of expression
RME     003016  176          MOV A,M
        003017  022          STAX DE
        003020  315 043 005  CAL,TERM
        003023  043          INX HL
        003024  023          INX DE
        003025  322 016 003  JNC,RME
        003030  315 162 003  CAL,EXPRS
        003033  101          MOV B,C
        003034  171          MOV A,C
        003035  376 000      CPI
        003037  302 050 003  JNZ,JUMP
        003042  041 210 005  LXI HL:TXT
        003045  303 000 001  JMP,DIRECT
JUMP    003050  041 000 007  LXI:PROG HL     Goto expression
DUP     003053  176          MOV A,M
        003054  376 215      CPI "CR"
        003056  043          INX HL
        003057  302 053 003  JNZ,DUP
        003062  176          MOV A,M
        003063  376 377      CPI 377
        003065  312 173 004  JZ,ERR1
        003070  270          CMP B
        003071  302 053 003  JNZ,DUP
        003074  303 027 001  JMP,BIB
!ERR    003077  241 305 322  DW "!ER"        Error & Break Texts
        003102  322 240      DW "R "
        003104  377          DB 377
AT      003105  240 301 324  DW " AT"
        003110  240          DB " "
        003111  377          DB 377
BREAK   003112  302 322 305  DW "BRE"
        003115  301 313      DW "AK"
        003117  377          DB 377
BIN     003120  000
LNE     003121  000                          Current line no. storage
        003122  000          NOP
ACT     003124  305          PUSH BC         Gets the value of a memory
        003124  353          XCHG              location
        003125  315 062 006  CAL,VAL         Part of the EXPRS subroutine
        003130  353          XCHG              to follow
        003131  012          LDAX BC
        003132  301          POP BC
        003133  107          MOV B,A
        003134  023          INX DE
        003135  303 247 003  JMP,GETNET
NOTEQ   003140  173          MOV A,E         If two expressions are not
        003141  271          CMP C             equal, execute statement
        003142  302 041 001  JNZ,EXEC          (Part of IF statement exe-
        003145  303 013 001  JMP,LPUB          cutor to follow)
VALDE   003150  315 062 006  CAL,VAL
        003153  120          MOV D,B
        003154  131          MOV E,C
        003155  311          RET
        003156  000 000 000  NOP
```

```
        003161  000          NOP
EXPRS   003162  345          PUSH HL          Calculates the value of an
        003163  021 150 006  LXI DE:EXPR-1      expression stored in memory.
        003166  016          MVI C              DE is the cursor
        003167  000          000
RETPT   003170  032          LDAX DE
        003171  315 043 005  CAL,TERM         Check for end of expression
        003174  322 201 003  JNC,GOMOR
        003177  341          POP HL
        003200  311          RET
GOMOR   003201  365          PUSH PSW         Save Operation in stack.
        003202  023          INX DE           Get Term/Factor
        003203  032          LDAX DE
        003204  376 247      CPI "'"          Single character value
        003206  312 271 003  JZ,ASC
        003211  376 250      CPI "C"          Memory location
        003213  312 123 003  JZ,ACT
        003216  376 241      CPI "!"          Random number
        003220  312 000 004  JZ,RND
        003223  315 062 005  CAL,CHEKN
        003226  332 350 003  JC,CONSTANT      Constant (number)
        003231  315 371 004  CAL,CHEKL
        003234  322 217 004  JNC,ERR5
        003237  023          INX DE
IVAR    003240  325          PUSH DE          Variable
        003241  315 343 004  CAL,GETADR
        003244  032          LDAX DE
        003245  107          MOV B,A
        003246  321          POP DE
GETNET  003247  361          POP PSW          Retrieve Operation
        003250  376 253      CPI "+"
        003252  312 277 003  JZ,ADD
        003255  376 255      CPI "-"
        003257  312 305 003  JZ,SUB
        003262  376 252      CPI "*"
        003264  312 313 003  JZ,MULT
        003267  376 257      CPI "/"
        003271  312 326 003  JZ,DIV
        003274  303 217 004  JMP,ERR5
ADD     003277  171          MOV A,C          Add C=C+B
        003300  200          ADD B
        003301  117          MOV C,A
        003302  303 170 003  JMP,RETPT
SUB     003305  171          MOV A,C          C=C-B
        003306  220          SUB B
        003307  117          MOV C,A
        003310  303 170 003  JMP,RETPT
MULT    003313  171          MOV A,C          C=C*B
        003314  005          DCR B
GBK     003315  201          ADD C
        003316  005          DCR B
        003317  320          JNZ,GBK
        003322  117          MOV C,A
        003323  303 170 003  JMP,RETPT
DIV     003326  171          MOV A,C          C=C/B
CTUE    003331  014          INCC
        003332  220          SUB B
        003333  312          JZ,ZER
        003336  332 344 003  JC,MIN
        003341  303 331 003  JMP,CTUE
MIN     003344  015          DCR C
ZER     003345  303 170 003  JMP,RETPT
CONSTANT 003350 023          INX DE
        003351  032          LDAX DE
        003352  315 062 005  CAL,CHEKN
        003355  332 350 003  JC,CONSTANT
        003360  353          XCHG
        003361  315 310 005  CAL,SURE
        003364  353          XCHG
        003365  107          MOV B,A
        003366  303 247 003  JMP,GETNET
ASC     003371  023          INX DE
        003372  032          LDAX DE
        003373  107          MOV B,A
        003374  023          INX DE
        003375  303 134 003  JMP,GETNET
RND     004000  041 051 004  LXI HL:SH+3      Random Number Generator
        004003  006 010      MVI B 010          by Jim Parker
RTOP    004006  007          RLC
        004007  007          RLC
        004010  007          RLC
        004011  256          XRA M
        004012  027          RAL
        004013  027          RAL
        004014  055          DCR L
        004015  055          DCR L
        004016  055          DCR L
        004017  176          MOV A,M
        004020  027          RAL
        004021  167          MOV M,A
        004022  054          INR L
        004023  176          MOV A,M
        004024  027          RAL
        004025  167          MOV M,A
        004026  054          INR L
        004027  176          MOV A,M
        004030  027          RAL
        004031  167          MOV M,A
        004032  054          INR L
        004033  176          MOV A,M
        004034  027          RAL
        004035  167          MOV M,A
        004036  004          DCR B
        004037  302 006 004  JNZ,RTOP
        004042  107          MOV B,A
        004043  303 134 003  JMP,GETNET
SH      004046  021 231 132  SEED
        004051  351          DATA
INPTX   004052  016 000      MVIC 000          Input a line of 72 characters
INPXT   004054  041 207 006  LXI HL:TEXT-1
INO     004057  327          RST INPUT
HRER    004057  327          MOV B,A
        004061  171          MOV A,C
        004062  376 000      CPI 000
        004064  170          MOV A,B
        004065  302 075 004  JNZ,MID
        004070  376 240      CPI "SP"          Do not accept space if outside
        004072  312 057 004  JZ,INO             quotes
MID     004075  376 242      CPI '"'
        004077  302 117 004  JNZ,GOON
        004102  171          MOV A,C
        004103  376 000      CPI 000
        004105  312 115 004  JZ,MRF
        004110  016 000      MVI C 000
        004112  303 117 004  JMP,GOON
MRF     004115  016 003      MVI C 003
GOON    004117  376 214      CPI X^L           If X^L redo line
        004121  302 133 004  JNZ,HRD
        004124  076 334      MVIA "/"
        004126  347          RST OUT
        004127  317          RST CRLF
        004130  303 052 004  JMP,INPTXT
HRD     004133  376 215      CPI "CR"
        004135  302 144      JNZ,CHM
        004140  303 316 005  JMP,HELP
        004143  000          NOP
CHM     004144  376 223      CPI X^S           If X^S go back a character
        004146  302 160 004  JNZ,CTN
        004151  076 337      MVI A "←"
        004153  347          RST OUT
        004154  053          DCX HL
        004155  303 057 004  JMP,IND
CTN     004160  175          MOV A,L
        004161  376 320      CPI 320           Low top address
```

```
        004163  312 200 004  JZ,ERR2          If over 72 characters, report
        004166  043          INX HL             error
STOR    004167  160          MOV M,B
        004170  303 057 004  JMP,INO
ERR1    004173  006 261      MVI B "1"
        004175  303 226 004  JMP,ERR
ERR2    004200  006 262      MVI B "2"
        004202  303 226 004  JMP,ERR
ERR3    004205  006 263      MVI B "3"
        004207  303 226 004  JMP,ERR
ERR4    004212  006 264      MVI B "4"
        004214  303 226 004  JMP,ERR
ERR5    004217  006 265      MVI B "5"
        004221  303 226 004  JMP,ERR
ERR6    004224  006 266      MVI B "6"
ERR     004226  317          RST CRLF          Print "ERR" message
        004227  021 077 003  LXI DE:ERR        "!ERR"
        004232  315 360 004  CAL,PRINTXT
        004235  170          MOV A,B
        004236  347          RST OUT
        004237  000          NOP
AT+     004240  021 105 003  LXI DE             " AT "
        004243  315 360 004  CAL,PR INTXT
        004246  072 121 003  LDA:STATN
        004251  107          MOV B,A
        004252  315 174 005  CAL,PBINBCD
        004255  307          RST 000
MKBIN   004256  325          PUSH DE           Get value of ASCII Numbers
        004257  053          DCX HL
        004260  176          MOV A,M
        004261  326 260      SUI 260
        004263  107          MOV B,A
        004264  053          DCX HL
        004265  176          MOV A,M
        004266  315 062 005  CAL,CHEKN
        004271  332 303 004  JC,STOC
        004274  016 000      MVI C 0
        004276  036 000      MVI E 0
        004300  303 330 004  JMP,INK2
STOC    004303  326 260      SUI 260
        004305  117          MOV C,A
ENT     004306  053          DCX HL
        004307  176          MOV A,M
        004310  315 062 005  CAL,CHEKN
        004313  332 323 004  JC,STOE
        004316  036 000      MVI E 0
        004320  303 327 004  JMP,INK3
STOE    004323  176          MOV A,M
        004324  326 260      SUI 260
        004326  137          MOV E,A
INK3    004327  043          INX HL
INK2    004330  043          INX HL
        004331  043          INX HL
        004332  315 102 005  CAL,BCDBIN
        004335  170          MOV A,B
        004336  062 120 003  STA:BIN
        004341  321          POP DE
        004342  311          RET
GETADR  004343  345          PUSH HL           Get Address of variable
        004344  021 007 005  LXI DE:VARSTOR
        004347  326 301      SUI 301
        004351  046 000      MVI H 000
        004353  157          MOV L,A
        004354  031          DAD DE
        004355  353          XCHG
        004356  341          POP HL
        004357  311          RET
PRINTXT 004360  032          LDAX DE           Print text pointed to by DE
        004361  376 377      CPI 377
        004363  310          RZ
        004364  347          RST OUT
        004365  023          INX DE
        004366  303 360 004  JMP,PRINTXT
CHEKLTR 004371  067          STC               Check if a character is a
        004372  077          CMC                 letter
        004373  376 301      CPI 301
        004375  332 004 005  JC,NOTAP
        005000  067          STC
        005001  376 333      CPI 333
        005003  311          RET
NOTAP   005004  077          CMC
        005005  311          RET
        005006  000          NOP
VARSTOR 005007-005042   000                      Variable storage
TERM    005043  376 215      CPI "CR"          Check for statement terminator
        005045  312 060 005  JZ,YES                (CR or :)
        005050  376 272      CPI ":"
        005052  312 060 005  JZ,YES
        005055  067          STC
        005056  077          CMC
        005057  311          RET
YES     005060  067          STC
        005061  311          RET
CHECKN  005062  067          STC
        005063  077          CMC
        005064  376 260      CPI 260
        005066  332 075 005  JC,NOTA
        005071  067          STC
        005072  376 272      CPI 272
NOT     005074  311          RET
NOTA    005075  077          CMC
        005076  311          RET
        005077  000 000 000  NOP
BCDBIN  005102  076 012      MVI A 012         BCD to BIN subroutine
        005104  200          ADD B
        005105  107          MOV B,A
        005106  015          DCR B
        005107  302 102 005  JNZ,BCDBIN
        005112  000 000 000  NOP
SEC     005115  113          MOV C,E
        005116  173          MOV A,E
        005117  376 377      CPI 377
THI     005121  310          RZ
        005122  076 144      MVI A 144
        005124  200          ADD B
        005125  107          MOV B,A
        005126  015          DCR C
        005127  303 121 005  JMP,THI
LIST    005132  021 001 007  LXI DE:PROG+1     List Command
NEXN    005135  032          LDAX DE
        005136  376 377      CPI 377
        005140  312 013 001  JZ,LPUB
        005143  107          MOV B,A
        005144  315 174 005  CAL,PBINBCD
        005147  076 240      MVI A "SP"
        005151  347          RST OUT
        005152  023          INX DE
        005153  032          LDAX DE
OU      005154  347          RST OUT
        005155  376 215      CPI "CR"
        005157  302 167 005  JNZ,MREN
        005162  023          INX DE
        005163  317          RST CRLF
        005164  303 135 005  JMP,NEXN
MREN    005167  023          INX DE
        005170  032          LDAX DE
        005171  303 154 005  JMP,OU
PBINBCD 005174  305          PUSH BC             Print Binary number
        005175  325          PUSH DE
        005176  026 000      MVI D 000
        005200  016 000      MVI C 000
        005202  170          MOV A  B
IFIR    005203  326 144      SUI 144
        005205  332 214 005  JC,ISEC
```

```
        005210  014          INC C
        005211  303 203 005  JMP,IFIR
ISEC    005214  006 144      MVI B 144
        005216  200          ADD B
        005217  107          MOV B,A
        005220  076 260      MVI A 260
        005222  201          ADD C
        005223  376 260      CPI 260
        005225  312 271 005  JZ,NP
        005230  347          RST OUT
GOM     005231  016 000      MVI C 000
        005233  170          MOV A,B
ITHR    005234  326 012      SUI 012
        005236  332 245 005  JC,FOR
        005241  014          INC C
        005242  303 234 005  JMP,ITHR
FOR     005245  006 012      MVI B 012
        005247  200          ADD B
        005250  107          MOV B,A
        005251  076 260      MVI A 260
        005253  201          ADD C
        005254  376 260      CPI 260
        005256  276 005      JZ,INV
IPR     005261  347          RST OUT
DPR     005262  076 260      MVI A 260
        005264  200          ADD B
        005265  347          RST OUT
        005266  321          POP DE
        005267  301          POP BC
        005270  311          RET
NP      005271  026 001      MVI D 001
        005273  303 231 005  JMP,COM
INU     005276  117          MOV C,A
        005277  257          XRA A
        005300  272          CMP D
        005301  171          MOV A,C
        005302  302 262 005  JNZ,DPR
        005305  303 261 005  JMP,IPR
SURE    005310  305          PUSH BC          Calls MKBIN with saving BC
        005311  315 256 004  CAL.MKBIN
        005314  301          POP BC
        005315  311          RET
HELP    005316  043          INX HL           Adds 377 terminator at end of
        005317  160          MOV A,B           INPUT Text
        005320  043          INX HL
        005321  066 377      MVI M 377
        005323  311          RET
IF      005324  021 151 006  LXI DE:EXPR      IF Statement
NGO     005327  043          INX HL           First expression
        005330  176          MOV A,M
        005331  376 243      CPI "#"
        005333  312 361 005  JZ,COMP
        005336  376 275      CPI "="
        005340  312 361 005  JZ,COMP
        005343  376 274      CPI "<"
        005345  312 361 005  JZ,COMP
        005350  315 043 005  CAL,TERM
        005353  315 334 006  CAL,DIN
        005356  303 327 005  JMP,NGO
COMP    005361  365          PUSH PSW
        005362  076 215      MVI A "CR"
        005364  022          STAX DE
        005365  315 162 003  CAL,EXPRS
        005370  305          PUSH BC
        005371  021 151 006  LXI DE:EXP
NXTVR   005374  043          INX HL           Second expression
        005375  176          MOV A,M
        005376  376 273      CPI ","
        006000  312 016 006  JZ,PHI
        006003  315 043 005  CAL,TERM
        006006  332 217 004  JC,ERR5
        006011  022          STAX DE
        006012  023          INX DE
        006013  303 374 005  JMP,NXTVR
PHI     006016  043          INX HL
        006017  076 215      MVI A "CR"
        006021  022          STAX D
        006022  315 162 003  CAL,EXPRS
        006025  321          POP DE
        006026  361          POP PSW
        006027  376 243      CPI "A"          Check relational operator
        006031  312 140 003  JZ,NOTEQ
        006034  376 274      CPI "<"
        006036  312 051 006  JZ,LESSTH
EQU     006041  173          MOV A,E          Check if =
        006042  271          CMP C
        006043  312 041 001 JZ,EXEC
        006046  303 013 001  JMP,LPUB
LESSTH  006051  173          MOV A,E          Check if <
        006052  271          CMP C
        006053  322 013 001  JC,EXEC
        006056  303 041 001  JMP,LPUB
        006061  000          NOP
VAL     006062  325          PUSH DE          Get address of memory location
        006063  021 151 006  LXI DE:EXPR
SHME    006066  043          INX HL
        006067  176          MOV A,M
        006070  022          STAX DE
        006071  315 043 005  CAL,TERM
        006074  315 342 006  CAL,PLIN
        006077  376 254      CPI ","
        006101  302 066 006  JNZ,SMHE
        006104  315 347 006  CAL,DCXN
        006107  315 162 003  CAL,EXPRS
        006112  305          PUSH BC
        006113  021 151 006  LXI DE:EXPR
MIG     006116  043          INX HL
        006117  176          MOV A,M
        006120  022          STAX DE
        006121  315 043 005  CAL,TERM
        006124  315 342 006  CAL,PLIN
        006127  376 251      CPI ")"
        006131  302 116 006  JNZ,MIG
        006134  315 347 006  CAL,DCXN
        006137  315 162 003  CAL,EXPRS
        006142  171          MOV A,C
        006143  301          POP BC
        006144  101          MOV B,C
        006145  117          MOV C,A
        006146  321          POP DE
        006147  311          RET
        006150  253          DB "+"
EXPR    006151-006207  000                    Expression buffer
TEXT    006210-006320  000                    Line buffer (Input text)
FIX     006321  021 151 006  LXI DE:EXP
MREN    006324  176          MOV A,M          Most of the following is
        006325  022          STAX DE           to patch up mistakes in
        006326  315 043 005  CAL,TERM          the original program
        006331  303 242 001  JMP,BACK
DIN     006334  332 217 004  JC,ERR5
        006337  022          STAX DE
        006340  023          INX DE
        006341  311          RET
PLIN    006342  023          INX DE
        006343  332 217 004  JC,ERR5
        006346  311          RET
DCXN    006347  033          DCX DE
        006350  076 215      MVI A "CR"
        006352  022          STAX DE
CIN     006354  043          INX HL
        006355  303 263 002  JMP,LD
DMS     006360  305          PUSH BC
        006361  315 054 004  CAL,INPXT
        006364  301          POP BC
        006365  311          RET
        006366-006377  000                    Extra space for user's use.
        007000+                               MINOL Programs
```

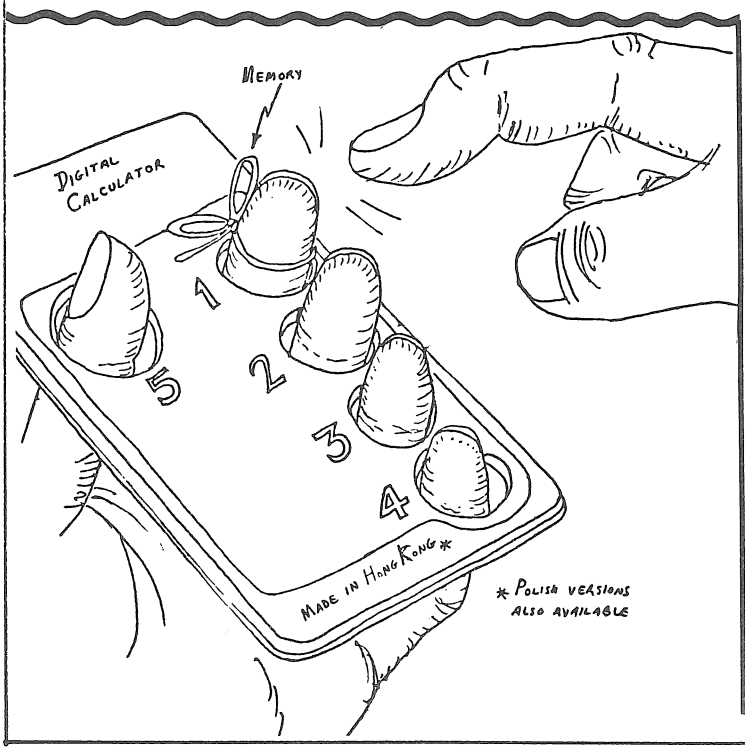
System monitor for 8080-based microcomputers

by Charlie Pack, 25470 Elana Rd., Los Altos Hills CA 94022

After developing an 82-byte octal loader using nothing but the switch panel on my Altair 8800, I concluded that a system monitor was needed to provide better communications between myself and my machine. The required functions were: (1) display the contents of one byte or any segment of memory in octal, (2) display the contents of all registers, (3) enter programs in octal from a keyboard, (4) load and store programs or data, and (5) jump to a program. The I/O device to be supported was an ASR-33 Teletype with paper-tape reader and punch. The Monitor described in this article meets these requirements.

Since the Monitor has several hundred statements, and the memory in my computer is limited, I needed a cross-assembler that would run on the computers and with the software to which I had access. So, I wrote one—would you believe—in COBOL?!! The program listing for the Monitor was printed musing my COBOL cross-assembler. Since COBOL is rather inefficient at non-structured string handling and subscripting, I made the source format fixed and eliminated punctuation. Register operands were made a part of the instruction code, for example MOV AM moves a byte from memory to accumulator. Otherwise, the microprocessor instructions are normal. The only pseudo-instructions used are ORG, DB, and DW. ORG sets the location counter; DB defines a single byte, and DW defines a doubleword (two bytes). Since it is difficult to do octal arithmetic in COBOL, the bells and whistles were left out of the cross-assembler.

A paper tape of this system is being delivered to the Program Repository at the Community Computer Center, 1919 Menalto, Menlo Park CA 94025.



```
* * * * * * * * * * * * * * * * SYSTEM MONITOR FOR 8080-BASED MICROCOMPUTERS * * * * * * * * * * * * * * * * *

          WRITTEN BY CHARLIE PACK, 25470 ELENA ROAD, LOS ALTOS HILLS, CA.  94022

00.00.  SYSTEM OVERVIEW.

00.01.  PURPOSES.

THE MAIN PURPOSE OF THE SYSTEM MONITOR IS TO PROVIDE COMMUNICATIONS BETWEEN THE USER AND AN 8080-BASED MICROCOMPUTER (SUCH AS
THE ALTAIR 8800 OR IMSAI 8080) VIA A TELETYPE OR A SERIAL CRT TFRMINAL AT THE MACHINE LANGUAGE LEVEL.
THE MONITOR ALSO PROVIDES DEBUGGING FACILITIES AND A SUBROUTINE LIBRARY THAT CAN BE USED BY OTHER PROGRAMS.  FURTHERMORE, THE
MONITOR CAN BE USED TO LOAD INTERPRETERS OR OTHER PROGRAMS.

00.02.  FUNCTIONAL CHARACTERISTICS.

VERSION 1.0 OF THE 8080 SYSTEM MONITOR IS DESIGNED TO RESIDF IN THE LOWER THREE PAGES (THE FIRST 768 BYTES) OF MEMORY, INCLUDING
THE STACK AREA.  IT IS NOT RECOMMENDED THAT IT BE RE-ASSEMBLED AND RELOCATED "AS IS" TO ANOTHER MEMORY AREA, BECAUSE OF THE LOGIC
WHICH CHECKS FOR VALID ADDRESSES.  THE MONITOR IS DESIGNED TO SUPPORT A TELETYPE MACHINE, SUCH AS AN ASR-33, WITH PAPER TAPE
READEP AND PUNCH.

00.03.  SUMMARY OF CAPABILITIES.

SEVEN DIFFERENT COMMANDS ALLOW THE MONITOR TO PERFORM THESE FUNCTIONS:
A)  EXAMINE ONE BYTE AT ANY ADDRESS.
B)  DISPLAY THE CONTENTS OF ALL REGISTERS (IN OCTAL).
C)  DISPLAY THE CONTENTS OF ANY SEGMENT OF MEMORY (AS A GROUP OF THREE OCTAL DIGITS FOR EACH BYTE).
D)  ENTER MACHINE-LANGUAGE PROGRAMS FROM A KEYBOARD, 3 OCTAL DIGITS PER BYTE.
E)  LOAD BINARY DATA OR PROGRAMS FROM PAPER TAPE.
F)  STORE BINARY DATA OR PROGRAMS ON PUNCHED PAPER TAPE.
```

G) JUMP TO A USER-SUPPLIED PROGRAM AT ANY ADDRESS.

00.04. SUBROUTINE LIBRARY.

SUBROUTINES ARE AVAILABLE WITHIN THE MONITOR WHICH CAN BE CALLED BY USER PROGRAMS TO PERFORM THESE FUNCTIONS:
A) DISLAY THE CONTENTS OF ONE BYTE OR OF ANY SEGMENT OF MEMORY IN OCTAL OR ASCII, OR DISPLAY THE CONTENTS OF ALL REGISTERS.
B) LOAD AND STORE BINARY DATA ON PUNCHED PAPER TAPE.
C) PERFORM I/O OPERATIONS ON TELETYPE (OR OTHER SERIAL) KEYBOARD AND PRINTER OR DISPLAY.

01.00. HARDWARE REQUIREMENTS AND IMPLEMENTATION.

01.01. HARDWARE REQUIREMENTS.

THE MONITOR REQUIRES AN 8080-BASED MICROPROCESSOR (SUCH AS AN ALTAIR 8800 OR IMSAI 8080) EQUIPPED WITH AT LEAST 1K OF MEMORY; A TELETYPE MACHINE OR OTHER SERIAL DISPLAY WITH AT LEAST 72 CHARACTERS PER LINE; AND A SUITABLE SERIAL I/O INTERFACE.

THE DISPLAY DEVICE SHOULD BE SET UP FOR FULL DUPLEX OPERATION ON PORTS 0 AND 1. THE MONITOR AS SUPPLIED USES STATUS BIT 0 (THE RIGHTMOST BIT) FOR RECEIVED DATA AVAILABLE AND STATUS BIT 7 (THE LEFTMOST BIT) FOR TRANSMITTER BUFFER EMPTY. IN BOTH CASES A "HIGH" (1) MEANS BUSY OR NOT READY STATUS AND A "LOW" (0) MEANS READY STATUS. THESE CONVENTIONS ARE USED BY MITS ON THEIR SIO (REV. 1) SERIAL INTERFACE BOARDS.

01.02. IMPLEMENTATION.

THE MONITOR HAS BEEN RUN ON AN ALTAIR 8800 WITH 8K OF STATIC RAM, AN SIO-A BOARD (REV. 1), AND AN ASR-33 TELETYPE SET UP FOR RS-232 INTERFACE. IT IS ALSO OPERATING AT MY HOME ON AN ALTAIR 8800 WITH 8K OF DYNAMIC RAM, A PROCESSOR TECHNOLOGY 3P+S I/O BOARD (MODIFIED SO THE STATUS BITS ARE THE SAME POLARITY AS THAT USED BY MITS ON THEIR SIO BOARDS), AND AN ASR-33 TELETYPE SET UP FOR 20 MA. CURRENT LOOP OPERATION.

02.00. OPERATING INSTRUCTIONS.

02.01. LOADING THE MONITOR.

TO LOAD THE MONITOR FROM PAPER TAPE, USE THE BOOTSTRAP ON THE LAST PAGE OF THE PROGRAM SOURCE LISTING. LOAD THE BOOTSTRAP USING THE CONTROL PANEL, STARTING AT LOCATION 003,320. BE SURE AND CHECK IT WHEN YOU'RE FINISHED, BY USING THE EXAMINE SWITCH. THEN LOAD AND START THE PAPER TAPE READER AND SWITCH TO RUN, IN THAT ORDER. NOTE THAT THE PAPER TAPE SHOULD HAVE ABOUT 5 INCHES OF LEADER WHICH CONSISTS OF NULL CHARACTERS.

WHEN THE MONITOR IS LOADED, A QUESTION MARK (?) SHOULD BE DISPLAYED. IF THE MONITOR DISPLAYS ONE OR TWO CHARACTERS OF GARBAGE, PRESS THE CARRIAGE RETURN (CR) KEY AND A ? WILL BE DISPLAYED. IF NOTHING IS DISPLAYED, AND THE TELETYPE OR VIDEO UNIT HAS NOT SHOWN ANY SIGN OF LIFE, THE MONITOR IS NOT LOADED CORRECTLY. IF THE PROBLEM PERSISTS, TRY SINGLE-STEPPING THROUGH THE BOOT LOADER. YOU MAY HAVE THE WRONG ADDRESS PORTS OR STATUS BITS.

02.02. ENTRY OF COMMANDS AND ADDRESS PARAMETERS.

THE MONITOR IS IN COMMAND MODE WHEN A QUESTION MARK (?) IS DISPLAYED. THIS PROMPT TELLS YOU IT IS READY FOR A COMMAND. EACH FUNCTION TO BE PERFORMED BY THE MONITOR IS IDENTIFIED BY A 1-CHARACTER ALPHABETIC COMMAND. A COMMAND IS USUALLY FOLLOWED BY ONE OR TWO ADDRESS PARAMETERS.

AN ADDRESS PARAMETER IS A 6-DIGIT SPLIT OCTAL NUMBER. THE FIRST 3 DIGITS ARE THE MEMORY ADDRESS WITHIN A PAGE (MAXIMUM VALUE IS OCTAL 377) AND THE LAST 3 DIGITS ARE THE MEMORY PAGE NUMBER. A PAGE IS A 256-BYTE SEGMENT OF ADDRESSABLE MEMORY. THE PAGE NUMBER CANNOT EXCEED THE HIGHEST NUMBERED PAGE OF EXISTING MEMORY, I.E. 017 FOR 4K, 037 FOR 8K, AND SO ON. EXAMPLES OF ADDRESS PARAMETERS FOLLOW:

```
000000 IS LOCATION 0 WITHIN PAGE 0
001000 IS LOCATION 1 WITHIN PAGE 0
377000 IS LOCATION 255 (DECIMAL) WITHIN PAGE 0
000001 IS LOCATION 0 WITHIN PAGE 1
```

377017 IS LOCATION 255 WITHIN PAGE 15 (DECIMAL) - THE HIGHEST NUMBERED LOCATION IF YOU HAVE 4K OF MEMORY.

NOTE THAT THE NUMBERING OF ADDRESSES AND PAGES BEGINS WITH 0, NOT 1.

IF WHILE ENTERING AN ADDRESS YOU MAKE A MISTAKE, YOU CAN TYPE @ (UPPER-CASE P ON A TELETYPE MACHINE) AT ANY TIME TO RETURN TO COMMAND MODE. THEN, YOU CAN RE-TYPE THE COMMAND AND RE-ENTER THE CORRECT ADDRESS. NO EMBEDDED SPACES ARE PERMITTED BETWEEN A COMMAND AND AN ADDRESS, BETWEEN TWO ADDRESSES, OR WITHIN AN ADDRESS. AN INVALID CHARACTER TYPED WHILE ENTERING AN ADDRESS IS NOT "ECHOED" AND IS IGNORED BY THE SYSTEM.

AFTER THE ADDRESS PARAMETERS HAVE BEEN ENTERED AS REQUIRED FOR THE PARTICULAR COMMAND, THE MONITOR WAITS FOR YOU TO PRESS CR (CARRIAGE RETURN) TO BEGIN THE COMMAND FUNCTION, OR TO TYPE AN @ TO ABORT AND RETURN TO COMMAND MODE. THE PURPOSES OF THIS EXTRA MANEUVER ARE TO ALLOW THE OPERATOR TO READY AN I-O DEVICE (SUCH AS A PAPER TAPE PUNCH) IF REQUIRED, OR TO SIGHT-VERIFY THE ADDRESS.

02.03. DESCRIPTION OF COMMAND FUNCTIONS.

02.03.01. EXAMINE (E).

EBBBBBB EXAMINES THE CONTENTS OF THE MEMORY BYTE AT LOCATION BBBBBB. THE CONTENTS OF THE MEMORY BYTE IS DISPLAYED AS A GROUP OF 3 OCTAL DIGITS. ANY EXISTING ADDRESS CAN BE EXAMINED.

FOR EXAMPLE E000000 FOLLOWED BY A CR (CARRIAGE RETURN) WILL DISPLAY 303 (SEE PROGRAM LISTING).

02.03.02. JUMP TO PROGRAM (J).

JBBBBBB RESULTS IN A JUMP TO LOCATION BBBBBB TO TURN CONTROL OVER TO A USER-SUPPLIED PROGRAM. THE LOCATION SPECIFIED MUST BE AN ADDRESS WITHIN EXISTING MEMORY BUT NOT WITHIN THE MONITOR AREA.

IF THE USER PROGRAM IS VERY COMPLEX, IT SHOULD HAVE ITS OWN STACK AREA AS THE MONITOR HAS ONLY 37 BYTES IN ITS STACK. THE USER PROGRAM MAY GO BACK TO THE MONITOR IN COMMAND MODE BY ISSUING A RET (RETURN) INSTRUCTION OR BY A JUMP TO LOCATION 000,000.

FOR EXAMPLE J000003 FOLLOWED BY CR WILL RESULT IN A JUMP TO LOCATION 0 IN PAGE 3.

02.03.03. KEYBOARD OCTAL LOADER (K).

KBBBBBB INVOKES THE KEYBOARD OCTAL LOADER. THE USER KEYS IN THREE ASCII DIGITS FOR EACH BYTE OF MEMORY, STARTING AT LOCATION BBBBBB AND INCREMENTING THROUGH CONSECUTIVE MEMORY LOCATIONS. EACH GROUP OF 3 DIGITS CAN HAVE ANY OCTAL VALUE FROM 000 TO 377. VALID DIGITS ARE "ECHOED" TO THE DISPLAY; INVALID CHARACTERS ARE IGNORED AND ARE NOT STORED IN MEMORY. ENTRY OF A CR AFTER KEYING IN THE BEGINNING ADDRESS IS OPTIONAL.

THE SPACE BAR (SP) AND CARRIAGE RETURN (CR) KEYS MAY BE USED AT ANY TIME TO FORMAT THE DISPLAY AS REQUIRED, WITHOUT AFFECTING DATA IN MEMORY. ON A TELETYPE MACHINE YOU MUST ENTER A CR TO KEEP FROM OVERPRINTING THE LAST CHARACTER ON THE LINE. WHEN YOU ARE FINISHED, TYPE @ TO RETURN TO COMMAND MODE (THE @ IS NOT STORED IN MEMORY).

THE KEYBOARD OCTAL LOADER ALSO ALLOWS THE USER TO BACK SPACE ONE OCTAL DIGIT OR ONE BYTE, BY TYPING A BACK ARROW (UPPER-CASE O ON A TELETYPE MACHINE). IF A BYTE HAS BEEN PARTIALLY ENTERED, THE BACK ARROW WILL BACK-SPACE ONE OCTAL DIGIT. FOR EXAMPLE: 2 123 RESULTS IN 123 (OCTAL) BEING STORED; 26 46 RESULTS IN 246 BEING STORED. IF A BYTE HAS BEEN COMPLETED, THE BACK ARROW WILL BACK-SPACE ONE BYTE. IT IS POSSIBLE TO BACK-SPACE ANY NUMBER OF BYTES AS LONG AS THE AREA OCCUPIED BY THE MONITOR IS NOT VIOLATED. BACK-SPACING IS NOT DESTRUCTIVE. FOR EXAMPLE:

USER ENTERS K000003 303 055 045 000 -- 303 045 000 IS STORED (3 BYTES) BEGINNING AT LOCATION 0 WITHIN PAGE 3.
USER ENTERS K020017 060 063 065 067 061 -- 061 063 065 067 IS STORED (4 BYTES) BEGINNING AT 020,017.

02.03.04. LOAD BINARY DATA (L).

LBBBBBBEEEEEE LOADS BINARY DATA FROM PAPER TAPE, BEGINNING AT ENDING LOCATION EEEEEE AND PROCEEDING BACKWARDS (DECREMENTING) TO

BEGINNING LOCATION BBBBBB INCLUSIVE. THE ADDRESS RANGE MUST BE IN EXISTING MEMORY, BUT NO PART OF IT CAN BE WITHIN THE MONITOR AREA.

THE REQUIRED PROCEDURE IS: A) LOAD TAPE IN READER. B) ENTER THE ADDRESSES AS SHOWN AND MAKE SURE THEY ARE CORRECT BY SIGHT VERIFICATION. C) TYPE CR (OR TYPE @ TO RETURN TO COMMAND MODE). D) START THE READER.

EACH CHARACTER POSITION ON THE PAPER TAPE WILL OCCUPY ONE BYTE OF MEMORY; THUS (EEEEEE - BBBBBB + 1) BYTES WILL BE LOADED. THE TAPE MUST HAVE A LEADER (ANY LENGTH) WHICH CONTAINS ALL NULL CHARACTERS (BINARY ZEROES) AND WHICH WILL NOT BE TRANSFERRED TO MEMORY. TRANSFER OF DATA TO MEMORY BEGINS AT THE FIRST NON-NULL CHARACTER ON THE TAPE. ONCE THE FIRST NON-NULL CHARACTER HAS BEEN ENCOUNTERED AND STORED IN MEMORY, ALL CHARACTERS (INCLUDING NULLS) AFTER THAT ARE TRANSFERRED TO DECREASING MEMORY LOCATIONS. THERE IS NO CHECKSUM ROUTINE; THE MONITOR ASSUMES THE TAPE IS CORRECTLY FORMATTED AND CONTAINS THE CORRECT NUMBER OF DATA CHARACTERS.

FOR EXAMPLE, TO LOAD 32 BYTES OF DATA BEGINNING AT 020,003 AND ENDING AT 057,003 THE USER WOULD TYPE L020003057003 FOLLOWED BY A CR AND THEN START THE READER. WHEN LOADING IS COMPLETE THE MONITOR AUTOMATICALLY RETURNS TO COMMAND MODE.

02.03.05. MEMORY DUMP (M).

MBBBBBBEEEEEE DISPLAYS ALL MEMORY LOCATIONS BETWEEN BEGINNING ADDRESS BBBBBB AND ENDING ADDRESS EEEEEE INCLUSIVE. ANY AREA WITHIN EXISTING MEMORY MAY BE DISPLAYED, ONE GROUP OF 3 OCTAL DIGITS FOR EACH BYTE. EACH LINE DISPLAYED WILL SHOW 16 BYTES OF MEMORY, PLUS THE ADDRESS OF THE FIRST BYTE ON THE LINE IN SPLIT OCTAL.

FOR EXAMPLE, M000000003000 FOLLOWED BY A CR WILL DISPLAY (REFER TO PROGRAM LISTING):

```
000,000: 303 050 000 000
```

02.03.06. REGISTER DUMP (R).

R CAUSES THE CONTENTS OF THE REGISTERS TO BE DISPLAYED (IN OCTAL) IN THE FOLLOWING ORDER: PSW A C B E D L H THE DISPLAY BEGINS IMMEDIATELY AFTER THE "R" COMMAND IS TYPED; NO PARAMETERS ARE REQUIRED. NOTE THAT THE REGISTERS ARE NOT DISPLAYED IN SEQUENCE! THIS IS BECAUSE OF THE WAY IN WHICH THEY ARE "PUSHED" ON THE STACK.

WHEN THE REGISTER DUMP IS INVOKED BY THE MONITOR THE CONTENTS OF THE PSW AND ACCUMULATOR WILL ALWAYS BE 126 122 AND THEREFORE ARE OF LITTLE VALUE. HOWEVER, THE REGISTER DISPLAY SUBROUTINE CAN ALSO BE CALLED FROM A USER PROGRAM.

02.03.07. SAVE BINARY DATA (S).

SBBBBBBEEEEEE SAVES BINARY DATA ON PAPER TAPE, BEGINNING AT ENDING LOCATION EEEEEE AND PROCEEDING BACKWARDS (DECREMENTING) TO BEGINNING LOCATION BBBBBB INCLUSIVE. THE ADDRESS RANGE MUST BE IN EXISTING MEMORY. THE REQUIRED PROCEDURE IS: A) LOAD BLANK PAPER TAPE IN THE PUNCH. B) ENTER THE ADDRESSES AS SHOWN AND MAKE SURE THEY ARE CORRECT BY SIGHT VERIFICATION. C) TYPE CR (OR TYPE @ TO RETURN TO COMMAND MODE).

THE PUNCH WILL BEGIN AUTOMATICALLY BY PUNCHING OUT A 5-INCH LEADER CONTAINING NULL CHARACTERS (BINARY ZEROES). AFTER THAT, ONE CHARACTER POSITION ON THE TAPE IS PUNCHED FOR EACH BYTE OF MEMORY UNTIL (EEEEEE - BBBBBB + 1) BYTES HAVE BEEN PROCESSED.

THE BYTE AT LOCATION EEEEEE CANNOT BE ZERO BECAUSE ZEROES (NULLS) ARE USED TO REPRESENT THE LEADER. A 3-INCH TRAILER OF NULLS IS PUNCHED AT THE END TO "EJECT" THE TAPE SO THE USER CAN CUT IT OFF. THERE IS NO CHECKSUM ROUTINE BUT THE OUTPUT WILL BE PROPERLY FORMATTED FOR USE WITH THE "L" COMMAND.

FOR EXAMPLE, TO STORE 32 BYTES OF DATA BEGINNING AT LOCATION 020,003 AND ENDING AT 057,003 THE USER WOULD TYPE S020003057003 FOLLOWED BY A CR. WHEN THE PROCESS IS COMPLETE THE MONITOR AUTOMATICALLY RETURNS TO COMMAND MODE.

IT IS POSSIBLE TO PUNCH OUT A COPY OF THE MONITOR ITSELF BY USING THE COMMAND S000000377002 . HOWEVER, GARBAGE WILL BE LEFT IN THE STACK AREA (000,003 THROUGH 000,047) ON THE TAPE.

02.03.08. RETURN TO COMMAND MODE (@).

@ MAY BE TYPED AT ANY TIME TO RETURN TO COMMAND MODE, EXCEPT WHEN THE MONITOR IS IN THE PROCESS OF TYPING CHARACTERS,

PUNCHING OUT A PAPER TAPE, OR DISPLAYING CHARACTERS ON THE VIDEO SCREEN. TO INTERRUPT THE MONITOR DURING OUTPUT IT IS NECESSARY TO STOP THE COMPUTER AND RE-START THE PROGRAM AT LOCATION 000,000.

02.04. ERROR MESSAGES.

THE MONITOR USES SEVERAL ERROR MESSAGES. AFTER THE ERROR MESSAGE IS DISPLAYED, THE MONITOR RETURNS TO COMMAND MODE AND DISPLAYS A QUESTION MARK (?). A DESCRIPTION OF EACH ERROR MESSAGE FOLLOWS:

WHAT? -- AN INVALID COMMAND WAS TYPED IN.

> MEM -- ADDRESS IS OUTSIDE EXISTING MEMORY. POSSIBLY THE PAGE NUMBER AND LOCATION WITHIN THE PAGE WERE REVERSED, SUCH AS 002377 INSTEAD OF 377002. THIS COULD ALSO HAPPEN WHEN USING THE KEYBOARD OCTAL LOADER IF THE UPPER LIMIT OF EXISTING MEMORY HAS BEEN REACHED.

L > H -- WHEN AN ADDRESS RANGE HAS BEEN TYPED IN, THE LOWER ADDRESS (BBBBBB) IS GREATER THAN THE UPPER ADDRESS (EEEEEE); THEREFORE THE ADDRESS RANGE IS NOT VALID.

V MON -- ADDRESS VIOLATES MONITOR AREA. THIS COULD HAPPEN IF AN INCORRECT ADDRESS IS TYPED IN, OR IF THE KEYBOARD OCTAL LOADER IS IN USE AND THE USER INADVERTENTLY TRIED TO BACKSPACE INTO THE MONITOR AREA.

03.00. SUBROUTINE LIBRARY.

03.01. GENERAL INFORMATION ABOUT SUBROUTINES.

THE MONITOR CONTAINS A NUMBER OF SUBROUTINES WHICH MAY BE CALLED FROM USER PROGRAMS. EXCEPT WHERE OTHERWISE NOTED, ALL GENERAL PURPOSE REGISTERS (B, C, D, E, H AND L) ARE EITHER SAVED OR NOT USED; THEREFORE AFTER A SUBROUTINE IS EXECUTED THOSE REGISTERS WILL HAVE THE SAME CONTENTS AS THEY DID BEFORE THE SUBROUTINE WAS EXECUTED. AS A RULE, THE ACCUMULATOR AND PSW ARE NOT SAVED.

03.02. DESCRIPTION OF SUBROUTINES.

03.02.01. REGDMP DISPLAYS THE CONTENTS OF ALL REGISTERS (EXCEPT SP AND PC) IN THE FOLLOWING ORDER:

PSW A C B E D L H (NOTE ODD-BALL SEQUENCE!)

IN ADDITION TO THE GENERAL PURPOSE REGISTERS, PSW AND A ARE SAVED, SO THE DISPLAYED CONTENTS OF THESE ARE MEANINGFUL.

03.02.02. OUT3 DISPLAYS 3 OCTAL DIGITS WHICH REPRESENT THE CONTENTS OF THE ACCUMULATOR.

03.02.03. MDUMP DISPLAYS THE CONTENTS OF MEMORY, 3 OCTAL DIGITS PER BYTE. THE STARTING (LOWER) ADDRESS MUST BE LOADED IN REGS. D AND E, AND THE ENDING (UPPER) ADDRESS MUST BE LOADED IN REGS. H AND L.

03.02.04. PTINP LOADS DATA FROM PAPER TAPE (SEE THE DESCRIPTION OF THE "L" COMMAND). THE STARTING (LOWER) ADDRESS MUST BE LOADED IN REGS. D AND E, AND THE ENDING (UPPER) ADDRESS MUST BE LOADED IN REGS. H AND L.

03.02.05. PTOUT SAVES DATA ON PAPER TAPE (SEE THE DESCRIPTION OF THE "S" COMMAND). THE STARTING (LOWER) ADDRESS MUST BE LOADED IN REGS. D AND E, AND THE ENDING (UPPER) ADDRESS MUST BE LOADED IN REGS. H AND L.

03.02.06. INCH GETS ONE CHARACTER FROM THE KEYBOARD AND PUTS IT IN THE ACCUMULATOR. THE LEFTMOST BIT (BIT 7) IS ZEROED OUT AND IS NOT USED BY THE MONITOR. SOME DEVICES USE THAT BIT AS A PARITY BIT.

03.02.07. OUTSTR DISPLAYS A CHARACTER STRING UP TO 256 BYTES IN LENGTH. AT ENTRY, REGS. H AND L MUST CONTAIN THE BEGINNING ADDRESS OF THE STRING. THE FIRST BYTE OF THE STRING MUST BE THE LENGTH OF THE STRING WITHOUT THE LENGTH BYTE. THE LENGTH BYTE IS NOT DISPLAYED.

FOR EXAMPLE, TO DISPLAY THE LETTERS "ABC" THE STRING WOULD CONTAIN (IN OCTAL) 003 101 102 103 . REGS. H AND L SHOULD POINT TO

THE BYTE THAT CONTAINS 003.

WARNING: IF THE LENGTH BYTE CONTAINS ZERO, 256 BYTES WILL BE DISPLAYED. ALSO NOTE THAT LINE FEED, CARRIAGE RETURN AND OTHER SPECIAL CHARACTERS WILL PERFORM THEIR NORMAL FUNCTIONS ON THE OUTPUT DEVICE BEING USED.

03.02.08. OUTNEW STARTS A NEW LINE ON THE OUTPUT DEVICE. A CR IS ISSUED, FOLLOWED BY A LF (LINE FEED).

03.02.09. OUTSP SKIPS ONE CHARACTER POSITION ON THE OUTPUT DEVICE. A SP (SPACE) IS ISSUED.

03.02.10. OUTCH SENDS ONE CHARACTER TO THE DISPLAY DEVICE. THE CHARACTER TO BE OUTPUTTED MUST BE IN THE ACCUMULATOR AT ENTRY. ANY OF THE 256 POSSIBLE BIT COMBINATIONS CAN BE OUTPUTTED. SPECIAL CHARACTERS (LF, CR, ETC.) WILL PERFORM THEIR NORMAL FUNCTIONS. THE PSW AND ACCUMULATOR ARE SAVED.

04.00. MODIFICATIONS TO THE MONITOR.

04.01. MEMORY SIZE.

THE 8080 MONITOR WILL RUN IN ANY COMPUTER WITH 1K OR MORE OF MEMORY. THE ADDRESS INPUT ROUTINE, HOWEVER, COMPARES USER-ENTERED ADDRESSES WITH A CONSTANT TO DETERMINE IF THEY ARE WITHIN EXISTING MEMORY. THIS CONSTANT IS LABELLED MTOPH AND IT RESIDES AT LOCATION 002,377. THE VALUE OF THIS BYTE SHOULD EQUAL THE NUMBER OF THE HIGHEST PAGE IN MEMORY. PAGES ARE 256-BYTE CONTIGUOUS SEGMENTS OF MEMORY AND ARE NUMBERED BEGINNING WITH 0 FOR THE PAGE WHICH BEGINS AT THE LOWEST POSSIBLE MEMORY LOCATION. SO IF YOU HAVE 1K OF MEMORY YOU HAVE 4 PAGES, THE HIGHEST BEING PAGE 003. FOR 4K OF MEMORY THE HIGHEST IS PAGE 017 (OCTAL), FOR 8K IT IS 037, FOR 12K IT IS 057 AND SO ON. MAKE SURE THE VALUE OF THE BYTE AT MTOPH IS CORRECT FOR THE AMOUNT OF MEMORY YOU HAVE.

04.02. INPUT-OUTPUT.

IF YOUR HARDWARE CONFIGURATION DOESN'T CONFORM TO THE MITS SIO BOARD REV. 1 SPECIFICATIONS, AND YOU NEED TO MODIFY THE MONITOR, THERE ARE FOUR AREAS TO CONSIDER: SUBROUTINES PTINP, INCH AND OUTCH, AND THE BOOT LOADER. ASIDE FROM THE BOOT LOADER AND PTINP WHICH DO THEIR OWN INPUT, ALL OTHER ROUTINES THAT NEED I/O USE INCH AND OUTCH. EXACTLY WHAT CHANGES ARE NEEDED DEPENDS ON WHAT ADDRESS PORTS YOU WISH TO USE, POLARITY OF THE STATUS BITS AND WHICH STATUS BITS YOU WISH TO USE. CONSULT THE LITERATURE FOR YOUR SERIAL I-O BOARD.

I RECOMMEND THE FOLLOWING PROCEDURE FOR MAKING CHANGES TO THE I/O:

A) LOAD THE BOOTSTRAP PROGRAM (LAST PAGE OF PROGRAM SOURCE LISTING) AND MAKE THE NECESSARY CHANGES TO THE IN 000 AND ANI INSTRUCTIONS. ALSO PATCH IN A HLT (166) IN PLACE OF THE PCHL (351) INSTRUCTION.
B) SINGLE STEP THROUGH THE LOADER TO MAKE SURE IT WILL WORK CORRECTLY.
C) LOAD THE MONITOR; WHEN FINISHED THE CPU SHOULD HALT.
D) PATCH IN THE REQUIRED CHANGES TO THE MONITOR IN SUBROUTINES PTINP, INCH AND OUTCH.
E) START AT LOCATION 000,000 AND SINGLE STEP TO MAKE SURE THE I/O WILL WORK CORRECTLY.
F) RE-START AT 000,000 AND SWITCH TO RUN. THE MONITOR SHOULD ENTER COMMAND MODE AND TYPE A QUESTION MARK (?).

05.00. SOFTWARE SUPPORT.

I INTEND TO USE THE MONITOR FOR FURTHER PROGRAM DEVELOPMENT AND THEREFORE WILL SUPPORT IT FOR A REASONABLE PERIOD OF TIME (SAY, AT LEAST A YEAR OR SO). IF YOU HAVE PROBLEMS WITH IT, I DON'T MIND BEING CALLED AT HOME - THE PHONE IS (415) 941-0495 - EVENINGS BETWEEN 7 AND 10 PM PREFERRED.

UNDER CONSIDERATION ARE SUBSTANTIAL ENHANCEMENTS TO THE MONITOR AND TO THE SUBROUTINE LIBRARY, INCLUDING I/O SUPPORT FOR "TV TYPEWRITER", PARALLEL INTERFACE AND ACR; DECIMAL ARITHMETIC ROUTINES; CHARACTER STRING MANIPULATION; AND POSSIBLY A TEXT EDITOR. ANOTHER VERSION (1.1) OF THIS MONITOR IS ALSO OPERATIONAL AND RESIDES IN HIGH MEMORY; CONSIDERATION IS BEING GIVEN TO BURNING IT INTO ROM. VERSION 1.1 IS DESIGNED TO ENHANCE ALTAIR BASIC AND ALTAIR MONITOR-ASSEMBLER-EDITOR.

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T      PAGE   1

  1                            * 8080 SYSTEM MONITOR   VERSION 1.0   04-19-76
  2                            *
  3                            * WRITTEN BY CHARLIE PACK
  4                            *            25470 ELENA ROAD
  5                            *            LOS ALTOS HILLS, CA.  94022
  6                            *            PHONE:  (415) 941-0495 EVENINGS
  7                            *
  8                            * THIS PROGRAM OCCUPIES MEMORY PAGES 000 THROUGH 002.
  9                            * MEMORY LOCATIONS 000,003 THROUGH 000,047 ARE USED
 10                            *   FOR THE PROGRAM STACK.
 11                            *
 12  000 000   303 050 000          JMP     BEGIN
 13                            *
 14  000 050                        ORG     050000
 15  000 050   061 050 000   BEGIN  LXI SP  BEGIN   INITIALIZE STACK POINTER
 16                            *
 17  000 053   315 346 002          CALL    OUTNEW  START A NEW LINE
 18  000 056   076 077              MVI A   077     ?
 19  000 060   315 362 002          CALL    OUTCH   DISPLAY A ?
 20  000 063   315 315 002   GETCOM CALL    INCH    GET CHARACTER FROM KEYBOARD
 21  000 066   376 015              CPI     015     CARRIAGE RETURN?
 22  000 070   312 050 000          JZ      BEGIN     JMP IF TRUE
 23  000 073   376 040              CPI     040     SPACE ?
 24  000 075   332 063 000          JC      GETCOM    JMP IF TRUE
 25  000 100   315 362 002          CALL    OUTCH   "ECHO" CHARACTER
 26  000 103   315 111 000          CALL    TESTCH
 27  000 106   303 050 000          JMP     BEGIN
 28                            *
 29  000 111   376 105       TESTCH CPI     105     E=EXAMINE ONE BYTE?
 30  000 113   312 171 000          JZ      EXAMIN    JMP IF TRUE
 31  000 116   376 112              CPI     112     J=JUMP TO PROGRAM?
 32  000 120   312 206 000          JZ      JUMPTO    JMP IF TRUE
 33  000 123   376 113              CPI     113     K=KEYBOARD OCTAL LOAD?
 34  000 125   312 230 000          JZ      OCTAL     JMP IF TRUE
 35  000 130   376 114              CPI     114     L=LOAD BINARY?
 36  000 132   312 076 002          JZ      LOADER    JMP IF TRUE
 37  000 135   376 115              CPI     115     M=MEMORY DUMP?
 38  000 137   312 366 001          JZ      MEMORY    JMP IF TRUE
 39  000 142   376 122              CPI     122     R=REGISTER DUMP?
 40  000 144   312 042 001          JZ      REGDMP    JMP IF TRUE
 41  000 147   376 123              CPI     123     S=STORE BINARY?
 42  000 151   312 176 002          JZ      PUNCH     JMP IF TRUE
 43  000 154   041 162 000          LXI H   BADCOM  LOAD ADDRESS OF MESSAGE
 44  000 157   303 331 002          JMP     OUTSTR  DISPLAY ERROR MSG. AND RETURN TO COMMAND MODE
 45  000 162   006 040       BADCOM DW      006040  LENGTH OF MSG. AND SP
 46  000 164   127 110              DW      127110  W AND H
 47  000 166   101 124              DW      101124  A AND T
 48  000 170   077                  DB      077     ?
 49                            *
 50                            * ROUTINE TO EXAMINE ONE BYTE
 51                            *
 52  000 171   315 145 001   EXAMIN CALL    ADDRIN  GET ADDRESS IN H-L
 53  000 174   315 345 001          CALL    WAIT    PRESS CR TO CONTINUE
 54  000 177   315 360 002          CALL    OUTSP   SKIP ONE SPACE ON OUTPUT DEVICE
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T

 55    000 202   176                   MOV AM          MOVE BYTE TO ACCUMULATOR
 56    000 203   303 100 001           JMP     OUT3    DISPLAY 3 OCTAL DIGITS AND
 57                             *                         RETURN TO COMMAND MODE.
 58                             *
 59                             * ROUTINE TO GIVE CONTROL TO A USER-SUPPLIED PROGRAM
 60                             *
 61    000 206   315 145 001   JUMPTO CALL     ADDRIN  GET ADDRESS IN REGS, H AND L
 62    000 211   072 376 002           LDA     MBOTH   MAKE SURE ADDRESS IS NOT
 63    000 214   075                   DCR A             WITHIN MONITOR AREA!
 64    000 215   274                   CMP H
 65    000 216   322 273 002           JNC     VMON
 66    000 221   315 345 001           CALL    WAIT    USER PRESSES CR TO CONTINUE
 67    000 224   315 346 002           CALL    OUTNEW  START A NEW LINE ON OUTPUT DEVICE
 68    000 227   351                   PCHL            GO TO USER PROGRAM
 69                             *
 70                             * KEYBOARD OCTAL LOADER ROUTINE
 71                             *
 72    000 230   315 145 001   OCTAL  CALL     ADDRIN  GET BEGINNING ADDRESS
 73                             *
 74    000 233   072 376 002   OCT010 LDA      MBOTH   MAKE SURE ADDRESS IS NOT
 75    000 236   075                   DCR A             WITHIN MONITOR AREA!
 76    000 237   274                   CMP H
 77    000 240   322 273 002           JNC     VMON
 78                             *
 79    000 243   001 000 002   OCT020 LXI B    000002  C=000,B=002
 80    000 246   021 000 000           LXI D    000000  E=000;D=000
 81    000 251   315 315 002   OCT030 CALL     INCH
 82    000 254   376 060               CPI     060     < 0?
 83    000 256   332 347 000           JC      OCT100    JMP IF TRUE
 84    000 261   376 070               CPI     070     > 7?
 85    000 263   322 347 000           JNC     OCT100    JMP IF TRUE
 86    000 266   110                   MOV CB
 87    000 267   015                   DCR C
 88    000 270   015                   DCR C           FIRST DIGIT?
 89    000 271   302 301 000           JNZ     OCT040    JMP IF NOT
 90    000 274   376 064               CPI     064     > 3?
 91    000 276   322 251 000           JNC     OCT030    JMP IF TRUE; INVALID CHAR.
 92    000 301   315 362 002   OCT040 CALL     OUTCH   "ECHO" THE CHARACTER
 93    000 304   346 007               ANI     007     ZERO LEFTMOST 5 BITS
 94    000 306   110                   MOV CB
 95    000 307   015                   DCR C           THIRD DIGIT?
 96    000 310   372 325 000           JM      OCT060    JMP IF TRUE
 97    000 313   015                   DCR C           SECOND DIGIT?
 98    000 314   372 322 000           JM      OCT050    JMP IF TRUE
 99    000 317   007                   RLC             ROTATE
100    000 320   007                   RLC               LEFT
101    000 321   007                   RLC                 3 BITS
102    000 322   007           OCT050 RLC              ROTATE
103    000 323   007                   RLC               LEFT
104    000 324   007                   RLC                 3 BITS
105    000 325   262           OCT060 ORA D
106    000 326   127                   MOV DA          "OR" WITH DATA BYTE
107    000 327   005                   DCR B           DECREMENT DIGIT COUNTER
108    000 330   362 251 000           JP      OCT030    JMP IF NOT LAST DIGIT
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T

109    000 333   162                    MOV MD          MOVE BYTE TO MEMORY
110    000 334   043                    INX H           INCREMENT MEMORY POINTER
111    000 335   072 377 002            LDA     MTOPH
112    000 340   274                    CMP H           STILL INSIDE EXISTING MEMORY?
113    000 341   332 263 001            JC      AD>MEM    RETURN TO COMMAND MODE IF NOT
114    000 344   303 243 000            JMP     OCT020
115    000 347   376 015        OCT100  CPI     015     CARRIAGE RETURN?
116    000 351   312 376 000            JZ      OCT110    JMP IF TRUE
117    000 354   376 040                CPI     040     SPACE?
118    000 356   312 004 001            JZ      OCT120    JMP IF TRUE
119    000 361   376 137                CPI     137     BACK ARROW?
120    000 363   312 012 001            JZ      OCT150    JMP IF TRUE
121    000 366   376 100                CPI     100     @?
122    000 370   302 251 000            JNZ     OCT030    JMP IF NOT - INVALID CHARACTER
123    000 373   303 362 002            JMP     OUTCH     JMP IF TRUE - "ECHO" THE CHAR-
124                              *                        ACTER AND RETURN TO COMMAND MODE
125    000 376   315 346 002    OCT110  CALL    OUTNEW  START A NEW LINE ON OUTPUT DEVICE
126    001 001   303 251 000            JMP     OCT030
127    001 004   315 362 002    OCT120  CALL    OUTCH   "ECHO" THE CHARACTER
128    001 007   303 251 000            JMP     OCT030
129    001 012   315 362 002    OCT150  CALL    OUTCH   "ECHO" THE CHARACTER
130    001 015   110                    MOV CB
131    001 016   015                    DCR C           THIRD OCTAL DIGIT?
132    001 017   372 032 001            JM      OCT160    JMP IF TRUE
133    001 022   015                    DCR C           SECOND OCTAL DIGIT?
134    001 023   372 243 000            JM      OCT020    JMP IF TRUE - START THIS BYTE
135                              *                        OVER AGAIN.
136    001 026   053                    DCX H           GO BACK AND RE-DO PREVIOUS BYTE.
137    001 027   303 233 000            JMP     OCT010
138    001 032   004            OCT160  INR B           GO BACK TO FIRST DIGIT
139    001 033   172                    MOV AD
140    001 034   346 300                ANI     300     ZERO OUT SECOND OCTAL DIGIT
141    001 036   127                    MOV DA
142    001 037   303 251 000            JMP     OCT030
143                              *
144                              * REGISTER DUMP ROUTINE
145                              *
146                              * THIS SUBROUTINE MAY BE CALLED FROM A USER PROGRAM.
147                              *
148    001 042   345            REGDMP  PUSH H          PUSH ALL REGISTERS ON THE STACK
149    001 043   325                    PUSH D
150    001 044   305                    PUSH B
151    001 045   365                    PUSH A
152    001 046   041 000 000            LXI H   000000
153    001 051   071                    DAD SP          STACK POINTER HAS ADDR OF FIRST REG.
154    001 052   006 007                MVI B   007     COUNTER DECREMENTS FROM 007 TO 000
155    001 054   315 346 002            CALL    OUTNEW  START A NEW LINE ON OUTPUT DEVICE
156    001 057   315 360 002    REGDMQ  CALL    OUTSP   SKIP ONE SPACE ON OUTPUT DEVICE
157    001 062   176                    MOV AM
158    001 063   315 100 001            CALL    OUT3    DISPLAY 3 OCTAL DIGITS (ONE BYTE)
159    001 066   043                    INX H           POINT TO NEXT BYTE
160    001 067   005                    DCR B           DECREMENT COUNTER
161    001 070   362 057 001            JP      REGDMQ    LOOP BACK IF NOT ZERO
162    001 073   361                    POP A           RESTORE ALL REGISTERS
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T

163    001 074   301                   POP B
164    001 075   321                   POP D
165    001 076   341                   POP H
166    001 077   311                   RET
167                            *
168                            * ROUTINE TO OUTPUT 3 OCTAL DIGITS (1 BYTE OF MEMORY)
169                            * TO SYSTEM OUTPUT DEVICE.  BYTE TO BE OUTPUTTED IS IN REG. A.
170                            *
171                            * THIS SUBROUTINE MAY BE CALLED FROM A USER PROGRAM.
172                            *
173    001 100   305           OUT3   PUSH B          PUSH REGS. B AND C ON THE STACK
174    001 101   325                  PUSH D          PUSH REGS. D AND E ON THE STACK
175    001 102   006 002              MVI B   002     OCTAL DIGIT COUNTER
176    001 104   127                  MOV DA          SAVE BYTE TO BE OUTPUTTED
177    001 105   172           OUT3A  MOV AD          LOAD BYTE IN ACCUM.
178    001 106   110                  MOV CB
179    001 107   015                  DCR C           THIRD OCTAL DIGIT?
180    001 110   372 127 001          JM      OUT3C     JMP IF TRUE
181    001 113   015                  DCR C           SECOND OCTAL DIGIT?
182    001 114   372 124 001          JM      OUT3B     JMP IF TRUE
183    001 117   346 300              ANI     300     ZERO ALL BITS THAT DO NOT
184                            *                      APPLY TO FIRST (LEFTMOST)
185                            *                      OCTAL DIGIT.
186    001 121   017                  RRC             SHIFT
187    001 122   017                  RRC               RIGHT
188    001 123   017                  RRC                 3 BITS
189    001 124   017           OUT3B  RRC             SHIFT
190    001 125   017                  RRC               RIGHT
191    001 126   017                  RRC                 3 BITS
192    001 127   346 007       OUT3C  ANI     007     ZERO OUT UNNEEDED BITS
193    001 131   366 060              ORI     060     CONVERT DIGIT TO ASCII
194    001 133   315 362 002          CALL    OUTCH   OUTPUT THE DIGIT
195    001 136   005                  DCR B           DECREMENT DIGIT COUNTER
196    001 137   362 105 001          JP      OUT3A     JMP IF NOT ALL 3 DIGITS
197                            *                        HAVE BEEN OUTPUTTED YET.
198    001 142   321                  POP D           RESTORE REGS. D AND E
199    001 143   301                  POP B           RESTORE REGS. B AND C
200    001 144   311                  RET             RETURN TO CALLER
201                            *
202                            * ROUTINE TO OBTAIN A 6-DIGIT SPLIT OCTAL ADDRESS
203                            * FROM THE SYSTEM KEYBOARD
204                            *
205    001 145   305           ADDRIN PUSH B          PUSH REGS. B AND C ON THE STACK
206    001 146   001 001 002          LXI B   001002  C=001,B=002
207    001 151   305           ADLUP1 PUSH B
208    001 152   046 000              MVI H   000     GET LOW ADDR FIRST IN REG. H AND
209                            *                        THEN TRANSFER TO REG. L.
210    001 154   315 315 002   ADLUP2 CALL    INCH
211    001 157   376 100              CPI     100     @ RETURNS TO COMMAND MODE
212    001 161   312 303 001          JZ      ADQUIT
213    001 164   376 060              CPI     060     LESS THAN 0?
214    001 166   332 154 001          JC      ADLUP2    JMP IF TRUE; INVALID CHARACTER
215    001 171   376 070              CPI     070     GREATER THAN 7?
216    001 173   322 154 001          JNC     ADLUP2    JMP IF TRUE; INVALID CHARACTER
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T
 217   001 176   110                  MOV CB
 218   001 177   015                  DCR C
 219   001 200   015                  DCR C            FIRST OCTAL DIGIT?
 220   001 201   302 211 001          JNZ     ADECHO     JMP IF NOT
 221   001 204   376 064              CPI     064      GREATER THAN 3?
 222   001 206   322 154 001          JNC     ADLUP2     JMP IF TRUE; INVALID CHARACTER
 223   001 211   315 362 002  ADECHO CALL     OUTCH    "ECHO" THE DIGIT
 224   001 214   346 007              ANI     007      ZERO OUT LEFTMOST 5 BITS
 225   001 216   110                  MOV CB
 226   001 217   015                  DCR C            THIRD OCTAL DIGIT?
 227   001 220   372 235 001          JM      ADORAH     JMP IF TRUE
 228   001 223   015                  DCR C            SECOND OCTAL DIGIT?
 229   001 224   372 232 001          JM      ADROT3     JMP IF TRUE
 230   001 227   007                  RLC              ROTATE
 231   001 230   007                  RLC                LEFT
 232   001 231   007                  RLC                  3 BITS
 233   001 232   007          ADROT3 RLC              ROTATE
 234   001 233   007                  RLC                LEFT
 235   001 234   007                  RLC                  3 BITS
 236   001 235   264          ADORAH ORA H            "OR" WITH
 237   001 236   147                  MOV HA             ADDRESS REG.
 238   001 237   005                  DCR B            DECREMENT DIGIT COUNTER
 239   001 240   362 154 001          JP      ADLUP2     JMP IF NOT LAST DIGIT
 240   001 243   301                  POP B
 241   001 244   015                  DCR C            HAVE BOTH HI & LO ADDR. BEEN ENTERED?
 242   001 245   372 254 001          JM      ADTEST     JMP IF TRUE
 243   001 250   154                  MOV LH           MOVE TO LOW ADDRESS REGISTER
 244   001 251   303 151 001          JMP     ADLUP1
 245   001 254   072 377 002  ADTEST LDA      MTOPH
 246   001 257   274                  CMP H            IS ADDR WITHIN EXISTING MEMORY?
 247   001 260   322 311 001          JNC     ADEXIT     JMP IF TRUE
 248   001 263   041 274 001  AD>MEM LXI H    ADMSG1   LOAD ADDRESS OF ERROR MESSAGE
 249   001 266   315 331 002          CALL    OUTSTR   DISPLAY AN ERROR MESSAGE
 250   001 271   303 050 000          JMP     BEGIN    BACK TO COMMAND MODE
 251   001 274   006 040      ADMSG1 DW       006040   LENGTH OF MSG, AND SP
 252   001 276   076 040              DW      076040   > AND SP
 253   001 300   115 105              DW      115105   M AND E
 254   001 302   115                  DB      115      M
 255   001 303   315 362 002  ADQUIT CALL     OUTCH    "ECHO" THE CHARACTER
 256   001 306   303 050 000          JMP     BEGIN    BACK TO COMMAND MODE
 257   001 311   301          ADEXIT POP B            RESTORE REGS. B AND C
 258   001 312   311                  RET
 259                           * ADDRESS COMPARE ROUTINE
 260                           *
 261                           * LOW ADDRESS VALUE IS IN REGS. D-E
 262                           * HIGH ADDRESS VALUE IS IN REGS. H-L
 263                           * ROUTINE RETURNS TO COMMAND MODE IF THE
 264                           *  VALUE IN D-E EXCEEDS THE VALUE IN H-L.
 265                           *
 266   001 313   172          ADCOMP MOV AD
 267   001 314   274                  CMP H            D:H  COMPARE MEMORY PAGE #
 268   001 315   330                  RC                 RETURN IF O.K.
 269   001 316   302 325 001          JNZ     ADL>H      D>H - ERROR
 270   001 321   173                  MOV AE
 271   001 322   275                  CMP L            E:L  COMPARE ADDRESS WITHIN PAGE
 272   001 323   330                  RC                 RETURN IF O.K.
 273   001 324   310                  RZ                 RETURN IF O.K.
```

```
001 325   041 336 001   ADL>H  LXI H   ADMSG2  LOAD ADDRESS OF ERROR MESSAGE
001 330   315 331 002          CALL    OUTSTR  DISPLAY AN ERROR MESSAGE
001 333   303 050 000          JMP     BEGIN   BACK TO COMMAND MODE
001 336   006 040       ADMSG2 DW      006040  LENGTH OF MSG, AND SP
001 340   114 040              DW      114040  L AND SP
001 342   076 040              DW      076040  > AND SP
001 344   110                  DB      110     H
                        *
                        * AFTER ENTERING THE ADDRESS, THIS ROUTINE ALLOWS THE USER
                        * TO ENTER CR TO CONTINUE OR @ TO RETURN TO COMMAND MODE.
                        * THE PURPOSE OF THIS IS TO ALLOW THE USER TO VERIFY THE
                        * ADDRESS(ES) HE HAS TYPED IN, BEFORE THE SPECIFIED OPERATION
                        * BEGINS.
                        *
001 345   315 315 002   WAIT   CALL    INCH    GET CHARACTER FROM KEYBOARD
001 350   376 015              CPI     015     CARRIAGE RETURN?
001 352   310                  RZ              RETURN IF TRUE
001 353   376 100              CPI     100     @ ?
001 355   302 345 001          JNZ     WAIT      LOOP IF NOT
001 360   315 362 002          CALL    OUTCH   "ECHO" THE CHARACTER AND
001 363   303 050 000          JMP     BEGIN     GO BACK TO COMMAND MODE
                        *
                        * MEMORY DUMP
                        *
001 366   315 145 001   MEMORY CALL    ADDRIN
001 371   353                  XCHG            GET LOW ADDRESS RANGE IN D-E
001 372   315 145 001          CALL    ADDRIN  GET HIGH ADDRESS RANGE IN H-L
001 375   315 313 001          CALL    ADCOMP  CHECK D-E NOT > H-L
002 000   315 345 001          CALL    WAIT    USER ENTERS CR TO CONTINUEJ
                        *
                        * THE FOLLOWING ROUTINE MAY BE CALLED FROM A USER PROGRAM,
                        * USER MUST LOAD THE STARTING (LOWER) ADDRESS IN D-E AND THE
                        * ENDING (HIGHER) ADDRESS IN H-L.  DISPLAY OF MEMORY CONTENTS
                        * PROCEEDS FROM THE BEGINNING TO THE ENDING ADDRESSES, INCLUSIVE.
                        *
002 003   305           MDUMP  PUSH B          PUSH REGS, B AND C ON THE STACK
002 004   325                  PUSH D          PUSH REGS, D AND E ON THE STACK
002 005   315 346 002   M0010  CALL    OUTNEW  START A NEW LINE ON OUTPUT DEVICE
002 010   172                  MOV AD
002 011   315 100 001          CALL    OUT3    DISPLAY MEMORY PAGE NO.
002 014   076 054              MVI A   054
002 016   315 362 002          CALL    OUTCH   DISPLAY A COMMA (,)
002 021   173                  MOV AE
002 022   315 100 001          CALL    OUT3    DISPLAY MEMORY ADDRESS WITHIN PAGE
002 025   076 072              MVI A   072
002 027   315 362 002          CALL    OUTCH   DISPLAY A COLON (:)
002 032   006 020              MVI B   020     SET COUNTER TO DISPLAY 16
                        *                       BYTES (IN OCTAL) PER LINE.
002 034   315 360 002   M0020  CALL    OUTSP   SKIP ONE CHARACTER POSITION ON OUTPUT DEVICE
002 037   032                  LDAX D          LOAD MEMORY BYTE INTO ACCUM.
002 040   315 100 001          CALL    OUT3    DISPLAY 3 OCTAL DIGITS (ONE BYTE)
002 043   174                  MOV AH
002 044   272                  CMP D           COMPARE D-E WITH H-L,  IF D-E
002 045   332 073 002          JC      MEXIT     IS NOT LESS THAN H-L, MEMORY
002 050   302 063 002          JNZ     M0030     DUMP IS COMPLETE.
002 053   175                  MOV AL
002 054   273                  CMP E
002 055   332 073 002          JC      MEXIT
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T
 332   002 060   312 073 002          JZ      MEXIT
 333   002 063   023            M0030 INX D           INCREMENT LOW RANGE ADDRESS POINTER
 334   002 064   005                  DCR B           DECREMENT COUNTER
 335   002 065   302 034 002          JNZ     M0020     LOOP BACK - MORE SPACE AVAILABLE ON LINE
 336   002 070   303 005 002          JMP     M0010     LOOP BACK TO START NEW LINE
 337   002 073   321            MEXIT POP D           RESTORE D AND E
 338   002 074   301                  POP B           RESTORE B AND C
 339   002 075   311                  RET
 340                            *
 341                            * BINARY LOADER
 342                            *
 343   002 076   315 145 001    LOADER CALL    ADDRIN
 344   002 101   353                  XCHG            GET LOW ADDRESS RANGE IN REGS, D-E
 345   002 102   315 145 001          CALL    ADDRIN  GET HIGH ADDRESS RANGE IN REGS, H-L
 346   002 105   315 313 001          CALL    ADCOMP  CHECK D-E NOT > H-L
 347   002 110   072 376 002          LDA     MBOTH   MAKE SURE LOW ADDRESS IS
 348   002 113   075                  DCR A             NOT WITHIN MONITOR AREA!
 349   002 114   272                  CMP D
 350   002 115   322 273 002          JNC     VMON
 351   002 120   315 345 001          CALL    WAIT    USER PRESSES CR TO CONTINUE
 352   002 123   315 362 002          CALL    OUTCH   "ECHO" THE CHARACTER
 353                            *
 354                            * THE FOLLOWING PAPER TAPE INPUT ROUTINE MAY BE CALLED FROM A
 355                            * USER PROGRAM,  USER MUST LOAD THE STARTING (LOWER) ADDRESS
 356                            * IN REGS, D-E AND THE ENDING (HIGHER) ADDRESS IN REGS, H-L,
 357                            * DATA IS READ STARTING AT THE HIGHEST LOCATION AND WORKING BACKWARDS,
 358                            *
 359   002 126   345            PTINP PUSH H          PUSH REGS, H AND L ON THE STACK
 360   002 127   333 000        LD020 IN       000     INPUT STATUS CHANNEL
 361   002 131   346 201              ANI     201     DATA AVAILABLE?
 362   002 133   302 127 002          JNZ     LD020     JMP IF NOT
 363   002 136   333 001              IN      001     INPUT DATA CHANNEL
 364   002 140   376 000              CPI     000     ON LEADER?
 365   002 142   312 127 002          JZ      LD020     JMP IF TRUE
 366   002 145   167            LD030 MOV MA          MOVE BYTE TO MEMORY
 367   002 146   172                  MOV AD          DETERMINE IF WE ARE FINISHED
 368   002 147   274                  CMP H             D : H
 369   002 150   332 160 002          JC      LD040
 370   002 153   173                  MOV AE
 371   002 154   275                  CMP L             E : L
 372   002 155   322 174 002          JNC     LDEXIT
 373   002 160   053            LD040 DCX H           DECREMENT MEMORY ADDRESS
 374   002 161   333 000        LD050 IN       000     INPUT STATUS CHANNEL
 375   002 163   017                  RRC             DATA AVAILABLE?
 376   002 164   332 161 002          JC      LD050     JMP BACK IF NOT
 377   002 167   333 001              IN      001     INPUT DATA CHANNEL
 378   002 171   303 145 002          JMP     LD030
 379   002 174   341            LDEXIT POP H          RESTORE REGS, H AND L
 380   002 175   311                  RET
 381                            *
 382                            * ROUTINE TO STORE BINARY ON PAPER TAPE
 383                            *
 384   002 176   315 145 001    PUNCH CALL     ADDRIN
 385   002 201   353                  XCHG            GET LOW ADDRESS RANGE IN REGS, D AND E
 386   002 202   315 145 001          CALL    ADDRIN  GET HIGH ADDRESS RANGE IN REGS, H AND L
 387   002 205   315 313 001          CALL    ADCOMP  CHECK D-E NOT > H-L
 388   002 210   315 345 001          CALL    WAIT    USER PRESSES CR TO CONTINUE
```

```
002 213   315 346 002          CALL    OUTNEW  START A NEW LINE ON OUTPUT DEVICE
                             *
                             * THE FOLLOWING PAPER TAPE OUTPUT ROUTINE MAY BE CALLED FROM
                             * A USER PROGRAM.  USER MUST LOAD THE STARTING (LOWER) ADDRESS
                             * IN REGS. D AND E AND THE ENDING (HIGHER) ADDRESS IN REGS. H AND L.
                             * DATA IS OUTPUTTED STARTING WITH THE HIGHEST ADDRESS AND WORKING
                             * BACKWARDS.
                             *
002 216   305                PTOUT   PUSH B          SAVE REGS. B AND C ON THE STACK
002 217   345                        PUSH H          SAVE REGS. H AND L ON THE STACK
002 220   006 062                    MVI B   062     PUNCH OUT 5 INCHES OF LEADER (NULLS)
002 222   076 000                    MVI A   000
002 224   315 362 002        PCH020  CALL    OUTCH
002 227   005                        DCR B
002 230   302 224 002                JNZ     PCH020
                             *
002 233   176                PCH030  MOV AM          PUNCH OUT ALL DATA IN USER-SPECIFIED AREA.
002 234   315 362 002                CALL    OUTCH
002 237   172                        MOV AD
002 240   274                        CMP H           D:H
002 241   332 251 002                JC      PCH040
002 244   173                        MOV AE
002 245   275                        CMP L           E:L
002 246   322 255 002                JNC     PCH050
002 251   053                PCH040  DCX H
002 252   303 233 002                JMP     PCH030
                             *
002 255   006 036            PCH050  MVI B   036     PUNCH OUT 3 INCHES OF TRAILER (NULLS) -
002 257   076 000                    MVI A   000       SERVES TO "EJECT" DATA FOR REMOVAL FROM PUNCH.
002 261   315 362 002        PCH060  CALL    OUTCH
002 264   005                        DCR B
002 265   302 261 002                JNZ     PCH060
002 270   341                        POP H           RESTORE REGS. H AND L
002 271   301                        POP B           RESTORE REGS. B AND C
002 272   311                        RET
                             *
                             * ROUTINE TO PRINT AN ERROR MESSAGE IF AN
                             * OPERATION TRIES TO VIOLATE THE MONITOR AREA.
                             *
002 273   315 346 002        VMON    CALL    OUTNEW  START NEW LINE ON OUTPUT DEVICE
002 276   041 307 002                LXI H   VMSG3   ADDRESS OF MESSAGE
002 301   315 331 002                CALL    OUTSTR  DISPLAY THE ERROR MESSAGE
002 304   303 050 000                JMP     BEGIN   BACK TO COMMAND MODE
002 307   005 126            VMSG3   DW      005126  LENGTH OF MSG; AND V
002 311   040 115                    DW      040115  SP AND M
002 313   117 116                    DW      117116  O AND N
                             *
                             * KEYBOARD INPUT ROUTINE - MAY BE CALLED FROM A USER PROGRAM
                             *
002 315   333 000            INCH    IN      000     INPUT STATUS CHANNEL
002 317   346 201                    ANI     201     IS DATA AVAILABLE?
002 321   302 315 002                JNZ     INCH      JMP IF NOT
002 324   333 001                    IN      001     INPUT DATA CHANNEL
002 326   346 177                    ANI     177     ZERO OUT PARITY BIT
002 330   311                        RET
                             *
                             * OUTPUT ROUTINES MAY BE CALLED FROM A USER PROGRAM
                             *
```

```
STMT   PG-ADDR   OBJECT CODE   S O U R C E   S T A T E M E N T
 447                           * OUTSTR OUTPUTS A CHARACTER STRING UP TO 255 BYTES IN LENGTH;
 448                           * STRING TO BE OUTPUTTED MUST BE PRECEEDED BY A ONE-BYTE LENGTH FIELD.
 449                           *
 450   002 331   305           OUTSTR PUSH B          PUSH B AND C ON THE STACK
 451   002 332   106                  MOV BM          LENGTH FIELD TO COUNTER
 452   002 333   043           OUTSTS INX H           POINT TO NEXT CHARACTER
 453   002 334   176                  MOV AM          NEXT CHARACTER TO ACCUM.
 454   002 335   315 362 002          CALL    OUTCH   OUTPUT THE CHARACTER
 455   002 340   005                  DCR B           DECREMENT THE COUNTER
 456   002 341   302 333 002          JNZ     OUTSTS    LOOP BACK IF NOT FINISHED
 457   002 344   301                  POP B           RESTORE B AND C
 458   002 345   311                  RET
 459
 460   002 346   076 015       OUTNEW MVI A   015     STARTS A NEW LINE ON OUTPUT DEVICE
 461   002 350   315 362 002          CALL    OUTCH
 462   002 353   076 012              MVI A   012
 463   002 355   303 362 002          JMP     OUTCH
 464   002 360   076 040       OUTSP  MVI A   040     SKIPS ONE CHARACTER POSITION ON OUTPUT DEVICE.
 465   002 362   365           OUTCH  PUSH A
 466   002 363   333 000       OUTLUP IN      000     INPUT STATUS CHANNEL
 467   002 365   346 200              ANI     200     READY?
 468   002 367   302 363 002          JNZ     OUTLUP    JMP IF NOT
 469   002 372   361                  POP A
 470   002 373   323 001              OUT     001     OUTPUT DATA CHANNEL
 471   002 375   311                  RET
 472                           *
 473   002 376   003           MBOTH  DB      003     LOWEST MEMORY PAGE AVAILABLE TO USER.
 474   002 377   037           MTOPH  DB      037     HIGHEST MEMORY PAGE AVAILABLE TO USER;
 475                           *
 476                           * BOOTSTRAP LOADER FOR SYSTEM MONITOR VERSION 1.0
 477                           *   (TO BE LOADED FROM SYSTEM CONTROL PANEL).
 478                           * REQUIRES 42 BYTES OF MEMORY.
 479                           *
 480   003 320                        ORG     320003
 481   003 320   041 000 003          LXI H   000003  HIGHEST ADDR. + 1 OF MONITOR
 482   003 323   333 000       BLINP1 IN      000     INPUT STATUS CHANNEL
 483   003 325   346 201              ANI     201     IS DATA AVAILABLE?
 484   003 327   302 323 003          JNZ     BLINP1    JUMP IF NOT
 485   003 332   333 001              IN      001     INPUT DATA CHANNEL
 486   003 334   376 000              CPI     000     ON LEADER?
 487   003 336   312 323 003          JZ      BLINP1    JUMP IF TRUE
 488   003 341   053           BLMOVE DCX H           DECR. MEMORY ADDRESS
 489   003 342   167                  MOV MA          MOVE BYTE TO MEMORY
 490   003 343   076 000              MVI A   000
 491   003 345   275                  CMP L           HAS LOWEST ADDR. BEEN LOADED?
 492   003 346   332 356 003          JC      BLINP2    JUMP IF NOT
 493   003 351   274                  CMP H
 494   003 352   332 356 003          JC      BLINP2
 495   003 355   351                  PCHL            JUMP TO MONITOR
 496   003 356   333 000       BLINP2 IN      000     INPUT STATUS CHANNEL
 497   003 360   346 001              ANI     001     IS DATA AVAILABLE?
 498   003 362   302 356 003          JNZ     BLINP2    JUMP IF NOT
 499   003 365   333 001              IN      001     INPUT DATA CHANNEL
 500   003 367   303 341 003          JMP     BLMOVE

TOTAL NUMBER OF STATEMENTS = 0500    ERROR STATEMENTS = 0000
```

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

U.S. Subscriptions:

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

For foreign subscriptions:

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

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

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

Necessary Information:

Name (last name first) _____

Mailing Address _____

_____

City _____ State _____ Zip Code _____

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

Optional Information:

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

Make & model _____ Manufacturer _____

CPU model _____ CPU Manufacturer _____

I/O Devices _____

Mass storage peripherals _____

_____

_____

_____

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

_____

_____

_____

_____

_____

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

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

OVER

Directories of:

_____ Users of home computers and their equipment
_____ Computer stores and distributers
_____ Manufacturers of computer kits
_____ Computer clubs
_____ Sources of used equipment
_____ Microprocessor and minicomputer manufacturers

Source code listings and documentation: For which microprocessors? _____

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

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

Importance Rating	Software Description

_____
_____
_____

Place
13-cent
stamp
here

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

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

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

BOOKSTORE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

101 BASIC COMPUTER GAMES
David Ahl, editor. 1974. 250 pp. $7.50.

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

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

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

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

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

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

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

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

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

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

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

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

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

WHOLE EARTH EPILOG
Stewart Brand, editor. 1974. 318 pp. $4.

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

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

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

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