AUGUST 1979 • Volume 4, Number 8 $2.00 in USA/$2.40 in Canada

# BYTE®

the small systems journal

A McGRAW-HILL PUBLICATION


LISP

# SYSTEMS – SOLUTIONS

If you have a problem that can be solved by a computer—we have a systems solution.

- Two central processors with maximum RAM capacities of 56K and 384 K bytes
- Three types of disk drives with capacities of 175K, 1.2M and 16M bytes
- Two dot matrix printers with 80 and 132 line capacity
- A Selectric typewriter interface and a daisy wheel printer

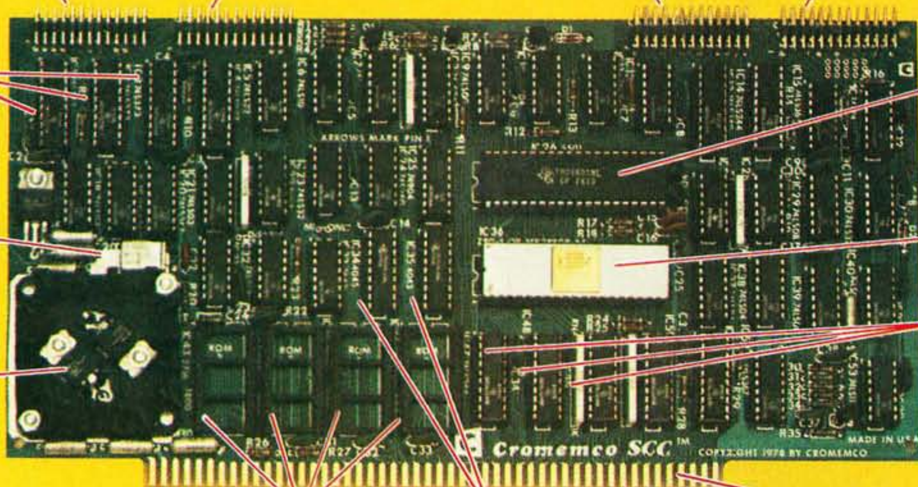
Match these to your exact need, add one or more of our intelligent terminals and put together a system from one source with guaranteed compatibility in both software and hardware.

Southwest Technical Products systems give you unmatched power, speed and versatility. They are packaged in custom designed woodgrain finished cabinets. Factory service and support on the entire system and local service is available in many cities.

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Circle 350 on inquiry card.




# The single card computer with the features that help you in real life

### COMPLETE COMPUTER

In this advanced card you get a professional quality computer that meets today's engineering needs. And it's one that's complete. It lets you be up and running fast. All you need is a power supply and your ROM software.

The computer itself is super. Fast 4 MHz operation. Capacity for 8K bytes of ROM (uses 2716 PROMs which **can be programmed by our new 32K** BYTESAVER® PROM card). There's also 1K of on-board static RAM. Further, you get straightforward interfacing through an RS-232 serial interface with ultra-fast speed of up to 76,800 baud — software programmable.

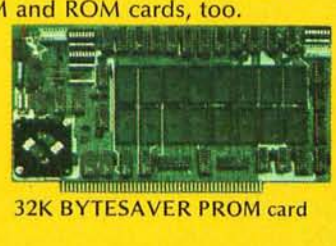
Other features include 24 bits of bidirectional parallel I/O and five on-board programmable timers.

Add to that vectored interrupts.

### ENORMOUS EXPANDABILITY

Besides all these features the Cromemco single card computer gives you enormous expandability if you ever need it. And it's easy to expand. First, you can expand with the new Cromemco 32K BYTESAVER PROM card mentioned above. Then there's Cromemco's broad line of S100-bus-compatible memory and I/O interface cards. Cards with features such as relay interface, analog interface, graphics interface, opto-isolator input, and A/D and D/A conversion. RAM and ROM cards, too.


Card Cage


32K BYTESAVER PROM card

### EASY TO USE

Another convenience that makes the Model SCC computer easy to use is our Z-80 monitor and 3K Control BASIC (in two ROMs). With this optional software you're ready to go. The monitor gives you 12 commands. The BASIC, with 36 commands/functions, will directly access I/O ports and memory locations — and call machine language subroutines.

Finally, to simplify things to the ultimate, we even have convenient card cages. Rugged card cages. They hold cards firmly. No jiggling out of sockets.

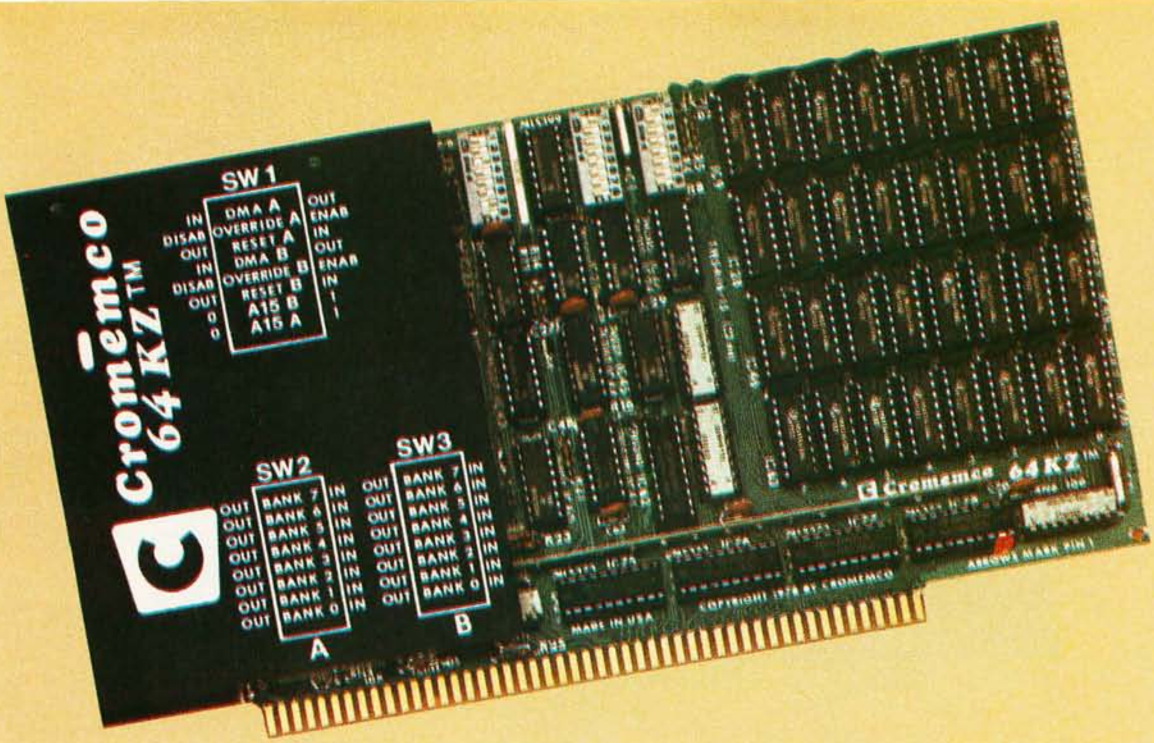
### AVAILABLE NOW/LOW PRICE

The Cromemco Model SCC is available now at a low price of only $450 factory assembled ($395 kit).

So act today. Get this high-capability computer working for you right away.

**Cromemco**
incorporated
Specialists in computers and peripherals
280 BERNARDO AVE., MOUNTAIN VIEW, CA 94040 • (415) 964-7400

Circle 80 on inquiry card.




# You can do surprising things when you have 64 kilobytes of fast RAM on one card

**4 MHz FAST—AND EXPANDABLE**

Here's 64 kilobytes of memory on one RAM card. Yes, we mean 512K bits of read/write memory on this single card.

And, yes, we mean it's fast. With 150-nanosecond chip access times — so the card can operate in fast Z-80 systems with no wait states. Repeat, no wait states.

**EXPANDABLE ON TWO LEVELS**

Not only does the new Model 64KZ give you a large, fast RAM but it is expandable on two levels.

First, through our Cromemco Bank Select feature, you can expand to 512 kilobytes in eight 64K banks.

Or, with our Extended Bank Select feature, you can expand memory space to as much as 16 megabytes.

This expandability we call your obsolescence insurance.

The legend on the card's heat sink is an easy reference for address and bank selection.

**BENCHMARK IT**

Obviously, the speed and memory capacity of this new card give you a lot of power.

You can see that for yourself in our new 7-station Multi-User Computer System which uses these Model 64KZ cards. This S100-bus system outperforms the speed of many if not most timesharing systems of up to 10 times the Cromemco price.

And yet where some of these much more expensive and cumbersome systems clearly slow to a snail's pace when timesharing, the Cromemco system using Bank Select switching runs surprisingly fast.

**SEE IT NOW**

See the new Model 64KZ at your computer dealer now. Study the literature on it. See how for only $1785 you can get around that ever-present barrier of memory that's too little and too slow.

For high reliability all Cromemco memory cards are burned in at the factory in these temperature-controlled ovens.

Cromemco Multi-User System shown with 7 stations

**Cromemco**
incorporated
280 BERNARDO AVE., MOUNTAIN VIEW, CA 94040 • (415) 964-7400
Tomorrow's computers now



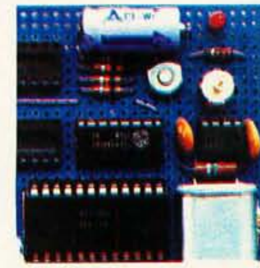
Circle 80 on inquiry card.

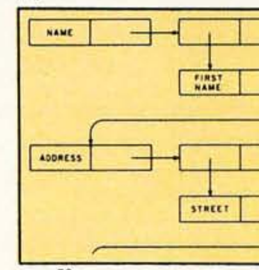
BYTE August 1979
Volume 4, Number 8

# In the Queue

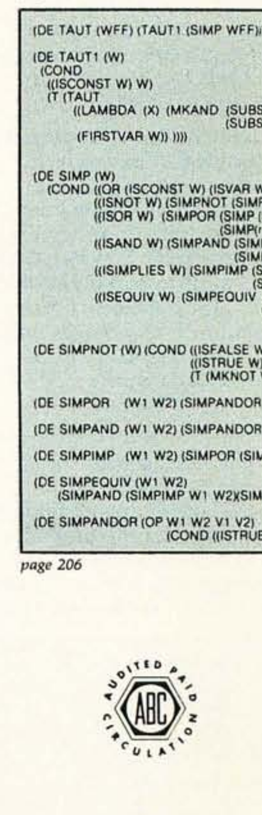
## Foreground




*page 18*



*page 50*



*page 206*

## Background



## Nucleus



BYTE is published monthly by BYTE Publications Inc, 70 Main St, Peterborough NH 03458. A wholly-owned subsidiary of McGraw-Hill, Inc. Address all mail except subscriptions to above address: phone (603) 924-7217. Address subscriptions, change of address, USPS Form 3579, and fulfillment questions to BYTE Subscriptions, PO Box 590, Martinsville NJ 08836. Second class postage paid at Peterborough NH 03458 and at additional mailing offices—USPS Publication No. 102410 (ISSN 0360-5280). Subscriptions are $18 for one year, $32 for two years, and $46 for three years in the USA and its possessions. In Canada and Mexico, $20 for one year, $36 for two years, $52 for three years. $32 for one year air delivery to Europe. $32 surface delivery elsewhere. Air delivery to selected areas at additional rates upon request. Single copy price is $2 in the USA and its possessions, $2.40 in Canada and Mexico, $3.50 in Europe, and $4 elsewhere. Foreign subscriptions and sales should be remitted in United States funds drawn on a US bank. Printed in United States of America.

Address all editorial correspondence to the editor at the above address. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE. 

BYTE® is available in microform from University Microfilms International, 300 N Zeeb Rd, Dept PR, Ann Arbor MI 48106 USA or 18 Bedford Row, Dept PR, London WC1R 4EJ ENGLAND.

Subscription WATS Line: (800) 258-5485

Office hours: Mon-Thur 8:30 AM — 4:30 PM
Friday 8:30 AM — Noon

# In This BYTE

**About the Cover**

*This month, Ken Lodding has created a fantasy on far-out applications with a LISP theme. The surface of some asteroid has been discovered. A monolith engraved with the S-expression form of a LISP program is gazed upon by some astronauts. We presume some archeology of this monolith will have to be done to uncover the balance of the program. We leave it to readers familiar with LISP to identify the textbook from which these S-expression fragments were taken, and the purpose of the program.*




**Publishers**
Virginia Londoner
Gordon R Williamson
**Associate Publisher**
John E Hayes
**Assistant**
Jill E Callihan

**Editorial Director**
Carl T Helmers Jr
**Executive Editor**
Christopher P Morgan
**Editor in Chief**
Raymond G A Cote
**Senior Editor**
Blaise W Liffick
**Editor**
Richard Shuford
**Assistant Editor**
Kent Richard
**Editorial Assistants**
Gale Britton
Faith Ferry
**New Products Editor**
**Clubs, Newsletters**
Laura A Hanson
**Drafting**
Jon Swanson

**Production Editors**
David William Hayward
Ann Graves
Faith Hanson
Warren Williamson
**Art Director**
Ellen Bingham
**Production Art**
Wai Chiu Li
Christine Dixon
Holly Carmen LaBossiere
Deborah Porter

**Advertising Director**
Patricia E Burgess
**Assistants**
Ruth M Walsh
Marion Gagnon
Janet Ames
**Adv/Prod Coordinator**
Thomas Harvey
**Advertising Billing**
Noreen Bardsley
Don Bardsley

**Traffic Department**
Mark Sandagata
Thomas Yanni

**Circulation Manager**
Gregory Spitzfaden
**Assistants**
Pamela R Heaslip
Agnes E Perry
Melanie Bertoni
Barbara Ellis
**Dealer Sales**
Ginnie F Boudrieau
Anne M Baldwin
**Receptionist**
Jacqueline Earnshaw

**Typographers**
Cheryl A Hurd
Stephen Kruse
Debe L Wheeler
**Photostat Technician**
Tully Londner

**Comptroller**
Kevin Maguire
**Assistant**
Mary E Fluhr

**National Advertising Sales Representatives:**
Hajar Associates Inc
**East**
280 Hillside Av
Needham Heights MA 02194
(617) 444-3946
521 Fifth Av
New York NY 10017
(212) 682-5844
**Midwest**
664 N Michigan Av
Suite 1010
Chicago IL 60611
(312) 337-8008
**West, Southwest**
1000 Elwell Ct
Suite 227
Palo Alto CA 94303
(415) 964-0706/(714) 540-3554

Officers of McGraw-Hill Publications Company: Gordon L. Jones, President; Group Vice Presidents: Daniel A. McMillan, James E. Boddorf; Senior Vice Presidents: Russell F. Anderson; Ralph R. Schulz, Editorial; Vice Presidents: James E. Hackett, Controller; Thomas H. King, Manufacturing; Robert L. Leyburn, Circulation; John W. Patten, Sales; Edward E. Schirmer, International.

Officers of the Corporation: Harold W. McGraw Jr., President, Chief Executive Officer and Chairman of the Board; Robert F. Landes, Senior Vice President and Secretary; Ralph J. Webb, Treasurer.

# Editorial

## Returning to the Tower of Babel, or... Some Notes About LISP, Languages and Other Topics...

*by Carl Helmers*

This is the August issue of BYTE. It is also the third consecutive year that we've chosen to have a computer language as an issue content theme—a choice which is reflected in a number of articles, as well as the cover painting by Ken Lodding.

In the past two years, the August issues have had themes of APL (1977) and Pascal (1978). This year, we continue the August emphasis on languages with a special issue devoted to the language LISP. An experiment in editorial policy is also reflected in this issue. John Allen was responsible for the solicitation and technical reviewing of the articles concerning LISP in this issue, truly functioning in the capacity of "Guest Editor" of BYTE. John has been involved with computation research involving LISP for some time, and he is in touch with many of the members of the artificial intelligence community. Some of his comments on LISP appeared in the March 1979 issue of BYTE in the form of a guest editorial. As a result of his earlier writings about LISP as an appropriate tool of expression for personal computing, we asked him to take charge of the LISP oriented technical content of this issue and several issues to follow. Readers will find a wealth of information as a result of John's efforts.

By making LISP a feature of this issue of BYTE, we are emphasizing the history of LISP's utility in artificial intelligence and computation research. The language is derived from the work of John McCarthy in the early 1960's. LISP will have its place in personal computing, alongside a number of other styles of expression. For lack of appropriate systems software, I have not personally used LISP to any extent, but I believe that I have the beginnings of an abstract appreciation of its potential. This perspective comes from personal contact with individuals who use LISP regularly, as well as reading which includes the articles in this issue as collected by John Allen.

In a recent (May 24 1979) conversation with Gary Kildall on the occasion of the fifth IEEE Computer Society Asilomar Conference on Microcomputing, I mentioned the LISP issue. Gary has a background in computer systems software work with special emphasis on small scale computer systems of the kind used by BYTE readers. He is the first implementor of the PL/M compilers for Intel's 8080 microprocessors, and he and his firm, Digital Research, are responsible for one of the most widely used 8080 and Z-80 oriented software products, the CP/M operating system. I learned some interesting points from Gary about LISP and its significance to the use of computers, viewpoints which are worth repeating for readers.

Gary made the statement that LISP is basically his preferred language. He explained that LISP has a certain natural elegance, but that people often tend to write FORTRAN or BASIC-like sequential "PROGs" as opposed to the implicitly parallel and recursive tree structures natural to LISP. He emphasized that this is a mistake. LISP represents a different point of view from which to analyze problems.

*Text continued on page 154*

# "My 8 to 5 minifloppy™ now works nights and weekends."


"I own a fast-growing business and before I bought my computer system I put in a lot of late hours keeping up with my accounting and inventory control. Now the computer does my number crunching quickly, so I have time after hours to have some fun with the system. My son and I started out playing Star Trek on the system, and now we're learning to play chess.

"When I was shopping around for my system, the guys in the computer stores demonstrated all the unique features of the minifloppy. I've got to admit that at first I didn't really understand all the technical details. But now that I use the system every day, I really appreciate the minifloppy's fast random access and data transfer. I like the reliability, too.

"I'm glad I went with Shugart drives. Look, when you lay out your own money for a system, you want dependable performance and good value. Do what I did. Ask for the system with the minifloppy."

## If it isn't Shugart, it isn't minifloppy.

Shugart Associates

435 Oakmead Parkway, Sunnyvale, California 94086

**See opposite page for list of manufacturers featuring Shugart's minifloppy in their systems.**

**TM minifloppy is a registered trademark of Shugart Associates**

# Letters

## Puzzling Rotation Explained

Ken Barbier poses a question in "Puzzling Rotation" (May 1979 BYTE, page 216) which is intimately related to my comments on periodic decimal expansions in that same issue (page 210).

Any number N which has a repeating, periodic decimal expansion of 1/N with maximum period length (N−1) gives rise to a *magic* number X = INT((1/N)*10(N−1)). As he pointed out, any multiple of X such as K×X (with K less than N) contains the same digits as does X, but cyclically rotated. N=7 is the only example in base 10 arithmetic less than 10; larger values of N are, for example, 17 (yielding X=0588235294117647) and 19 (which gives X=052631578947368421). In base 8, some interesting numbers are given by N=5 (X=1463, base 8) and N=11 (base 10) (X=0564272135 base 8); in base 15, a *magic* X is 124936DCA5B8.

I have not been able to find any *magic* numbers in base 4, base 16, or base 64; perhaps some reader can prove that none exists for bases which are powers of 4.

If the length of the repetition period of 1/N is shorter than the maximum, then the *magic* number X generated by the above algorithm will still re-appear with digits cyclically permuted, but other numbers also appear in the course of the multiplication. Try, for example, N=13, X=076923, in base 10.

For some insight into why these numbers are *magic*, you might want to try calculating by hand, long-division-style, some examples like 1/7, 2/7, 3/7, etc. According to E T Bell's biographical book *Men of Mathematics* (page 225), one of the greatest mathematicians of all time, Carl Friedrich Gauss, worked out the decimal expansions of 1/N for all N up to 1000 while he was a teen-ager. (And in the 1790's, he didn't have a home computer!) The results of his calculations inspired him to discover and prove one of the most beautiful theorems of number theory, "quadratic reciprocity." Playing games with numbers is still a fine route to inspiration. Good luck!

**Mark Zimmermann**
**Caltech 130-33**
**Pasadena CA 91125**

## More Puzzling

Regarding "An Added Attraction" (Machine Language Puzzler May 1979 BYTE, page 209), I would like to share a twist on the problem of adding two 8 bit values in registers B and C and my solution.

First, let me admit that when I glanced through the puzzle rules, I mistakenly assumed that all subtraction operations, as well as the addition operations, were prohibited in the solution. The reason I made this slip is that the problem now becomes a little harder (something akin to the business of multiplying using addition instructions only).

Anyway, my first brute force attempt at this different problem required 12 bytes:

```
        XRA   A
LOOP1   INR   A
        DCR   B
        JNZ   LOOP1
LOOP2   INR   A
        DCR   A
        JNZ   LOOP2
        HLT
```

This works by initializing a counter using the byte-saving exclusive-or operation. The counter is then incremented once for each time that register B must be decremented, until the register reaches zero. Repeating this sequence using register C results with the sum in the accumulator. Of course, this approach ignores overflow detection, as did the original solutions published in BYTE.

Being dissatisfied with the above, I noticed a much simpler solution in 7 bytes:

```
        MOV   A,B
LOOP    INR   A
        DCR   C
        JNZ   LOOP
        HLT
```

Interestingly, this is only 2 bytes more than the optimum solution presented in the Puzzler, where subtraction is permitted.

**Steve Duerksen**
**Microcomputer Consultant**
**15 Dearborn St**
**Wellesley MA 02181**

## Attention: Gamblers

A newsletter is being started for computer enthusiasts interested in analyzing gambling systems, the Stock and Futures Markets, etc. The first issue will be priced at $1 and those interested should indicate preference for form, content, and subscription rate. Contact Michael R Downing, c/o Joe Computer, 22713 Ventura Blvd, Suite F, Woodland Hills CA 91364.■

# BEFORE YOU BUY COMPUTER #1, VISIT #1 COMPUTERLAND

If the truth is that you want a computer . . . then we want to be your computer store.

We're ComputerLand, the #1 computer store chain in the U.S. What's meaningful about that fact is, that ComputerLand has been chosen by more people as having what they've been looking for. And, since you're looking, let us tell you what you'll find, when you visit a ComputerLand store.

You'll find a product line that's continually evaluated to provide you with the widest and best selection in quality, brand name microcomputers anywhere. You'll find an enthusiastic and knowledgeable staff able to interpret all the equipment specifications, in terms of how they apply to you, and in a way you'll understand. You'll find demonstration areas where you can get a firsthand experience of running a computer yourself.

## COMPUTERS FOR BUSINESS

You'll find educational materials to give you a total insight into the world of microcomputers.

You'll find a fully equipped service department to provide whatever assistance is required to keep your computer running in top-notch condition. You'll find computer user's clubs to join, where you can share ideas with people as enthusiastic as yourself. And, with each new visit, you'll find excitement—from the people you deal with, the equipment they offer, and from your own ever-growing personal involvement.

Enough about us. How about what computers do. To attempt to describe all the things your computer might do, would be to describe your imagination. So instead, we'll briefly list some of the many things for which small computers are already being used.

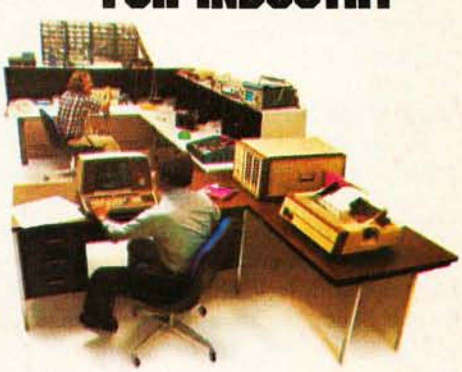
**In business,** the advent of the versatile and compact microcomputer has put the benefits of computing within reach of small companies. With systems starting at less than $6000, the businessman can computerize things like accounting, inventory control, record keeping, word processing and more. The net result is the reduction of administrative overhead and the improvement of efficiency which allows the business to be managed more effectively.

## COMPUTERS FOR THE HOME

**In the home,** a computer can be used for personal budgeting, tracking the stock market, evaluating investment opportunities, controlling heating to conserve energy, running security alarm systems, automating the garden's watering, storing recipes, designing challenging games, tutoring the children . . . and the list goes on.

**In industry,** the basic applications are in engineering development, process control, and scientific and analytical work. Users of microcomputers in industry have found them to be reliable, cost-effective tools which provide computing capability to many who would otherwise have to wait for time on a big computer, or work with no computer at all.

## COMPUTERS FOR INDUSTRY



And now we come to you, which leads us right back to where we started: **If you want a computer, then we want to be your computer store.**

Whether you want a computer for the home, business or industry, come to ComputerLand first. We'll make it easy for you to own your first computer. Because, simply put, we really want your business. When you come right down to it, **that's what makes us #1.**

ComputerLand Corp.
14400 Catalina St.
San Leandro, CA 94577
(415) 895-9363
Franchise Opportunities Worldwide.

**ComputerLand**®
WE KNOW SMALL COMPUTERS

ComputerLand Europe
Europa Terrassen
8 Rue Jean Engling
Dommeldange, Luxembourg
Phone 43 29 05 Telex 2423

# An Overview of LISP

John Allen
Signetics
811 E Acques Ave
Mail Stop 38
Sunnyvale CA 94086

## LISP is a higher level machine language.

LISP is simple and difficult, elegant and ad hoc; it is a beautiful blend of foresight and fortuity. LISP is a programming language, often characterized as a special purpose list-processing language. But LISP is no more a special purpose programming language than mathematics is a special purpose language for floating-point computations. Just as there's more to mathematics than the accounting and bookkeeping properties present in "general purpose" programming languages, there's much more to LISP than "just another programming language."

The best description of the LISP programming language is that it is a *high level machine language.* That is, it shares many of the facets of contemporary machine language —the necessity for attention to detail and the freedom to manipulate the machine's data and programs without restriction— yet LISP is high level in that the language contains the expressive power and convenience of traditional high level languages. The *contradiction* is resolvable: a LISP machine is just a higher level machine whose data items are organized differently from the binary bit patterns of most machines, and the LISP programming language is the *assembly language* for this machine.

### LISP Data Structures

Before introducing the constructs of the language, we must discuss the data items of the language. In a traditional language we would find numeric constants. In LISP, the analogous constants are called *atoms.* An atom is either a numeral or a *literal atom* —a string of upper case alphanumeric characters such that the first character in the string is an alphabetic character. For example, *ABC123, 12,* and *NIL* are atoms, but *1A2* and *(A B)* are not.

LISP also has composite constants called *lists.* Lists are built out of atoms and other lists as follows:

- Any atom or list can be an element of a list.
- Given any collection $e_1$, ..., $e_n$ of list elements, then ($e_1$ ... $e_n$) is also a list.

So, *(A B)* is a list; as is *(A B C),* and *(A 1 (ABC 23)).* The last example is a list of three elements; its third element is also a list — of two elements: the atom *ABC* and the numeral *23.*

Atoms and lists are the basic LISP data structures. However, a robust production version of LISP includes many more data objects including arrays, arbitrary precision numbers, strings, and representation of functions as data objects. Regardless of the scope of the data representations in a specific LISP implementation, it is a fundamental property that all data objects are "first class objects," constructible, testable and available without restriction. This uniform behavior of data is a property shared by few other languages.

### First

We need some operations on these data structures. Just as we should have a subtraction operation in arithmetic machines to decompose numbers, we have LISP instructions to decompose lists. One such operation is *first;* it extracts the first element of a list. For example:

*first[(A B C)]* *gives: A*

This example is written in LISP's *external syntax* called meta-LISP or M-LISP; it is an instance of prefix notation. The programming language, the *internal notation,* is called S-expression LISP or S-LISP. Initially, we will present algorithms in M-LISP since it is closer to traditional programming notation. However, since S-LISP is our machine language we will insist on developing facility with that notation.

In a traditional architecture, both instructions and data are stored in memory. The processor usually has complete freedom to manipulate any of these objects as either data or instructions. An object accessed by the instruction counter is interpreted as an instruction; other accesses to items usually imply a data interpretation. One goal is the representation of LISP instructions as data items in the LISP machine such that the processing unit of the LISP machine will have equal flexibility in interpreting the encoded information. An object may sometimes play the role of program, and sometimes of data.

To represent program as data we must specify a translation of each M-LISP instruction into a list representation:

**External Notation**

<operation>[<operand>$_1$; ... ;<operand>$_n$]

**List Notation**

(<operation>$^T$<operand>$_1^T$... <operand>$_n^T$)

**About the Author**

*John Allen, our guest editor for this special LISP theme issue, is the author of the book* Anatomy of LISP *and currently product engineer at Signetics Corporation. He is also founder of The LISP Company, an organization to produce LISP related products.*

The raised T means perform the translation process recursively.

For this translation to be meaningful, we must also describe how the recursion process is to terminate:

An *operation* in external notation is something like *first* or +, whereas an *operation*$^T$ must be an atom or a list. We translate the operation name to an appropriate atom: *first* translates to *FIRST*, and + to *PLUS*.

The operand of *first[(A B C)]* is the constant *(A B C)*. We will translate a constant $\alpha$ to the construct *(QUOTE* $\alpha$*)*. For example, we represent the constant *(A B)* as *(QUOTE(A B))*. This solution is similar to the *quoting* convention of natural language: Cleveland is a city, but "Cleveland" is a 9-letter word. The *QUOTE* operator is more than simple pedantry; it will play a critical role in the *fetch* operation of the LISP machine.

To summarize, our list notation consists of a representation of the operation followed by the representations of the operands. Those operands themselves may specify operations, or they may specify constant operands by using the *quote* operation. For example, we represent *first[(A B C)]* as *(FIRST (QUOTE (A B C)))* and *(FIRST (FIRST (QUOTE ((A B) C))))* represents *first[first[((A B) C)]]*.

Values are obtained on a LISP machine in much the same manner as one obtains values from a pocket calculator. We type in an S-LISP expression, and the calculator displays the result. The evaluation of an expression can be quite involved. If an operand specifies a further operation, then the current instruction must be suspended while that subsidiary computation is performed. So, evaluating *(FIRST (FIRST (QUOTE ((A B) C))))* would involve the following:

The leftmost *FIRST* must wait since its operand requires evaluation; similarly the next *FIRST* must wait to take care of its argument. But its argument is a quoted expression. *QUOTE* is kind, requiring no further computation, but it always returns its argument as value. Here it returns the list *((A B) C)*. The inner *FIRST* completes now, returning *(A B)* to the outermost *FIRST*; it is nudged into activity and finally returns *A*.

Consider *(FIRST (QUOTE (FIRST (QUOTE (A B)))))*. Notice that the embedded expression *(FIRST (QUOTE (A B)))* has the appearance of a LISP instruction. However, that expression is surrounded by *(QUOTE ... )*, therefore it is simply a list; ie, a constant. The final result of the evaluation will be the atom *FIRST* (since the computation encodes the M-expression *first[(FIRST (QUOTE (A B)))]* ).

Since quoted expressions appear so frequently, we will introduce an abbreviation. We write *(QUOTE* $\alpha$*)* as *'*$\alpha$. So, the previous example *(FIRST (QUOTE (FIRST (QUOTE (A B)))))* could be expressed as: *(FIRST '(FIRST (QUOTE (A B))))*; or as *(FIRST '(FIRST '(A B)))*. This abbreviation will appear many times throughout the LISP articles in this and following issues.

**Rest**

We also have an instruction named *REST*. You may think of the instruction as either a machine operation or as the translation of an M-LISP expression. *REST*, like *FIRST*, expects a list as its argument. However, *REST* returns a value representing the list with the first element removed. The expression:

*(REST '(A B C))*

yields:

*(B C)*.

Similarly, the expression:

*(REST '(B C))*

yields:

*(C)*

What about *(REST '(C))*? When we remove the last element from a list we get the empty list. Its representation in LISP is *( )*.

The operations *first* and *rest* are called *selector* functions since they are used to select components from a composite data object.

# The Intecolor 8070 business system. At twice the price, it would still be inexpensive.

Because ISC is the world's leading manufacturer of color terminals, we're able to offer unparalleled color performance—at phenomenal prices.

Our 8070 Series I Business System is a perfect example of reliable, yet extremely reasonable products. It's a complete 8080A microcomputer system that includes the following standard features: A dual floppy disk drive with 591K bytes of storage; a 19" color data display with an easily readable 80 characters x 48 line format; and a 60 CPS Impact Matrix printer. All for a remarkable *$7000, single unit price.

And that price doesn't stop at hardware. Intecolor's 18K Business BASIC in ROM has 16 digit accuracy and a PRINT USING feature that tailors output to any specifications.

If you need greater storage capabilities, choose the 8071—same system, but with a dual double-headed floppy disk drive. Giving you 1182K bytes of storage, for only *$800 more, single unit.

Either way, the Intecolor Series I can relieve your business of a considerable amount of paperwork, without costing you a great deal of money.

(Terms—5% discount for prepayment, or net 20 days.)

For more information and a complete demonstration, see your nearest computer store, or contact your ISC sales representative.

**Color Communicates Better**


Unretouched photo of screen

Furniture not included

*U.S. domestic prices

**ISC SALES REPRESENTATIVES: AL:** 205/883-8660, **AK:** (GA) 404/449-5961, **AZ:** 602/994-5400, **AR:** (TX) 214/840-2169, **CA:** Alhambra 213/281-2280, Goleta 805/964-8751, Irvine 714/557-4460, Los Angeles 213/476-1241, Mountain View 415/964-9300, San Diego 714/292-8525, **CO:** 303/759-0809, **CT:** (GA) 404/449-5961, **DE:** (GA) 404/449-5961, **DC:** (VA) 703/569-1502, **FL:** Orlando 305/425-5505, Ft. Lauderdale 305/776-4800, Melbourne 305/723-0766, Tallahassee 904/878-6642, **GA:** 404/455-1035, **HI:** 808/524-8633, **ID:** (UT) 801/973-7969, **IL:** (No.) 312/564-5440, (So.) (MO) 816/765-3337, **IN:** (IL) 312/564-5440, **IA:** (MO) 816/765-3337, **KS:** (MO) 816/765-3337, **KY:** 606/273-3771, **LA:** 504/626-9701, **ME:** (GA) 404/449-5961, **MD:** (VA) 703/569-1502, **MA:** (GA) 404/449-5961, **MI:** 313/227-7067, **MN:** 612/822-2119, **MS:** (AL) 205/883-8660, **MO:** 816/765-3337, **MT:** (CO) 303/759-0809, **NB:** (MO) 816/765-3337, **NH:** (GA) 404/449-5961, **NJ:** (GA) 404/449-5961, **NV:** (AZ) 602/994-5400, **NM:** 505/292-1212, **NY:** (GA) 404/449-5961, **NC:** 919/682-2383, **ND:** (MN) 612/822-2119, **OH:** Dayton 513/429-9040, Cleveland 216/464-8113, Columbus 614/436-2051, **OK:** (TX) 214/840-2169, **OR:** 503/620-5800, **PA:** 412/922-5110, **RI:** (GA) 404/449-5961, **SC:** 803/798-8070, **SD:** (MN) 612/822-2119, **TN:** 615/482-5761, **TX:** 214/840-2169, El Paso Area (Las Cruces, NM) 505/523-0601, Houston Only 713/780-2511, **UT:** 801/973-7969, **VT:** (GA) 404/449-5961, **VA:** 703/569-1502, **WA:** 206/455-9180, **WV:** 412/922-5110, **WI:** (IL) 312/564-5440, **WY:** (CO) 303/759-0809.
**EUROPEAN EXPORT SALES: EUROPE:** (MA) 617/661-9424, **BELGIUM:** Brussels 02-242 36-04, **FRANCE:** Rueil Malmaison 749-40-37, **GREECE:** Athens 642-1368, **ITALY:** Roma 805-647/872-457, **THE NETHERLANDS:** Poeldijk 01749-7640, **SPAIN:** Barcelona 204 17 43, **SWEDEN:** Vallingby 08-380-370, **SWITZERLAND:** Mutschellen 057-54655, **UNITED KINGDOM:** Bournemouth 0202-293-115, **WEST GERMANY:** München 089-31881, **AUSTRALIA & NEW ZEALAND:** Melbourne 543-2077, Meadowbank 808-1444, Chermside 59-6436, Wellington 64-4585, Auckland 876-570, **CANADA:** Datamex, Ltd. (Distributor) Dorval 514/636-9774, Ottawa 613/224-1391, Toronto 416/787-1208, Vancouver 604/684-8625, **CENTRAL & SOUTH AMERICA & CARIBBEAN:** (GA) 404/394-9603, **MEXICO:** Monterrey 564-876, **FAR EAST:** (CA) 213/382-1107, **HONG KONG:** 5-742211, **JAPAN:** Tokyo 402-8596, **TAIWAN:** Taipei 02-7026284, **MIDDLE EAST: IRAN:** Tehran 891148, **ISRAEL:** Tel Aviv 266-291, **KUWAIT:** Kuwait 438 180/1/2, **LEBANON:** Beirut 221731 260110, **SAUDI ARABIA:** Jeddah 27790, Ryadh 25083-39732, **UNITED ARAB EMIRATES:** Sharjah 24068.
For sales and service in other countries contact ISC headquarters in Norcross, GA., U.S.A.

**Intelligent Systems Corp.** □ 5965 Peachtree Corners East □ Norcross, GA 30071 □ Telephone 404-449-5961 □ TWX 810-766-1581

Circle 176 on inquiry card.

## An operation which builds new objects is a constructor.

### List

Besides decomposing objects, we must be able to build new objects. The general name for an operation which builds new objects is a *constructor*. One LISP constructor is *LIST*. Here are some examples of usage:

*(LIST 'A 'B 'C)*

yields:

*(A B C).*

*(LIST 2 'B)*

yields:

*(2 B)*

Note that we did not quote the 2. LISP understands that numbers are constants. Also, the *LIST* operation will take an arbitrary number of operands; three in our first example, two in this one, and none in the next:

*(LIST)*

yields:

*( ).*



As with the other instructions, except *QUOTE, LIST* can handle instructions as operands.

Try to determine the result of:

*(LIST (FIRST (QUOTE (A)))*
*(REST (QUOTE (A B))) (QUOTE C)).*

Diligence may have been rewarded and you may have responded *(A (B) C)*. There's an equal probability that you got mired in parenthesis-counting and responded *( ? $ f i)*. One solution is to resort to M-LISP and recast the expression as: *list[first[(A)];rest[(A B)];C]*

Since we should develop our S-LISP expertise, we might also use our abbreviation: *(LIST (FIRST '(A)) (REST '(A B)) 'C).*

A more general technique is *pretty-printing*. Pretty-printing exploits additional lines and spaces to highlight the structure in complex expressions. For example:

```
(LIST (FIRST (QUOTE (A)))
      (REST (QUOTE (A B)))
      (QUOTE C))
```

or:

```
(LIST (FIRST '(A))
      (REST '(A B))
      'C)
```

In a modern LISP implementation we would find further aids for locating matching parentheses, just as an interactive Algol-like language should have aids for locating matching begin-end pairs.

### Concat

Another S-LISP operation for building lists is *CONCAT*. It is a two-operand instruction; its first operand can either be an atom or a list, but its second operand must reference a list. The effect of *CONCAT* is to build a new list whose first element is the first argument of the *CONCAT* and the remainder of the new list is the second operand of *CONCAT*. For example *(CONCAT 'A '(B)) would evaluate to (A B).*

Note that *LIST* takes an arbitrary number of arguments and builds a list whose elements are those arguments. On the other hand, *CONCAT* takes only two arguments, an element and a list, and adds the element to the front of the list. For example:

*(LIST '(A) '(C))*

gives:

*((A) (C))*

while:

*(CONCAT '(A) '(C))*

gives:

*((A) C)*

# What makes the Microtek Printer so different? Nothing!

# EXCEPT....

**THE PRICE:** $750 (with parallel interface)

## THE PERFORMANCE:

- 80 or 120 columns (software selectable)
- Plain paper
- Pin Feed
- Double width printing
- 125 characters per second, 70 lines per minute nominal throughput
- 9x7 Matrix (80 columns/line), 7x7 Matrix (120 columns/line)
- Vertical Format Unit
- 96-character ASCII (upper and lower case)
- Forms width continuously adjustable between 4.5 inches and 9.5 inches (including sprocket margins)
- Parallel (Centronics type) interface standard. Serial (RS-232) and IEEE-488 interfaces available

---

To: **MICROTEK,** Inc., 7844 Convoy Court, San Diego, California 92111 (714) 278-0633

☐ **Send me more information.**

☐ **Send me a printer with:**

☐ Parallel interface @ $750. ☐ Serial interface @ $835. ☐ IEEE-488 interface @ $895

☐ Check or Money Order enclosed. ☐ Charge my VISA card. ☐ Charge my Master Charge card.

name (please print) ________ card number ________

address ________ cardholder's signature ________

city ________ state ________ zip ________ exp date ________

phone ________

Add $15.00 for packaging & shipping. Dealer inquiries invited. California residents add 6% sales tax.

---

Circle 209 on inquiry card.

These constructors can be used at anytime to compose new data objects. Now we can decompose lists and make new ones. We can perform evaluation of simple expressions, much like the facilities of a hand calculator. Soon we will show how to add new operations to the LISP calculator.

### Recognizers and Predicates

In traditional assembly language programming we find instructions which test for zero or compare two numbers. In LISP we manipulate data objects built from atoms and lists. The "zero" of lists is the empty list, *( )*; and so we include a test for *( )*. Since elements of a list can either be atomic or lists themselves we include a test for "atomness", *atom*. Finally, we must be able to distinguish between two nonidentical atoms using an equality test.

All LISP operations compute values. The values which our previous operations produced were atoms or lists; these new operations called *predicates* produce "truth values" —true or false. In M-LISP, we represent true and false as *t* and *f*; however, in S-LISP, these truth values must be represented as data items, so we pick the atoms *T* and *NIL* as their representations:

*EQ:* Compare two atoms. That is, *EQ* is a two-operand instruction which gives value *T* just in case those operands represent the same atom.

*ATOM:* This single-operand instruction gives *T* if its operand is an atom, and gives *NIL* otherwise.

*NULL:* This single-operand instruction gives *T* just in case its operand is the empty list, *( )*.

For example:

**S-LISP**
*(ATOM 'A)* gives *T*
*(ATOM '(A))* gives *NIL*
*(EQ 'A 'B)* gives *NIL*
*(NULL '(A B))* gives *NIL*

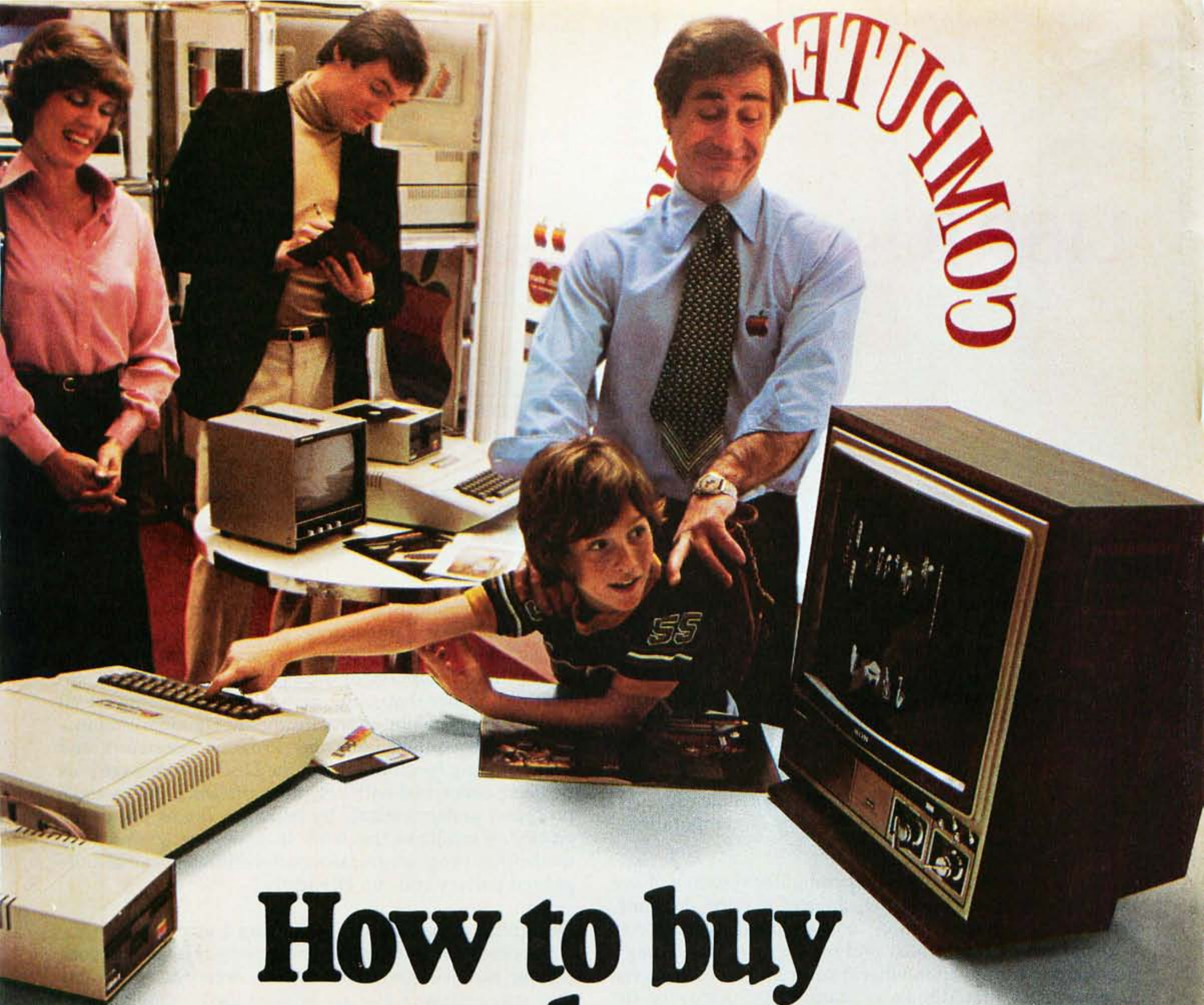
**M-LISP**
*atom[A]* gives *t*
*atom[(A)]* gives *f*
*eq[A;B]* gives *f*
*null[(A B)]* gives *f*

Since the predicates are value-producing they can be used with the other list-manipulating operations:

*(CONCAT (ATOM 'A)*
*(LIST 1 'A))* gives *(T 1 A)*

Notice that the *atom* predicate is of slightly different character than *eq* and *null*. Namely, *atom* is performing a "type test" on a data structure; such predicates are called *recognizers*.

*Text continued on page 118*



# How to buy a personal computer.

*In California, a store owner charts sales on his Apple Computer. On weekends though, he totes Apple home to help plan family finances with his wife. And for the kids to explore the new world of personal computers.*

*A hobbyist in Michigan starts a local Apple Computer Club, to challenge other members to computer games of skill and to trade programs.*

Innovative folks everywhere have discovered that the era of the personal computer has already begun—with Apple.

Educators and students use Apple in the classroom. Businessmen trust Apple with the books. Parents are making Apple the newest family pastime. And kids of all ages are learning how much fun computers can be.

Circle 9 on inquiry card.

**Visit your local computer store**

The excitement starts in your local computer store. It's a friendly place, owned by one of your neighbors. He'll show you exactly what you can use a personal computer for.

**What to look for**

Your neighborhood computer store has several different brands to show you. Chances are the salesman will recommend an Apple Computer. Apple's the one you can program yourself. So there's no limit to the things you can do. The more you use your Apple the more uses you'll discover. So it's important that Apple is the computer with more expansion capability. You can't outgrow Apple.

**It's your move**

Grab a piece of the future for yourself—we'll give you the address of the Apple dealer nearest you when you call our toll-free number. Then drop by and sink your teeth into an Apple.

**(800) 538-9696.**

In California,

**(800) 662-9238.**

apple computer

# LISP Based Systems for Education

J Laubsch, G Fischer, and H D Bocker
Institute for Information
University of Stuttgart
Stuttgart, GERMANY

## Future Computer Culture

There is sufficient evidence that personal computer systems will become as powerful as today's computer systems used in artificial intelligence research. Within the artificial intelligence community people are concerned about possible uses of computers in an evolving computer culture. The basic goals of artificial intelligence are to:

- synthesize systems that behave intelligently;
- understand intelligence in terms of computational concepts.

The human needs a personal computer system will one day help to satisfy cover the range of playing, learning, recreation, artistic creation, and personal assistance to expand one's own memory and reasoning power. Using a computer to build an intelligent tutor and an educational environment that stimulates learning by discovery (ie: through simulation, exploratory problem solving) are of central importance to artificial intelligence. Although canned software for educational applications will be widely available there remains a need for programming to tailor the system to the user's individual needs and requirements.

Our notion of what *programming* is all about will drastically change: it will cover a wide range of possible relationships between man and machine where a person creates and manipulates dynamic information structures according to personal tasks and taste. Program writing, in the historical sense of writing individual statements, is just one aspect of using a computer and will become less relevant, if not obsolete, compared to the understanding and modification of prefabricated software components.

## LISP Based Systems

Historically, LISP has been used as the *basic tool of artificial intelligence* since the computational ideas embedded in it, together with the program development system built around the language, lend themselves most naturally to the design of complex systems.

The design of LISP systems has been guided by an emphasis on supporting the user to solve complex, ill-structured, poorly understood problems at already early stages (eg: problem formulation, approximations to the final solution, support of debugging and program modification), rather than only the final step of coding a well understood problem or an already known algorithm in a given programming language. Program constructs and programming methodology in the LISP culture were particularly concerned with *cognitive efficiency* (ie: to make programs understandable by humans). It was one of the gratifying results of this work that these programs, with the help of program transformation systems, can also be proved correct and run efficiently.

## Designing a Personal Information System

Suppose you want to design a personal notebook containing people's names, addresses, interests, programs they use, messages you are sending them, appointments you make with them, etc. Such a system will consist of frequently changing information structures. As a personal information system it should model and extend that information system in our head. By using the system, we will feel the need for new features that should be incorporated (ie: an easy to learn command language or an instructional help facility to introduce a new user). A more advanced version of the system should be able to perform simple deductions. For instance, if we tell the system at some point of time, "My friend Jim has moved to San Francisco" and later ask it to, "List all friends in California," Jim should be included in the set. Eventually this system could "grow up" to become a personal assistant.

We will show that the computational ideas of LISP, as developed in the artificial intelligence community, are particularly well-suited for this kind of application.

## Basic Computational Ideas

We list those ideas which are relevant to the design of *complex* programs and transcend the capabilities of other languages and systems. In almost all interesting educational applications of computers, complex programs will be involved:

- *Incremental design.* E Sandewall feels that interactive middle-out programming (besides top-down and bottom-up approaches) is a natural way to build a complex system in a process of *structured growth*. We construct a simple version of the system, try it out, identify our misunderstandings and debug it. This knowledge, and our critique, will lead to modified specifications, and a new cycle of exploratory programming begins. Since LISP systems are incremental, old modules may be modified and new building blocks can be added with an immediate effect. The compilation of fully debugged code is available as an optional feature.

- *Complex dynamic data-structures.* Most information processing models and problems to be solved with the computer will deal with complex dynamic structures like lists, trees, nets, property lists, etc, and will not be based only on numbers and strings. In our above example, the information associated with a person could be represented in a natural way as the linked structure in figure 1. It should be easy to include new attributes or provide for a business as well as a home address.

  We define data structures abstractly through functions: constructors to build a datum; selectors to extract an attribute, and predicates to examine the type of a data structure. Including other representations, such as graphics, is easy since most LISP systems contain a higher level assembly language that gives access to the machine level.

- *Data-program equivalence.* A typical strategy to attack problems in artificial intelligence is to define *layers of languages*, each suited to a particular level of abstraction (eg: <user interface language> → <interim language 1> → . . . → LISP). The definition of LISP itself, as stated by John McCarthy, provides a good model for this approach, since most of a LISP system is itself written in LISP, except for a handful of *primitive functions*. For example, the user's command is translated into a program and then evaluated as in table 1.

  LISP facilitates this approach since the function EVAL lets the user evaluate any data as a program! The inverse is also true; it is quite easy to write programs which manipulate other programs as if they were data.

- *Pattern matching and data driven programming.* The system should respond to situations where the order in which certain actions are to be taken is not specified in advance. Furthermore, in many situations it will be impractical to specify a question literally: we might have to leave slots open which can be filled in by the system, using the knowledge contained in its data base. In our example, many other types of requests are possible. To translate them, patterns to *decompose* and *recompose* them can be defined.

| | |
|---|---|
| List All Friends in California | User Input |
| ⇓ | Translation |
| *(FOR ALL X IN (GET/FILE FRIENDS)*<br>(IF (GET/STATE X):CALIFORNIA<br>THEN (PRINTOUT (GET/NAME X)))) | Evaluation |
| ⇓ | |
| List of Addresses | System Output |

*Table 1: A typical problem approach may be to take a user command and translate it into program instructions. These program instructions are then executed by the computer. This is an example of taking a high level user language and converting it into efficient machine language.*

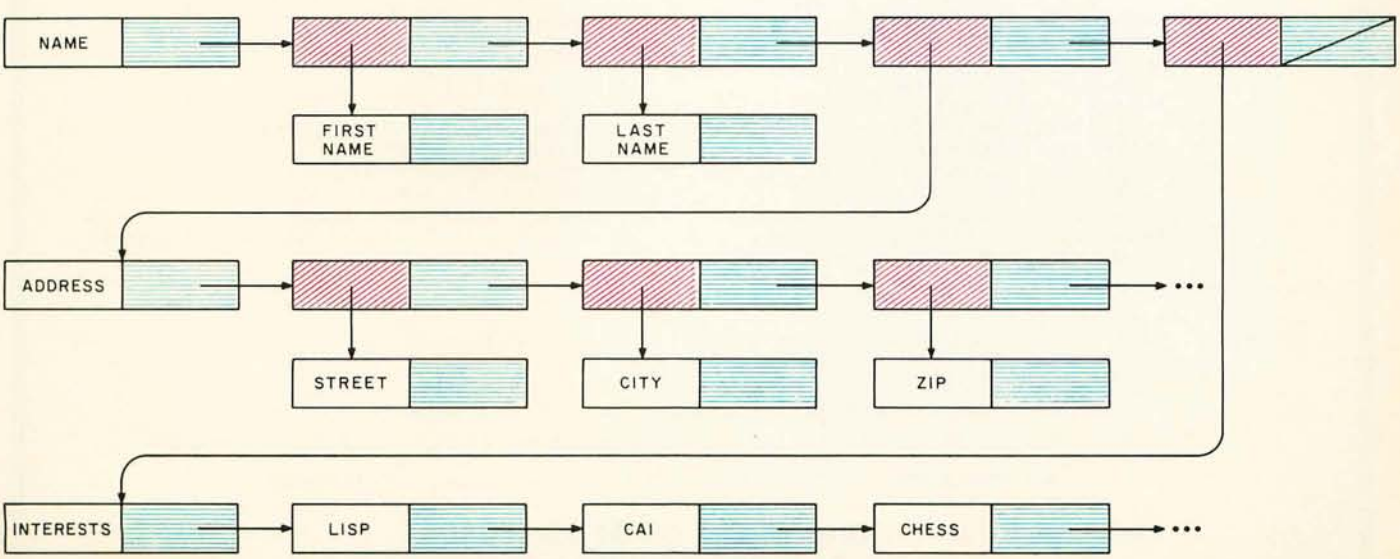



*Figure 1: An example of a linked list. This form of linked list is called a singly linked list. In a singly linked list, the user can only move in one direction, forward in the direction of the arrow. In a doubly linked list, the user can retrace the steps taken to arrive at the present location.*

Pattern languages like these are easy to implement in LISP (see Winston, Bocker and Fischer, and Kornfeld's article in this issue). Constructs consisting of condition action pairs form the basis of production systems as described by Newell and Simon. Procedure calls are triggered (and thus, data structures are manipulated) by the global state of a world (ie: the data/knowledge available) and not according to a predefined calling structure.

- *Property lists.* Property list-like structures form the basis of an associative memory. They were developed in list processing languages (eg: IPL-V and LISP) and have been generally (ie: in many programming languages) accepted as constructs which are conceptually easy to handle.

  They allow procedures to be linked to data items and evaluated depending on the current state of the system. For example, to update the address of Jim we may write:

  (APPLY (GET JIM UPDATE/ADDRESS) (READ))

  The first argument of APPLY is an address updating function, which is stored on the property list of JIM under the property UPDATE/ADDRESS. The second argument of APPLY is the argument the update function will become applied to. In our example these data will be requested from the user through the function READ.

The educational value of these ideas is that they provide *powerful ideas* for the personal computer user who wants to shape a reactive environment to his needs.

### LOGO Based Learning Environments

LOGO is, up to surface structure, more or less equivalent to LISP. LOGO as a programming language (developed by W Feurzeig and S Papert) was designed and developed to form the basis for learning environments in which the student taking an active role can learn about computers and use them to investigate issues in education and cognitive psychology. The LOGO system supports two different (by no means *disjoint*) environments: the Turtle, Graphics and Musicbox world (ie: peripheral devices which are controlled by a command language) and the LISP world. A well-engineered programming environment, based on an LSI-11, is commercially available as a stand alone, personal computing system. It integrates the language processor, editor, tracer, debugger, file management, document facilities and text processing into one system (comparable efforts to build similar systems around Pascal are still in their infancy).

LOGO projects working on computers and education can be found in many places around the world. We briefly summarize the experiences we gathered in our project in Darmstadt (see also Fischer):

- *Basic computational ideas* like recursion, the con-

# Both sides now

## North Star Announces —
**Double Density x 2 Sides = Quad Capacity!**

The North Star Horizon now delivers quad capacity by using two-sided recording on our new mini drives! That's 360,000 bytes per diskette! A four drive North Star system accesses over 1.4 megabytes of information on-line! Think of the application flexibility that so much information storage can give you!

North Star has quadrupled the disk capacity of the Horizon computer but prices have increased a modest 15 percent. On a dollar per byte basis, that's a bargain that is hard to beat!

The proven North Star disk controller was originally designed to accommodate the two-sided drives. North Star DOS and BASIC are upgraded to handle the new capacity, yet still run existing programs with little or no change. Of course, single sided diskettes are compatible with the new disk system.

Get both sides now! Quad capacity is available from your North Star dealer.

NorthStar

North Star Computers
1440 Fourth Street
Berkeley, CA 94710
(415) 527-6950 TWX/Telex 910-366-7001

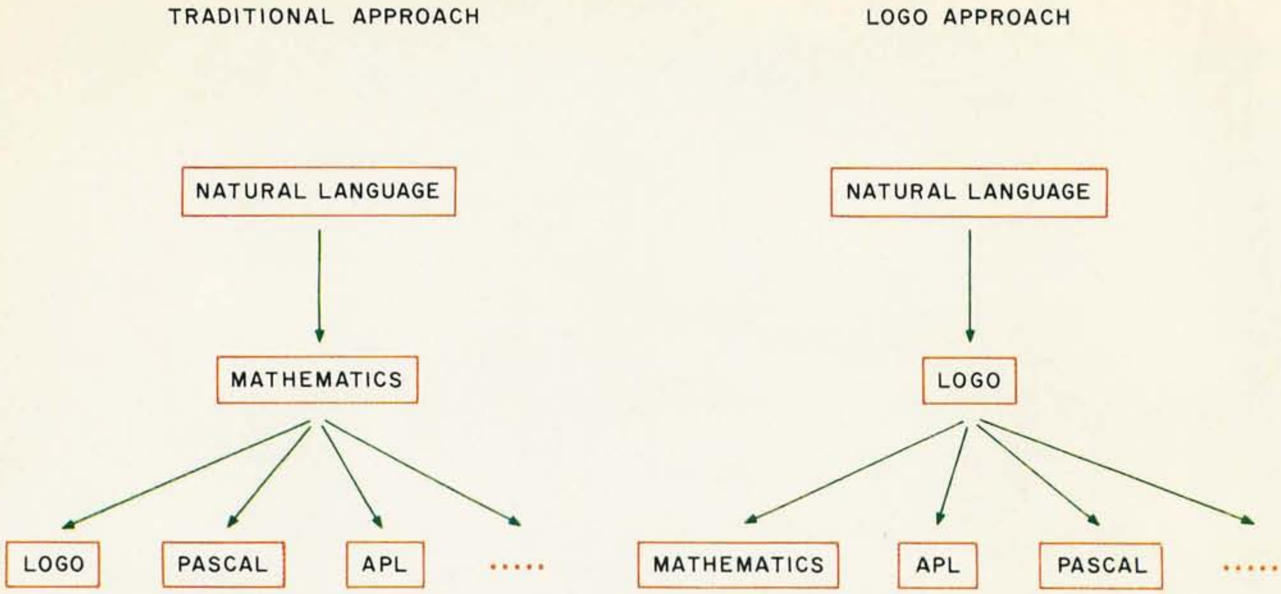



*Figure 2: Two different approaches to bridging the gap between natural language and the formal symbols of programming languages are represented. The traditional approach links everything closely with mathematics and uses mathematics as the bridge. In the authors' approach, the LOGO language is used as the bridge since it can be used to develop reasoning powers without having to become involved with the language of mathematics.*

cept of an interpreter, list processing and those mentioned above can be naturally integrated into interesting projects, caused no difficulties for students to understand, and can be considered as powerful in the sense that they are widely applicable (even in problem solving situations without the computer).

- *Graphic devices,* music box, etc, provide strong motivational support, excellent *entry points* to explore the world of computation because early success is possible and interaction with the machine is based on observable and intuitively understandable events.

- *Our experiences,* especially with young students, indicate that programming in LOGO may serve as a *bridge* between natural language communication and reasoning and the formal and abstract symbols and reasoning in mathematics and programming languages. The findings differ greatly from the traditional approaches where computer scientists try to keep things linked as closely as possible to mathematics, assuming that mathematics could serve as a bridge to programming (which we all know is questionable because most people are more alienated by mathematics than attracted). Figure 2 illustrates the two different approaches.

  Our findings can at least be partly explained by the cleanliness by which the basic computational ideas are embodied in LISP/LOGO.

- *Our programming environment* stimulates *learning by discovery*. New concepts are discovered by solving a problem through incremental writing and debugging of programs. The computer serves as a medium to test one's own understanding of concepts and of poorly understood systems (ie: if we really understand something, we can write a computer program that will do it).

- *Group projects* are easy to realize since the program development system supports the organization of modules as building blocks. In our example of a personal information system, one person could write the module to translate inputs into an internal representation, another person may write a deductive component and a third person could deal with the problem of how to answer requests or questions from the user.

- *Our programming methodology* differed in an essential way from other approaches. Procedures, including parameters and recursion as basic control structures, were introduced long before the concept of a variable was mentioned. These two aspects are not independent of each other. They basically introduce the learner to "pure LISP" (ie: a version of LISP without variables) and avoid the problems associated with side-effects and global variables.

- *Our empirical evidence* indicates that learning *other programming languages* (eg: BASIC, Pascal) after having learned LOGO was easy because constructs in these languages could be easily mapped into known concepts, whereas this statement does not hold in the other direction.

**Intelligent Computer Assisted Instruction**

Despite our belief that the most important impact of computers for educational applications will be the active independent use described in the previous section (the

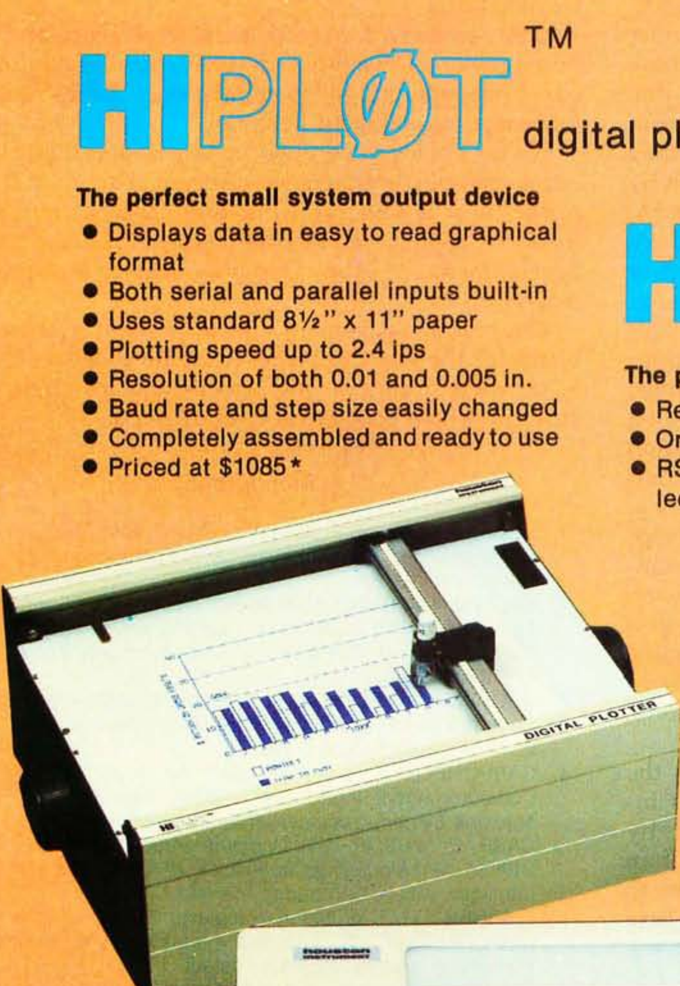
# Graphics for small systems were too expensive...

# Until Now

## HIPLØT™ digital plotters

**The perfect small system output device**

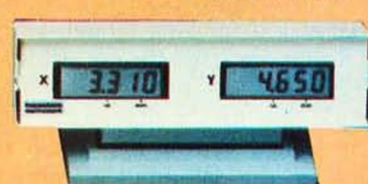
- Displays data in easy to read graphical format
- Both serial and parallel inputs built-in
- Uses standard 8½" x 11" paper
- Plotting speed up to 2.4 ips
- Resolution of both 0.01 and 0.005 in.
- Baud rate and step size easily changed
- Completely assembled and ready to use
- Priced at $1085*




## HIPAD™ digitizers

**The perfect small system input device**

- Resolution and repeatability of 0.005 in.
- Origin is completely relocatable
- RS232C and 8 bit parallel interface selectable at the connector
- Accuracies of ± 0.015 in. (0.4mm)
- Optional LC display shows actual values being inputted
- Digitizing surface 11" x 11"
- Priced at $795*




(Optional LC display)


*U.S. Domestic Price Only
TM Trademark of Houston Instrument

For rush literature requests
or local sales office information only,
persons outside Texas call toll free 1-800-531-5205.

**houston instrument** | DIVISION OF BAUSCH & LOMB

ONE HOUSTON SQUARE (512) 837-2820
AUSTIN, TEXAS 78753 TWX 910-881-2022

*"the graphics - recorder company"*

EUROPEAN OFFICE Rochesterlaan 6 8240 Gistel Belgium Phone 059/277445 Telex Bausch 81399

Circle 172 on inquiry card.

student *teaches* the computer), we do not overlook the rich potential of using intelligent programs to teach the student certain subjects, to involve and tutor him in game playing situations, and to diagnose his difficulties.

The traditional computer aided instruction was modelled on a reduced view of learning: present a stimulus item to the learner, receive a response and give a reinforcement. More advanced programs select the material to be presented according to how well the student is doing, or give him a possibility to select the particular topic he wants to study or practice. From a more comprehensive view of learning, it is essential to diagnose the learner's cognitive development and support him through a tutor who is himself an expert in the problem and can infer the conceptual difficulties this learner may encounter. A prototype is the Buggy program written by artificial intelligence researchers J Brown and R Burton, which goes far beyond traditional computer aided instruction programs by integrating artificial intelligence techniques and cognitive theories about learning, teaching and debugging.

Buggy relies on the basic pedagogical assumption, which was verified through extensive empirical findings, that students give wrong and arbitrary answers in only a few cases but tend, rather, to answer a different question or compute a result according to a different algorithm. They behave, in many cases, with *absolute consistency* with respect to their own theories. To provide real help, the teachers have to deduce the underlying misunderstanding (ie: the deep structure) from scarce observations on the surface. Buggy is a program which does this for simple arithmetic skills. The knowledge to draw an inductive inference is stored in a diagnostic model, which tries to capture possible deviations from the correct way of doing the task.

Another example that uses a diagnostic model is the Wumpus advisor (called Wusor II), which teaches inference strategies in the Wumpus game created by Gregory Yob. The program teaches the knowledge of an expert player by tailoring its advice and explanations to its current estimation of the player's knowledge. These programs may serve as prototypes of intelligent tutoring programs to teach the playing of games.

A different approach in intelligent computer aided instruction does not include an expert tutor, but is guided by the philosophy of creating a simulated environment which the user is free to explore at will. The discovery of this environment leads to the acquisition of new skills and knowledge. Prototypes of such systems are: Scholar, a question answering system to learn about geography in a mixed initiative dialogue (Carbonell); Sophie, a system to teach electronic trouble-shooting (Brown, Burton, Bell); and the Logic program developed at Stanford (Suppes). What makes these programs appear to behave intelligently is the fact that the knowledge they teach is used by these systems in many ways to carry out dialogues (for an overview see Laubsch).

A crucial component of friendly, intelligent, computer aided instruction systems is natural language (eg: the Sophie system). Rapid advances in artificial intelligence make it seem likely that natural language interfaces will be available for many applications of interest to the general public.

It is not possible to explain the details of these programs here down to an implementation level, because these systems are large and complex as compared to current standards. The historical evidence may suffice to show that all these systems have been implemented in large sophisticated LISP systems (eg: InterLISP) which have matured over more than a decade to support the development of systems of this size.

## Conclusions

LISP remains a *tool* in artificial intelligence and educational research, even though it has contributed greatly to our understanding of computational issues and their relevance to intelligent behavior.

We do not want to give the impression that all interesting uses of computers are centered around LISP. Some of the most innovative work was done by the Learning Research Group at Xerox Research Center in their development of the Dynabook and the Smalltalk language.

The real issues remain and pose many research problems for the years to come: to create cognitive theories; to create a science of intelligence, and to apply it successfully to the problems of education. ■

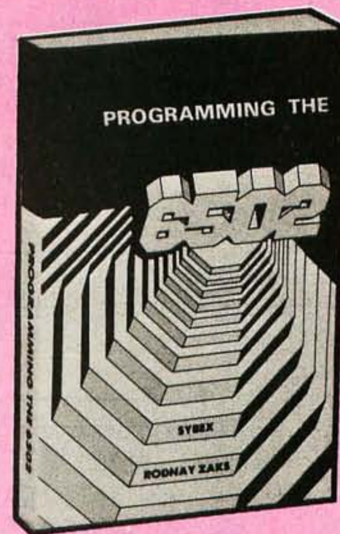
### Bibliography

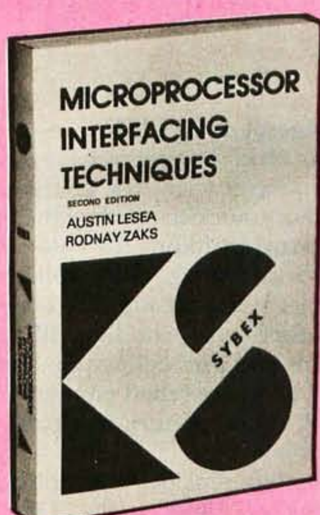

1. BOCKER, H D, and FISCHER, G, *Interaktives Problemlosen mit Computerhilfe: Problemaufgaben*, Forschungsgruppe CUU, Darmstadt, 1978.
2. BROWN, J S, BURTON, R R, and BELL, A G, "Sophie: A Step Toward Creating a Reactive Learning Environment," *International Journal of Man-Machine Studies*, volume 7, 1975, pages 675 thru 696.
3. BROWN, J, and BURTON, R, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," *Cognitive Science*, volume 2, 1978, pages 155 thru 191.
4. CARBONELL, J R, "AI in CAI: An Artificial Approach to Computer-aided Instruction," *IEEE Transactions on Man-Machine Systems*, volume MMS-II, number 4, 1970.
5. CARR, B, *WUSOR II: A Computer Aided Instruction Program with Student Modelling Capabilities, AI-Memo 417*, MIT Artificial Intelligence Lab, Cambridge MA, 1977.
6. FISCHER, G, "Das Losen komplexer Problemaufgaben durch naive Benutzer mit Hilfe des interaktiven Programmierens," Forschungsgruppe CUU, Darmstadt, 1977.
7. FEUERZEIG, W (editor), *Programming Languages as a Conceptual Framework for Teaching Mathematics, BBN Report Number 2165*, Cambridge MA, 1971.
8. General Turtle Corporation, 120 Boulevard Industriel, Boucherville Quebec, 34B 2X2 CANADA.
9. KAY, A, "Microelectronics and the Personal Computer," *Scientific American*, September 1977, pages 231 thru 244.
10. LAUBSCH, J, "Artificial Intelligence Methoden im CUU," K Brunnstein et al (editors), *Lecture Notes in Computer Science*, volume 17, Springer Verlag, Berlin, 1974, pages 385 thru 393.
11. McCARTHY, J, "A Micromanual for LISP—not the Whole Truth," *SIGPLAN Notices*, volume 13, number 8, August 1978, pages 215 thru 216.
12. PAPERT, S, "Uses of Technology to Enhance Education," *Logo Memo 8*, MIT Artificial Intelligence Lab, Cambridge MA, 1973.
13. SANDEWALL, E, "Programming in the Interactive Environment: The LISP Experience," *ACM Computing Survey*, volume 10, number 1, 1978, pages 35 thru 72.
14. SUPPES, P, SMITH, R, and BEARD, M, *University Level Computer assisted Instruction at Stanford, TR number 265*, IMSS, Stanford University, Stanford CA, 1975.
15. WINSTON, P, *Artificial Intelligence*, Addison Wesley, Reading MA, 1977.
16. YOB, G, "Hunt the Wumpus," *Creative Computing*, September and October, 1975, pages 51 thru 54.

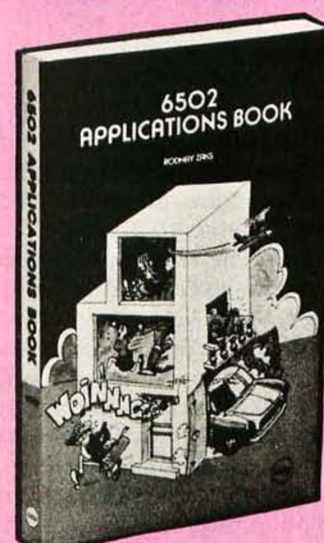
SYBEX

## LEADER IN COMPUTER EDUCATION

SYBEX, INC.
2020 Milvia Street
Berkeley, California
94704


- ☐ **PROGRAMMING THE Z80, ref C280,** by Rodnay Zaks, 330 pp. .......... $11.95
- ☐ **PROGRAMMING THE 6502, ref C202,** by Rodnay Zaks, 350 pp. (2nd Edition). .......... $11.95
- ☐ **6502 APPLICATIONS BOOK, ref D302,** by Rodnay Zaks, 288 pp. .......... $12.95
- ☐ **MICROPROCESSOR INTERFACING TECHNIQUES, ref C207,** by Rodnay Zaks, 420 pp. .......... $14.95
- ☐ **AN INTRODUCTION TO PERSONAL AND BUSINESS COMPUTING, ref C200,** by Rodnay Zaks, 250 pp. . $6.95
- ☐ **MICROPROCESSORS-FROM CHIPS TO SYSTEMS, ref C201,** by Rodnay Zaks, 420 pp. .......... $9.95
- ☐ FREE DETAILED CATALOGUE

**TO ORDER:**

**By Phone:** (415) 848-8233. Visa, Mastercharge, American Express.

**By Mail:** Check books above. Include payment.

**Shipping:** Add $1.50 per book (UPS), or 75¢ (4th class- allow 4 weeks).

**Tax:** In California, add tax.

NAME ______________________
COMPANY ______________________
ADDRESS ______________________
CITY ____________ STATE ________ ZIP ________
☐ Charge my: ☐ Visa ☐ Mastercharge ☐ American Express
Number ______________ Exp. Date ________
Signature ______________________

Circle 358 on inquiry card.

# The Lambdino Storage Management System

Gianfranco Prini
Instituto di Scienze dell'Informazione
Universita' di Pisa
Corsa Italia 40
I-56100 Pisa ITALY

Martin Rudalics
Institut fuer Mathematik
Johannes Kepler Universitaet Linz
A-4045 Linz/Auhof AUSTRIA

Lambdino is a statically scoped dialect of LISP (see glossary for definitions). The name Lambdino is a combination of lambda, Landin, and ino, where lambda stands for itself, Landin refers to a person. and ino is an Italian suffix for small. The reference to Peter Landin is due to the fact that he designed the first statically scoped applicative language based on the interpretive philosophy of LISP (as described in his paper entitled "The Mechanical Evaluation of Expressions"). Other predecessors of Lambdino include the anonymous language used by Reynolds in his work *Definitional Interpreters for Higher-Order Programming Languages* and in *Scheme* as described by G Sussman and G Steele.

A detailed description of Lambdino and the problems posed by its implementation are beyond the scope of this paper. Here we only want to sketch some ideas on which we have based its storage management system. Thus LISP or Scheme may be substituted for Lambdino throughout this paper.

An explicit design goal of Lambdino is its transportability onto a wide class of computers, including microcomputers. Particular care has been put into the development of the Lambdino storage management system in order to fit the space and time constraints of microcomputers. A machine independent version of Lambdino, implemented in MagmaLISP, has been realized and will be bootstrapped in the near future on several machines, including an IBM System/370 Model 168 (IBM 74) and a Zilog Z-80 Development System. The only assumption made in this implementation is that the memory of the host machine is structured into directly addressable bytes.

## Storage Management in LISP

Implementors of LISP systems have developed various techniques to make efficient use of free storage (ie: that part of the memory not occupied by the operating system and the LISP kernel including the data structure manipulating primitives and the garbage collector). In all these techniques, objects are manipulated via pointers, and arbitrary run time type checking is possible in both system programs and user defined functions. This is normally achieved by using typed pointers in a more or less explicit way. A typed pointer is a pair <T,A> which identifies an object type T located at address A. The length of A usually coincides with the address length of the host machine (eg: 18 bits in the PDP-10, 24 bits in the IBM System/370). In this way, the hardware addressing mechanism may be efficiently used for the implementation of most data structure manipulating primitives. The representation of T usually requires only a few bits (typically 2 or 3 in *small* systems with a limited number of data types, 7 or 8 in *large* ones).

Although it is possible to implement a typed pointer <T,A> as the concatenation of the bit strings representing T and A, in some systems only A is represented explicitly, while T is implied by (ie: is a function of) A.

Acknowledgements

*The work reported in this article has been partly supported by CNR during a period of four months spent by Martin Rudalics at the Instituto di Elaborazione dell' Informazione with a scholarship of the Italian Foreign Office.*

**A one-to-one correspondence between partitions and data types is implicitly established.**

There are basically three ways of implementing typed pointers.

### Contiguous Partitions

Free storage is divided into a number of areas called partitions which consist of contiguous memory cells (ie: bytes or words). Each partition is allowed to contain only data belonging to the same type (also referred to as the type of the partition, see figure 1). A one-to-one correspondence between partitions and data types is implicitly established by the implementation of the data structure manipulating primitives. The type T of an object is obtained by comparing its address A with the boundaries of the partitions.

This technique has been adopted by the PDP-10 implementation of LISP 1.6 and some early versions of MacLISP. In fact it is particularly suited to those computers in which typed pointers are not allowed to contain an explicit representation of T without a considerable waste of space. As an example, one word in the PDP-10 is 36 bits long and may contain exactly two addresses. If one half word were reserved for representing T, several bits would remain unused.

Contiguous partitions may be disadvantageous when the partition associated with a type T becomes full and the allocation of a new object of type T is requested. The garbage collector may then fail to recover sufficient space for allocating the new object, even though other partitions are nearly empty. This drawback may be eliminated by enlarging the overpopulated partition and contracting the underpopulated ones. A compacting garbage collector with additional phases is required for this purpose. After the compaction phase, the boundaries of the partition are redefined, data is moved to fit the new boundaries and all pointers to moved data are updated accordingly.

### Paged Partitions

Free storage is divided into pages of equal length (usually a power of 2, eg: 1 or 2 K bytes or 256 or 512 words). A page is referred to as busy or free, according to whether or not it currently contains data. Like contiguous partitions, each busy page may contain only data belonging to the same type, further referred to as the type of the page. The correspondence (usually many-to-one) between busy pages and their respective types is dynamically realized by a type table, which also keeps track of the free pages (see figure 2).

The type T of an object located at address A may be retrieved by accessing the type table using the most significant bits of A as an index (this is possible if the page length is a power of 2). When a object of type T is to be allocated and no more space is available in pages of type T, a new free page is used and its type is set to T. Thus, the partition associated with a given type is distributed over several pages. The garbage collector compacts all data of a given type into as few pages as possible.

This technique, which has been developed as an alternative to contiguous partition for the same class of computer architectures, has been empleyed in the PDP-10 implementation of INTERLISP and recent versions of MacLISP (as described by G Steele in *Data Representation in MacLISP*).

As for the efficiency, paged partitions and contiguous partitions with variable boundaries are comparable: the necessity of accessing the type table may lead to a slower type checking, but the garbage collector need not recompute boundaries and move data accordingly. A nice property of this technique is its compatibility and smooth interaction with timesharing operating systems that have paged virtual memories. In fact, the page table used by the operating system and the type table may be easily combined.

### Paged Partitions with Tagged Pointers

This technique is identical to the preceding one, except

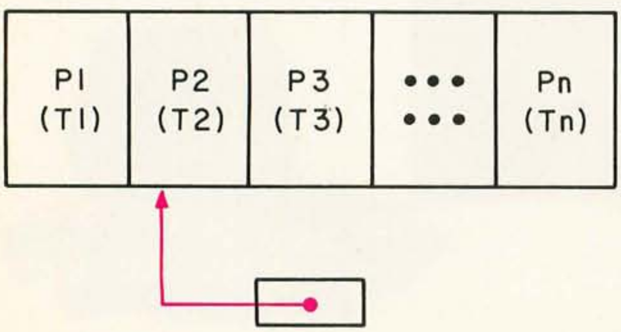



*Figure 1: Contiguous partitions: a pointer to an object of type T2.*

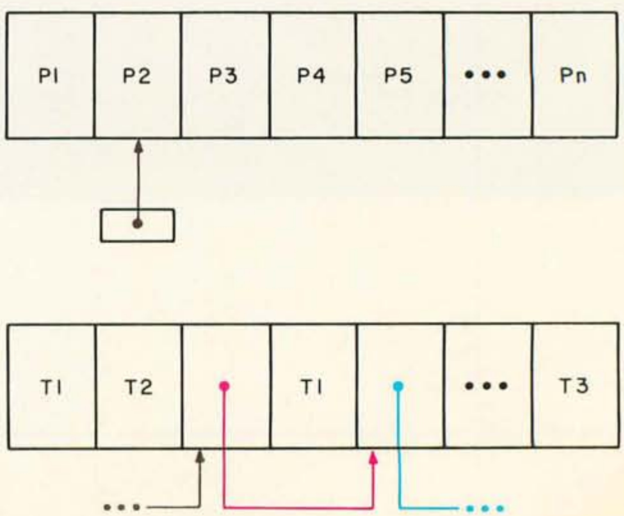



*Figure 2: Paged partitions: a pointer to an object of type T2.*

**Tinker, Tailor, Soldier, Sailor . . .**
**Doctor, Lawyer . . . the Chieftain's here.**

No matter whether you're a serious hobbyist or a serious businessman, the Chieftain 6800 microcomputer with capabilities that surpass the Z-80 is made for you.

Smoke Signal's quality-packed Chieftain I features two 5.25-inch mini-floppy drives and Chieftain II features two 8-inch floppy drives.

Both microcomputers provide 32K static memory, two serial I/O ports, a 2 MHz processor board, a 2K RAM monitor, a nine-slot motherboard with built-in baud rate generator and gold connectors for high reliability. The Chieftain's stylish leather-grained cabinet houses the above with its own cooling fan and regulated power supply.

Every Chieftain is complete with system software and is totally burned-in as well as tested to further insure high reliability.

And it's expandable to 64K memory with up to 2 megabytes floppy disk storage.

So see your nearest Smoke Signal dealer, he'll be glad to show you how to get your wampum's worth. Systems start at $2,595.

☐ Send information on your Chieftain microcomputer
☐ Send name of nearest dealer

Name_______________
Address_______________
Company _______________
City _______________
State/Zip _______________

SMOKE SIGNAL BROADCASTING

31336 Via Colinas, Westlake Village, California 91361, (213) 889-9340

Dealer inquiries invited.


# Hail to the Chieftain

Smoke Signal Broadcasting, 31336 Via Colinas, Westlake Village, CA 91361, (213) 889-9340

Circle 328 on inquiry card.

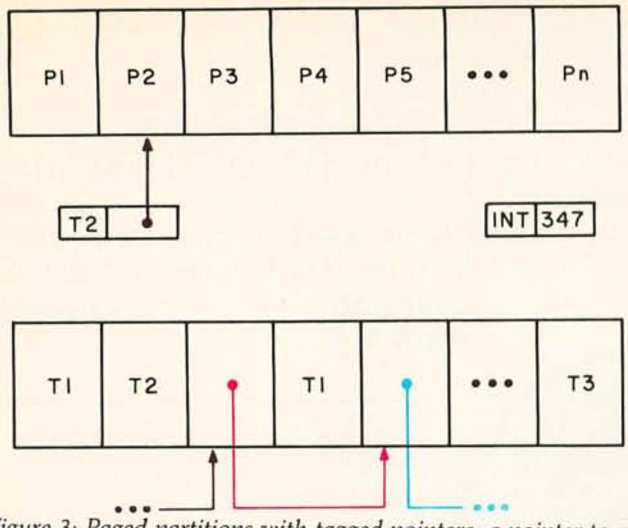


*Figure 3: Paged partitions with tagged pointers: a pointer to an object of the type T2 and the representation of the integer 347.*

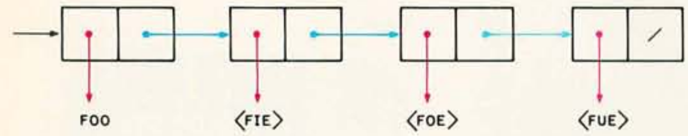


*Figure 4: Storage representation of an object created by MK-FOO.*

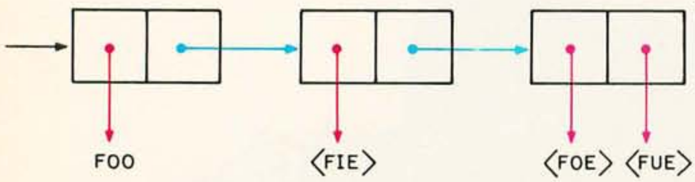


*Figure 5: Storage representation of an object created by MK=FOO.*

**LISP does not contain primitives for declaring new data types.**

for the fact that all pointers to an object also contain an explicit representation of its type T (see figure 3).

Tagged pointers have been adopted in MagmaLISP and the IBM version of InterLISP. They are convenient in computers whose word size exceeds the address length by a few bits, which may comfortably contain the representation of T type. As an example, a typed pointer <T,A> may be represented with a full word in the IBM System/370 by reserving 24 bits for A and the remaining 8 bits for T. It is interesting to note that the LISP machine (described by A Bawden, et al, in the *LISP Machine Progress Report*) implements typed pointers in this way.

Tagged pointers allow for a quick retrieval of the type of an object. Moreover, *short* constants such as characters, *small* integers, etc, may be directly represented in the address part of a typed pointer (see figure 3). The type T identifies them as *immediate* data not to be manipulated as pointers (note that no *private* pages are needed to store immediate data). The main drawback of this technique is that information is somehow duplicated: in fact, a type table is still needed by the garbage collector during the compaction phase.

## How To Get Rid of Most Terminating NILs

LISP (unlike ALGOL 68 and Pascal) does not contain primitives for declaring new data types. However, S expressions are an effective tool allowing the user to program new data types explicitly.

As an example, consider a record class named FOO whose instances contain the fields FIE, FOE, and FUE. The data type FOO may be programmed in LISP using proper lists (ie: lists ending with NIL) as follows:

```
(DEFINE MK-FOO  (FIE FOE FUE)
   (LIST ' FOO FIE FOE FUE))
(DEFINE IS-FOO    (X) (EQ (CAR X) ' FOO))
(DEFINE FIE-OF    (X) (CADR X))
(DEFINE FOE-OF    (X) (CADDR X))
(DEFINE FUE-OF    (X) (CADDDR X))
```

The storage representation of an object of type FOO is shown in figure 4. It is immediately evident that this representation is space consuming: in fact, the last cell may be eliminated, and the pointer turned into a pointer to <FUE> (see figure 5). To this purpose, MK-FOO and the other functions may be redefined as follows:

```
(DEFINE MK=FOO  (FIE FOE FUE)
   (CONS ' FOO (CONS FIE (CONS FOE FUE))))
(DEFINE  IS=FOO  (X)  (EQ (CAR X) ' FOO))
(DEFINE  FIE =OF  (X)  (CADR X))
(DEFINE  FOE=OF  (X)  (CADDR X))
(DEFINE  FUE=OF  (X)  (CDDDR X))
```

Unfortunately, when the structures created by MK=FOO are printed by the standard output routines of LISP (eg: for debugging purposes), their readability decreases considerably. For instance, (MK-FOO 1 2 (MK-FOO 3 4 5)) is printed as (FOO 1 2 (FOO 3 4 5)), whereas (MK=FOO 1 2 (MK=FOO 3 4 5)) yields (FOO 1 2 FOO 3 4 . 5), thus introducing an irritating extra dot while omitting one pair of significant parentheses.

It is possible to both maintain the clean formalism of proper lists, and represent them efficiently (as indicated in figure 5) by introducing the concept of NULLCDR cells. To this purpose an additional bit, B, is associated with each typed pointer, thus yielding a triple <T,B,A>. When B is clear, <T,B A> represents a typed pointer as usual. When B is set, <T,B,A> represents a LISP cell whose CDR is NIL (ie: a NULLCDR cell) and whose CAR has type T and is located at address A. NIL must be used explicitly in only a very few cases (see figure 6).

With the introduction of NULLCDR cells, only proper lists are allowed in Lambdino. This fact has several consequences:

- Space is not only saved in the implementation of user defined data structures, but also in the list representation of interpreted functions. Most lists in purely applicative programs contain less than 3 or 4 elements, hence the introduction of NULLCDR cells allows a save of 25 to 33% in space.
- The absence of the LISP *dot notation* slightly simplifies the I/O (input/output) routines.
- The time required by CONS for checking the type of its second argument is largely compensated by the time saved using NULL (or, better, NULLCDR) instead of NLISTP as a predicate for terminating recursions. Also, the functions CAR, CDR and NULLCDR need not make a storage access when their argument is a NULLCDR cell. This may lead to a significant save of time. As an example, the function:

```
(DEFINE EVLIS (X A)
  (COND ((NULL X) NIL)
        (T (CONS (EVAL (CAR X) A)
                 (EVLIS (CDR X) A)))))
```

may be written more efficiently as:

```
(DEFINE EVLIS (X A)
  (COND ((NULL X) NIL)
        (T (EVLIS1 X A))))

(DEFINE EVLIS1 (X A)
  (CONS (EVAL (CAR X) A)
        (COND ((NULLCDR X) NIL)
              (T (EVLIS1 (CDR X) A)))))
```

This improved version saves some storage accesses and one recursive call to (and return from) EVLIS.

RPLACA and RPLACD (if they are implemented at all!) generate an error when applied to NULLCDR cells.

Standard garbage collectors (including the Schorr-Waite algorithm) are unaffected by the presence of NULLCDR cells (pointers having the NULLCDR bit set are treated exactly as usual pointers).

### Lambdino Design Issues

The Lambdino storage management system is a mixture of contiguous partitions and tagged pointers with NULLCDR bits. More precisely, the free storage is divided into two variable partitions FIXLEN and VARLEN (see figure 7).

FIXLEN may contain only fixed length data (ie: data whose memory occupation depends only on their type). There are three FIXLEN data types in Lambdino, namely atoms, cells and interpreted closures. They are records with two fields with the following characteristics:

- Atoms have a TOPVAL field which may be any datum (eg: a function definition) and a PNAME field, which must be a string (property list lovers will be allowed to use this field for holding property lists in special versions of Lambdino).
- Cells have a CAR field which may be any datum and a CDR field which must be a list, though possibly empty.
- Interpreted closures have a FUN field which must contain a LAMBDA and an ENV field which contains an ALIST (they are similar to FUNARG objects in LISP).

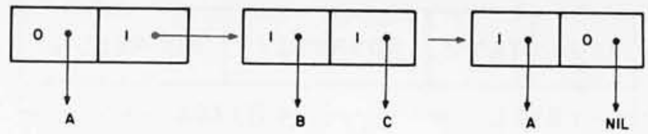



*Figure 6: Tagged pointers with NULLCDR bit: the example represents (A ((B) C)) and ((A)).*

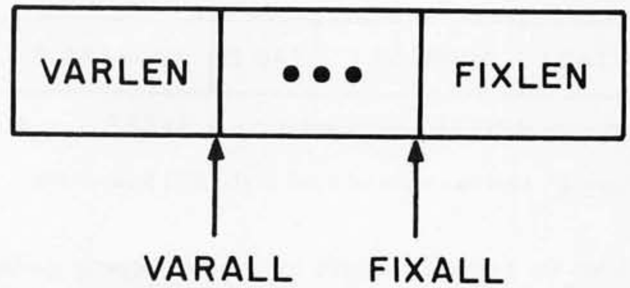



*Figure 7: Overall organization of the free storage in the Lambdino storage management system.*

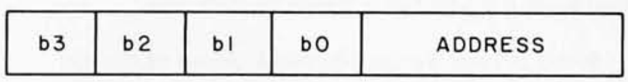



*Figure 8: Tagged pointers in the Lambdino storage management system.*

VARLEN is reserved for variable length data, ie: data which must contain explicit information on their memory occupation. There are three variable length data types in Lambdino, namely strings, compiled functions and compiled closures:

- Strings are mainly used for representing atom print-names.
- Compiled functions are binary code produced by the Lambdino compiler.
- Compiled closures contain a pointer to a compiled function (which corresponds to the FUN field of interpreted closures) and pointers to the values of its free variables (they correspond to the ENV field of interpreted closures).

A new datum is allocated by moving FIXALL to the left or VARALL to the right according to whether it is a FIXLEN or a VARLEN datum. When FIXALL and VARALL collide, a standard compacting garbage collector is invoked to contract VARLEN to the left and FIXLEN to the right. The common length of FIXLEN data

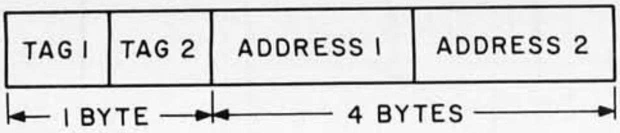



*Figure 9: Representation of a cell in the Zilog Z-80 Development System.*

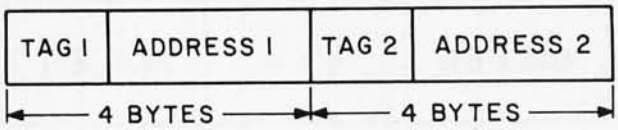



*Figure 10: Representation of a cell in the IBM System 370.*

allows the garbage collector to operate properly during the compaction phase without knowing the type of the objects. This guarantees an optimal use of the limited memory of the host microcomputer.

Data are referenced by a special kind of tagged pointers (see figure 8). The tag consists of four bits:

- b3 is used during the mark phase of the garbage collector.
- b2 is the NULLCDR bit: when it is set, the tagged pointer represents a NULLCDR cell.
- b1 and b0 are used together with A to determine the type of a datum.

The datum type is determined by bits b1 and b0 as follows:

- When either b1 or b0 is set, A is interpreted as the address of a fixed or variable length datum, according to whether A points into FIXLEN or VARLEN. In this case the three possible configurations of b1 and b0 are sufficient to cover the three types of FIXLEN and VARLEN data, respectively.
- When b1 and b0 are both clear, A is to be interpreted as an integer number. Integers constitute the seventh data type of Lambdino and are always represented as immediate data.

## Implementation Details

Our inplementation of Lambdino is supported by an abstract stack machine SM which contains the following primitives, in addition to standard arithmetic and control routines (we assume that A is a nonnegative Lambdino integer, V a nonnegative Lambdino integer less than 256, P an arbitrary Lambdino tagged pointer).

*(GETBYTE A)* returns an integer representing the contents of the byte located at address A.

*(PUTBYTE A V)* stores V into the byte located at address A.

*(GETCHAR)* reads the next character from the input stream and returns its integer representation.

*(PUTCHAR V)* writes the character represented by V into the output stream.

*(GETTYPE P)* returns the integer representation of the tag of P.

*(PUTTYPE P V)* returns a new pointer having tag V and the same address part as P.

The Lambdino storage management system, which is entirely written in terms of these primitives, contains parameters to define the size of addresses and to specify whether or not two tags have to be packed into one byte. When bootstrapping the system on a Zilog Z-80 Development System, 16 bits for the representation of addresses and the packed version of tags are recommended (see figure 9), while 24 bit addresses and unpacked tags should be used on an IBM System/370 (see figure 10).

## Concluding Remarks

We have developed an experimental implementation of Lambdino written in Lambdino itself. It includes a Lambdino interpreter, an interpreter for the stack machine SM and a compiler which translates Lambdino functions into SM programs. All these Lambdino functions have been debugged using a simple Lambdino interpreter written in MagmaLISP. As all functions of the system eventually call the previously defined primitives, the system can be (and will be soon) bootstrapped by compiling it to the machine code of SM using it own compiler, and by macroexpanding the resulting code to the machine language of the host computer. ■

### BIBLIOGRAPHY

1. Allen, J, *Anatomy of LISP*, McGraw-Hill, 1978.
2. Bawden, A, Greenblatt, R, Holloway, J, Knight, T, Moon, D, Weinreb, D, *LISP Machine Progress Report*, Memo Number 444, Laboratory for Artificial Intelligence, Massachusetts Institute of Technology, 1977.
3. *IBM System/370 Model 168 Theory of Operation*, Form Numbers SY22-6931/2/3/4/5/6, IBM Corporation, Poughkeepsie NY, 1974.
4. Landin, P, "The Mechanical Evaluation of Expressions," *Computer Journal*, volume 6, number 4, 1964, pages 308 thru 320.
5. Montangero, C, Pacini, G, Turini, F, "MAGMA-LISP: a Machine Language, for Artificial Intelligence," *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, 1975, pages 556 thru 561.
6. Moon, D, *MacLISP Reference Manual*, Laboratory for Computer Science, Massachusetts Institute of Technology, 1974.
7. Quam, L, Diffie, W, *Stanford LISP 1.6 Manual*, Artificial Intelligence Laboratory, Stanford University, 1972.
8. Reynolds, J, "Definitional Interpreters for Higher-Order Programming Languages," *Proceedings of the ACM National Convention*, 1972, pages 717 thru 740.
9. Steele, G, "Data Representation in MacLISP," Memo Number 420, Laboratory for Artificial Intelligence, Massachusetts Institute of Technology, 1977.
10. Sussman, G, Steele, G, "SCHEME: an Interpreter for Extended LAMBDA Calculus," Memo Number 349, Laboratory for Artificial Intelligence, Massachusetts Institute of Technology, 1975.
11. Teitelman, W, *INTERLISP Reference Manual*, Xerox Palo Alto Research Center, 1975.
12. Urmi, J, *INTERLISP/370 Reference Manual*, Department of Mathematics, Linkoeping University, 1976.
13. *Z-80 Development System Hardware User's Manual*, Zilog Incorporated, Cupertino CA, 1977.

# no loose ends

## All-In-One: computer, floppy, I/O, 16K RAM. $1595*


### New Heathkit® H89 All-In-One Computer

Heath takes the risk out of selecting a balanced computer system. Now, video terminal, floppy, keyboard and 8-bit computer are brought together in one self-contained, compact unit. Nothing hangs out.

### Two Z80's

The personal computer has never been simpler. Or smarter. Two Z80 microprocessors mean terminal never shares power with computer, as do most desk-top units. So this terminal is capable of a multitude of high-speed functions, all controllable by keyboard *or* software.

### 102K bytes storage

Built-in floppy disk system gives you fast access to programs and data. Each 5¼-inch diskette has more than 102K bytes of storage area, enough to hold entire files. The All-In-One comes with 16K RAM, expandable to 48K.

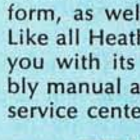
### Hundreds of uses at home or work

The All-In-One Computer runs programs written in MICROSOFT™ BASIC and ASSEMBLER Languages. And it accepts all current software written for the popular Heathkit H8 computer. You can choose from scores of practical programs for home and business.

### Learn by building

What better way to learn about computers than to build one yourself? The All-In-One is available in easy-to-build kit form, as well as completely assembled. Like all Heath electronic kits, it comes to you with its own easy-to-follow assembly manual and a nationwide network of service centers to assure smooth sailing.

### FREE CATALOG




**For complete details on the Heathkit H89 All-In-One Computer and nearly 400 other electronic kits for your home, work or pleasure, send today for the latest Heathkit Catalog of values.**

*$1195 without floppy. Mail order kit price, F.O.B. Benton Harbor, MI. Also available at Heathkit Electronic Centers at slightly higher prices. Prices subject to change without notice.

**Heathkit®**

CP-165

**HEATH COMPANY, DEPT. 334-560, BENTON HARBOR, MI 49022**

Circle 162 on inquiry card.

# Pattern-Directed Invocation Languages

William A Kornfeld
MIT Artificial Intelligence Laboratory
545 Technology Sq
Cambridge MA 02139

LISP was first developed for use in artificial intelligence research, the branch of computer science concerned with understanding the nature of intelligent activity by simulating it on a computer. LISP has proved so successful that it is the only high level language currently supported at the MIT Artificial Intelligence Laboratory. Much of its success is due to its syntax and data structures which make it a convenient base upon which to implement very high level special purpose languages.

One very important class of these high level languages is the so-called *pattern-directed invocation* languages. They made their first appearance in about 1970 with the *Planner* system at MIT. Since then, dozens of these languages have been built at sites around the world with different sets of features. The basic concepts involved can be traced back to the work of such logicians and philosophers as Frege, Russell, and Carnap in the earlier part of this century. They were concerned with representing and manipulating *facts* about the world. They began with *atomic facts* and described methods that could be used to deduce new facts from old. Pattern-directed invocation languages treat facts, represented as LISP lists, as elementary data types and usually collect them together into one or more *data bases*. Procedures can be written to derive new facts (or to decide if it is possible to derive a given fact) from those already in the data base.

In this article we will be mostly concerned with the basic concepts involved in pattern-directed invocation languages. Toward the end, a brief summary is given of some of the more advanced ideas that have found their way into these languages. Special attention is given to the problem of implementing these languages in a LISP system. Much of this implementation is surprisingly straightforward, once the basic concepts of LISP are understood. In fact, the task of implementing a system almost identical to the one described here was given to students in a beginning programming course at MIT. The students had had only a few weeks experience with LISP, and a total programming experience of a couple of months, but they had little problem with the assignment.

**About the Author:**
*William Kornfeld is a graduate student at the MIT Artificial Intelligence Laboratory. He is currently doing research in the semantics of pattern-directed invocation and extensions of these ideas to parallel processing.*

### Retrieval of Information by Pattern

Suppose we wanted to represent the knowledge, inside of our computer, that Lena is the mother of Paul. This sentence contains three important items; the two people, Lena and Paul, and the relationship — one being the mother of the other. This fact can be represented using the data structures of LISP as a list with three elements. We are free to choose any arrangement of the items in the list; placing the relation (mother-of) in the first, second, or third position of the list. I prefer to keep to the LISP (and mathematical) conventions of putting the relationship first, and having the arguments follow. This fact will be represented as:

```
(MOTHER-OF LENA PAUL)
```

We could have many such facts similarly represented by list structure inside of our machine. Some examples are:

```
(MOTHER-OF LENA FAY)
(WIFE-OF LENA SAM)
(MOTHER-OF FAY ROBERT)
(MOTHER-OF FAY ARLENE)
(FEMALE LENA)
(FEMALE FAY)
(MALE ROBERT)
(MALE SAM)
```

We call each of these facts an *assertion*. Assertions are pieces of arbitrary list structure (as far as the LISP interpreter is concerned). So that they may be used in our pro-

# THE INTELLIGENT COLOR GRAPHICS BOARD CGS-808


**Screen images are as photographed, unretouched.**

### Main Features

- Motorola MC6847 video display generator.
- Eight colors — green, yellow, blue, red, buff, cyan, magenta, orange.
- 11 programmable modes
  - 1 alphanumeric mode with 32 x 16 character and inverse video.
  - 2 semigraphic modes with 8 colors in 64 x 32 and 64 x 48.
  - 8 full graphic modes with 2 sets of 4 colors ranging from 64 x 64 to 128 x 192, and 2 sets of 1 color in 256 x 192.
- On-board 8085 microprocessor.
- 7½K bytes of on board static RAM.
- Up to 4K bytes of EPROM.
- True S-100 compatibility.

### CGS-808

Biotech Electronics proudly announces the CGS-808, a highly sophisticated color graphics board for the S-100 bus.

The CGS-808 is a single board graphics processor which is incredibly simple to use. It is I/O mapped and requires no memory space. It will operate in any S-100 system or as a stand-alone graphics processor. By changing the firmware pack, the CGS-808 can be upgraded to perform additional complex or custom graphic functions. It also provides a parallel I/O port to interface with digitizers, joysticks, light pens, or directly into a keyboard.

The CGS-808 is designed for a low cost, high quality, professional display for applications in medicine, business, education, science, industry, and video games.

### Immediate Availability

The CGS-808 is available assembled and tested for $385.00, or the bare "kit" with the MOS chip set for $99.00 (Chip set includes the MC6847, MC1372, 8085 and 2708 EPROM with graphics driver subroutines). Color monitors are available for $495.00. Delivery is from stock.

Dealer inquiries are invited.

For more information, call or write

**Biotech Electronics**
**P. O. Box 485**
**Ben Lomond, CA 95005**
**(408) 338-2686**

## Simple Pattern Matcher

A simple pattern matcher can be implemented as a LISP function of two arguments, an assertion and a pattern. Here are some examples of assertions and patterns that match:

(a b c d) *matches* (a ? ? d)
(a (b c) (d e)) *matches* (a ? (d ?))
(a ((b c) d) (e f g)) *matches* (? ((b c) ?) ?)

Examples of assertions and patterns that don't match are:

(a b c d) *doesn't match* (e ? ? d)
(a b c d) *doesn't match* (a ? d)
(a b (c d) e) *doesn't match* (a b (c d) ? ?)

Recursive procedures, such as this pattern matcher, are often thought of as procedures that take complex problems and convert them into simpler problems. Eventually this will reduce the calls to procedures that are sufficiently simple that they can be solved using already existing LISP functions.

The simple cases for this pattern matcher occur when either the pattern or the assertion is an atom. If the pattern is the atom ?, then the match should succeed because ?, by definition, matches anything. If the pattern is some other atom then the match should only succeed if the assertion is an atom, and the *same* atom. If the pattern is not an atom but the assertion is, the match should fail. These rules cover all cases where either the pattern or the assertion is an atom.

Now, suppose that neither is an atom. One way of converting the matching problem into a simpler problem is by decomposing both the pattern and the assertion into substructures and checking corresponding parts for a match. The LISP primitives FIRST and REST provide an easy way of doing this. Suppose we tried matching the pattern:

((a ? b) ? (c d))

against:

((a a b) (x y) (c d))

The pattern does match the assertion; we would like the matching function to decompose it correctly. When applied to a list, the function FIRST selects the first element, and the function REST selects everything but the first element. We can think of the subparts of the patterns (and assertions) selected by FIRST and REST as patterns themselves. A pattern matches an assertion if and only if the FIRST of the pattern matches the FIRST of the assertion and the REST of the pattern matches the REST of the assertion. The FIRST of the pattern in the example is (A ? B) and the FIRST of the assertion is (A A B). These match. Similarly, the REST of the pattern is (?(C D)) and the rest of the assertion ((X Y)(C D)). These also match. By successively taking FIRST and REST of patterns and assertions, atomic elements must eventually be reached. We already know how to handle all forms of atomic arguments to the matching function. No other cases can occur. Let us list the various cases discussed:

- If the pattern is the atom ? then the match should succeed.
- If the pattern is another atom and is equal to the assertion, then the match should succeed.
- Otherwise, if the pattern is an atom the match should fail.
- If the pattern is not an atom but the assertion is, the match should fail.
- If neither the pattern nor the assertion is an atom, then the match should succeed if and only if the FIRST of the pattern and assertion match *and* the REST of the pattern and assertion match.

These conditions can be coded fairly directly into a LISP function to do this. Each of the above conditions becomes one clause in the conditional COND expression:

```
(DEF MATCH (PATTERN ASSERTION)
   (COND ((EQUAL PATTERN ' ?) T)
         ((AND (ATOM PATTERN)
               (EQUAL PATTERN ASSERTION)) T)
         ((ATOM PATTERN) NIL)
         ((ATOM ASSERTION) NIL)
         (T (AND (MATCH (FIRST PATTERN)
                        (FIRST ASSERTION))
                 (MATCH (REST PATTERN) (REST ASSERTION))))))
```

grams, these assertions should be collected together into a data base. In LISP, the easiest way of making a data base of objects is to make a list of them and let this list be the value of some variable. (There are more efficient ways of collecting assertions into a data base. These are described in the box.) As we discover more assertions that we would like to include in the program, they can be added to the list. Assertions can be just as easily removed if we determine the fact to be no longer valid. Two LISP functions, ADD and REMOVE, can be written to add assertions to and remove assertions from the data base. Any program that wanted to change the contents of the data base would make use of these two functions. A function call of:

(ADD ' (MOTHER-OF LENA HAROLD))

would add that one assertion to the data base. A function call of:

(REMOVE ' (MOTHER-OF LENA ARTHUR))

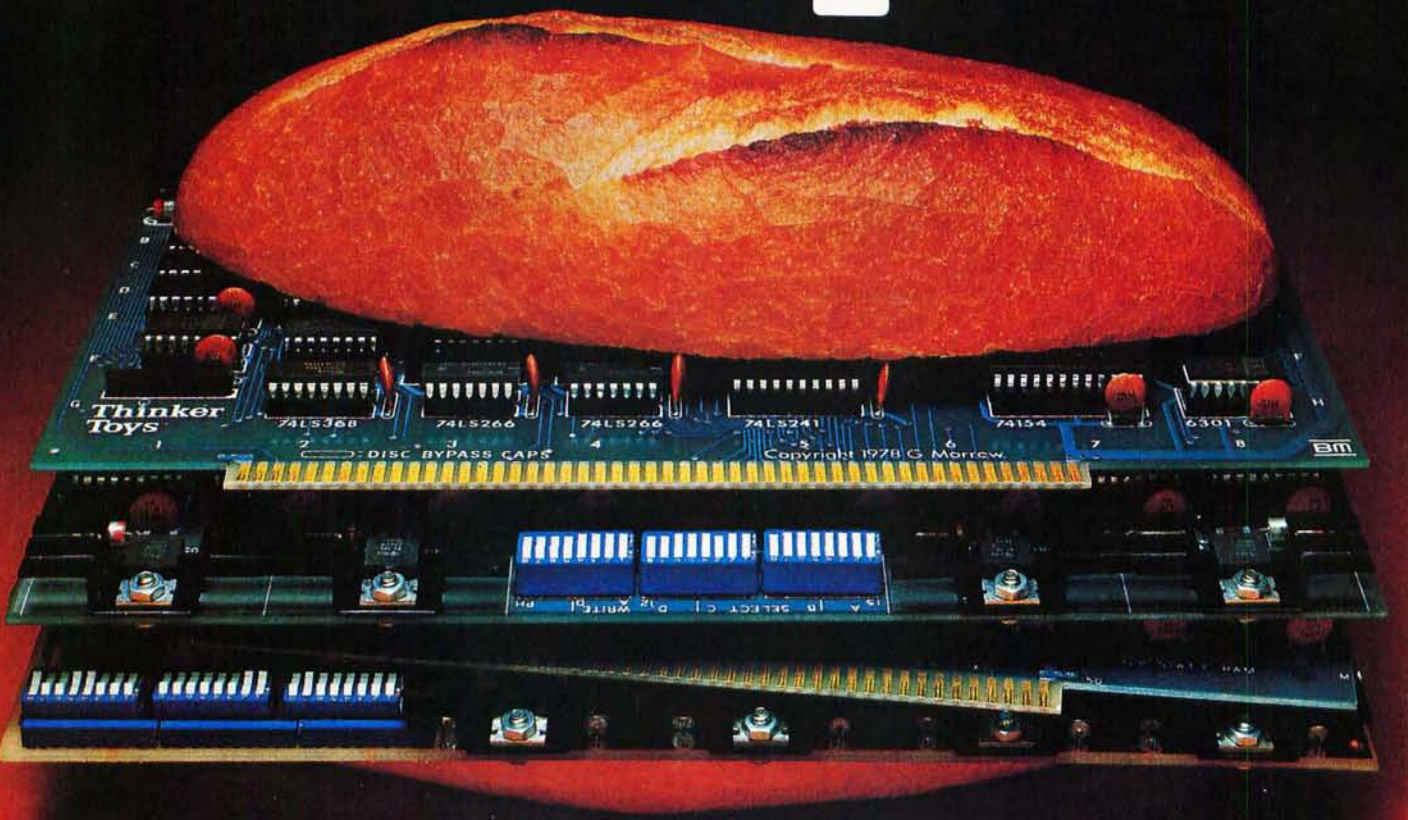
would remove that assertion from the data base.

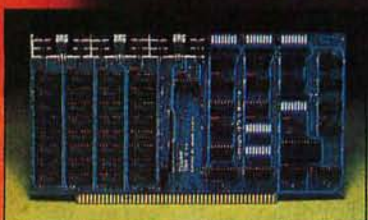
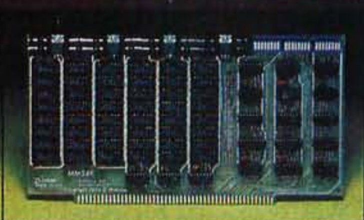
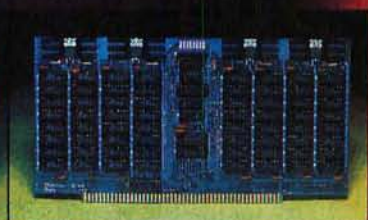
Next we will need some way to retrieve information stored in the data base. If we want to know whether or not Fay is the mother of Robert, the data base (really just a list) can be searched for the assertion:

(MOTHER-OF FAY ROBERT)

A function called RETRIEVE can do this easily. RETRIEVE takes one argument, an assertion, and returns T or NIL (*yes* or *no*) depending on whether or not the

# 2¢ a byte!




| | | | |
|---|---|---|---|
|  |  |  |  |
| MM16K/$349 = 2.1¢ | MM24K/$499 = 2¢ | SR16K/$299 = 1.8¢ | SR32K/$649 = 2¢ |

Now you can afford to sink your teeth into some big, feature-packed **static** memories. Because George Morrow's ultra-efficient designs have brought S-100 memory down to 2¢ a byte.

Introducing Morrow's new "MemoryMaster" Bank Select Logic memories, the top of the SuperRam™ line.

The SuperRam™ MemoryMaster 16K Static may be the most sophisticated S-100 memory at any price. The MM16K is switch-programmable to write-protect any of the four 4K blocks . . . or to open invisible 1K "windows" to accommodate VDM's or disk controllers. An on-board I/O device and jumper block allow you to use the memory-extending Bank Select Logic features of your software.

Yet, the SuperRam™ MemoryMaster 16K kit is just 2.1¢ a byte at $349. Assembled and tested, $399.

The SuperRam™ MemoryMaster is also available in 24K configuration: 3 individually write-protectable 8K blocks with Bank Select Logic capability. MM24K Kit, $499. Assembled and tested, $549.

Or, get your memory at a rock-bottom 1.8¢ a byte with the SuperRam™ 16K Static. It gives you 4 individual 4K blocks . . . plus the ability to switch-enable the Phantom Line for power-up sequencing. Kit, $299. Assembled and tested, $349.

But if you really need a big helping of memory, the SuperRam™ 32K Static serves up two individual 16K blocks for 2¢ a byte: $649 in kit. Assembled and tested, $699.

Whichever Morrow memory suits your taste, it will run perfectly in 2 MHz 8080, 4 MHz Z-80 or 5 MHz 8085 systems. And meets the Proposed IEEE S-100 Standard.

2¢ a byte! That's food for thought. And they're ready to take out at your local computer shop. Or if not, we deliver. Write Thinker Toys,™ 5221 Central Ave., Richmond CA 94804. Or call 415-524-2101 (10-4 Pacific Time any weekday).

*Morrow Designs*

**Thinker Toys**™

assertion is in the data base. To check for the existence of this particular assertion, we would execute:

```
(RETRIEVE ' (MOTHER-OF FAY ROBERT))
```

One of the nice features of LISP is that it is so easily *extensible*. It is possible to build languages on top of the basic LISP system that deal with higher level concepts as if they were primitives. The functions ADD, REMOVE, and RETRIEVE are three operations in a language we are building to manipulate assertions. So far, the language is very simple. The function RETRIEVE, for example, can only ask about *specific* assertions.

There are many more interesting questions that we would like the system to be able to answer, such as "Who is the mother of Robert?" In terms of these assertions this question could be answered by finding an assertion that has three elements, the first and third being the atoms MOTHER-OF and ROBERT, and the second element being anything at all. One way of saying this to the machine is by using a *pattern* such as:

```
(MOTHER-OF ? ROBERT)
```

where the ?s represent place holders, meaning that we will take anything in their positions.

One function, RETRIEVE, is modified to go down the list of assertions in our data base and compare the pattern with the individual assertions. If an assertion and a pattern *match*, the assertion will be returned as the value of RETRIEVE. Matching means that atoms in corresponding positions are the same, except for ?s in the pattern that require only that *something* be in the corresponding position in the assertion. Using our data base, the pattern given above will only match one assertion:

```
(MOTHER-OF FAY ROBERT)
```

By taking the second element of this list we will have found the mother of Robert. In general, more than one assertion in the data base can match a given pattern; it just happens that a person has only one mother, so we would not expect more than one assertion to tell us the mother of Robert. Suppose our question is "Who are the children of Fay?". We can make a pattern that represents this question by specifying a MOTHER-OF assertion with FAY in the mother position, and a ? in the child position:

```
(MOTHER-OF FAY ?)
```

The function RETRIEVE actually returns a *list* of all the assertions that match the given pattern so that it can accomodate the case where there is more than one match. Evaluation of the form:

```
(RETRIEVE ' (MOTHER-OF FAY ?))
```

should return:

```
((MOTHER-OF FAY ROBERT)
 (MOTHER-OF FAY ARLENE))
```

and can be further analyzed by a LISP function to extract the names of Fay's children.

The examples of assertions presented thus far have been in the form of a list of atoms. Assertions can be arbitrary pieces of list structure. The use of nested lists is an important tool for representing the structure inherent in the knowledge being represented. For example, we may wish to represent facts about the courses students have taken at a university. There might be one assertion for each student for each term he or she is registered. A possible record would be:

```
(COURSES BARBARA (SPRING 1978)
         (PHYSICS-2
          ALGEBRAIC-TOPOLOGY
          AESTHETICS))
```

The first element of the list designates it as a record of courses taken by a given student for a given term. This assertion expresses the fact that Barbara was registerd for the Spring term of 1978 and took three courses: Physics II, Algebraic Topology, and Aesthetics. With records of this kind and our pattern matcher we can ask various kinds of questions and have RETRIEVE return the list of assertions that pertain to the problem. Here are some examples:

*"Who was registered for courses in 1976?"*
(RETRIEVE ' (COURSES ? (? 1976) ?))

*"What courses did Sam take during his college career?"*
(RETRIEVE ' (COURSES SAM ? ?))

*"What courses did Barbara take in Spring of 1978?"*
(RETRIEVE ' (COURSES BARBARA (SPRING 1978)?))

There are certain questions that the simple pattern matcher we have described cannot address, such as "Who was registered for Algebraic Topology in the Spring of 1978?". More sophisticated schemes for pattern matching will be described later. A simple pattern matcher that can handle ?'s in patterns is very easy to write using the recursive control structures of LISP. It is described in the "Discrimination Networks" textbox.

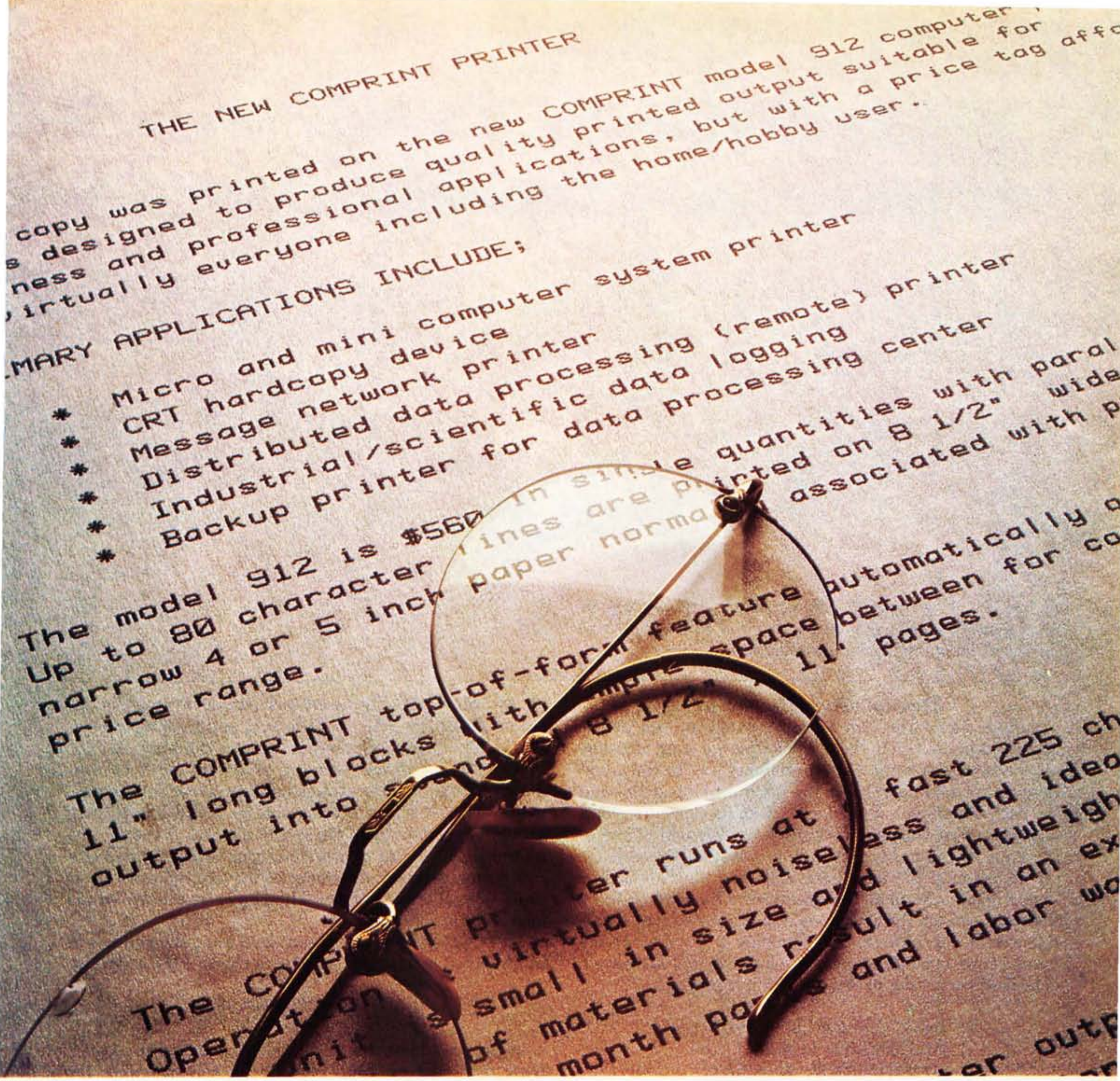
### Simple Deductions

There are a number of facts that are not explicitly contained in the data base of family relations described above that people can easily deduce. We might want to be able to answer the question "Who is the grandmother of Robert?". This question is posed to the system by the function call:

```
(RETRIEVE ' (GRANDMOTHER-OF ? ROBERT))
```

The data base contains no explicit GRANDMOTHER-OF assertions, so the function RETRIEVE, as defined thus far, would fail. The data base does contain enough facts that it is capable of answering this question. Looking at the assertions given earlier it is obvious that the answer is Lena. How do we arrive at this? First we find a

*Text continued on page 42*




# This exceptional print quality for $560!

**The new Comprint model 912 printer for computers and terminals:**

- Fast—225 characters/second (170 lpm)
- 80 character lines on 8½" wide paper
- Quiet, non-impact operation
- 6 month warranty

**Available now at computer stores and industrial distributors.**

comprint

**Computer Printers International, Inc.** 340 E. Middlefield Rd., Mt. View, CA 94043 • 415 969-6161

Circle 48 on inquiry card.

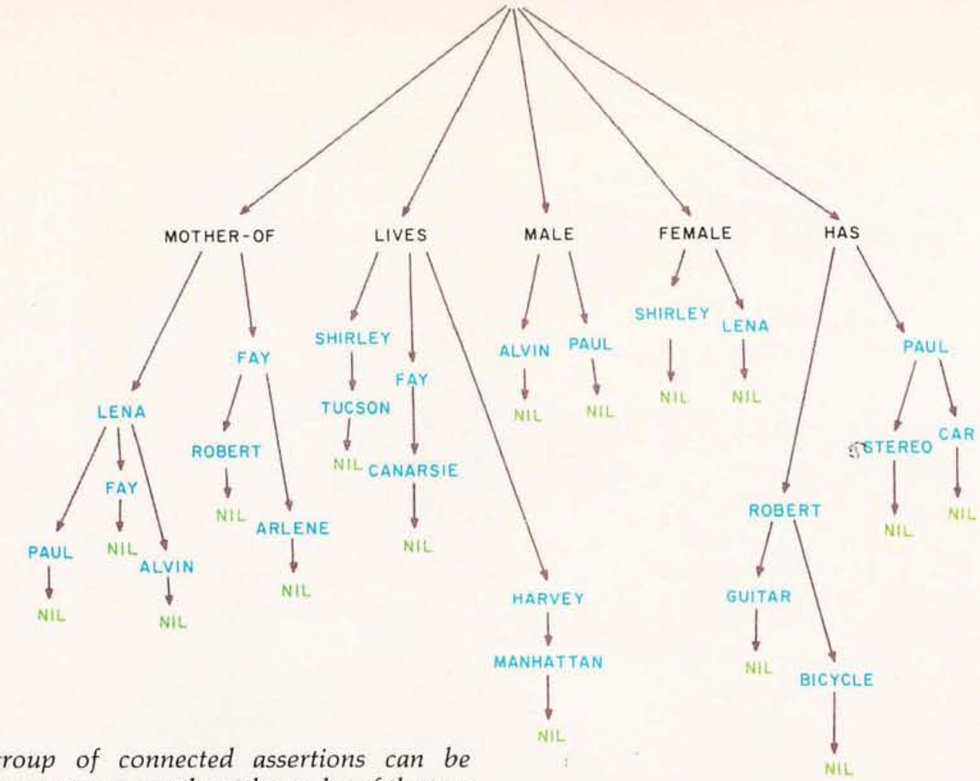



*Figure 1: A group of connected assertions can be represented by a tree structure where the nodes of the tree represent locations within an assertion.*

## Discrimination Networks

Simple data bases can be represented as lists of the assertions contained in them. Each time we want to determine whether or not a pattern matches any of the assertions in the data base, the entire list must be scanned and the pattern matcher applied to each of its elements. For a large data base this may take too long. We would like to represent the data base in such a way that the average search through the data base will take much less time than a linear scan of all the assertions. One way of doing this is to arrange the assertions into groups so that a partial test of the pattern can eliminate a number of the groups from consideration. Let's suppose that we have a data base consisting of the following assertions:

(MOTHER-OF LENA PAUL)
(MOTHER-OF LENA ALVIN)
(MOTHER-OF LENA FAY)
(MOTHER-OF FAY ROBERT)
(MOTHER-OF FAY ARLENE)
(LIVES SHIRLEY TUCSON)
(LIVES FAY CANARSIE)
(LIVES HARVEY MANHATTAN)
(MALE ALVIN)
(MALE PAUL)
(FEMALE SHIRLEY)
(FEMALE LENA)
(HAS ROBERT GUITAR)
(HAS ROBERT BICYCLE)
(HAS PAUL STEREO)
(HAS PAUL CAR)

One way of grouping these assertions, suggested by the given list, is by the first elements of the assertions. Thus, all the MOTHER-OF assertions would be together, as would the LIVES, MALE, FEMALE, and HAS assertions. If the first element of the pattern was the atom LIVES, then only one group of three assertions need be examined. Some of these groups can be further subdivided; the MOTHER-OF assertions can be divided into three groups depending upon the second element of the list (the mother). The group of assertions can be represented as a *tree* structure where the nodes of the tree represent locations within the assertion. The above assertions would appear as in figure 1.

This tree can be easily constructed using the pointers of LISP. When an attempt is made to check if the assertion:

(MOTHER-OF LENA FAY)

is in the data base, the root node is searched for a subnode marked with MOTHER-OF. If this is found, the search continues, otherwise a failure is reported. The pointer is followed to the MOTHER-OF node. This is then searched for a LENA subnode. This is found, the pointer followed, and a search is made for a FAY subnode. This also is found, and it contains a NIL subnode indicating that the assertion ends there. Tracing the path leading to this point gives the assertion. By representing the knowledge in this way, much of the data base no longer has to be searched to find what we want.

This can be extended to ? variables in patterns. Whenever we try to compare a ? against a node, *all* paths must be taken. And this example deals only with *flat* list structure (ie: lists of atoms). The concept can be extended to arbitrary list structure. The result is less intuitive and beyond the scope of this article. It is an interesting problem to think about.


**WE'RE ALTOS COMPUTER SYSTEMS. Our SUN-SERIES ACS8000 business/scientific computer creates a new standard in quality and reliability in high technology computers.**

**HIGH TECHNOLOGY** The ACS8000 is a single board, Z80®* disk-based computer. It utilizes the ultra-reliable Shugart family of 8 inch, IBM compatible, disk drives. A choice of drives is available: single or double density, single or double sided. Select the disk capacity you need, when you need it: ½M, 1M, 2M, or 4M bytes. The ACS8000 features the ultimate in high technology hardware: a fast 4 MHz Z80 CPU, 64 kilobytes of 16K dynamic RAM, 1 kilobyte of 2708 EPROM, an AMD 9511 floating point processor, a Western Digital floppy disk controller, a Z80 direct memory access, Z80 Parallel and Serial I/O (two serial RS232 ports, 1 parallel port), and a Z80 CTC Programmable Counter/Timer (real time clock). In essence, the best in integrated circuit technology.

**BUILT-IN RELIABILITY** The ACS8000 is a true single board computer. This makes it inherently reliable and maintainable. The board and the two Shugart drives are easily accessible and can be removed in less than five minutes. All electronics are socketed for quick replacement. Altos provides complete diagnostic utility software for drives and memory.

**QUALITY SOFTWARE** Unlimited versatility. The ACS 8000 supports the widely accepted CP/M®** disk operating system and FOUR high level languages: BASIC, COBOL, PASCAL and FORTRAN IV. All available NOW.

**PRICE** ACS 8000-1, single density, single-sided [½ Mb] $3,840
ACS 8000-2, double density, single-sided [1 Mb] $4,500
ACS 8000-3, single density, double-sided [1 Mb] $4,800
ACS 8000-4, double density, double-sided [2 Mb] $5,300

**Brackets show disk capacity per standard two drive system. All models come standard with 32 Kb RAM and two 8″ disk drives as shown above. Expansion to 64 Kb is $363 per 16 Kb. FPP, DMA, software optional. Dealer/OEM discounts available. Delivery: 30 days ARO, all models.**

Circle 6 on inquiry card.

*Z80 is a trademark of Zilog, Inc.
**CP/M is a trademark of Digital Research, Inc.

**ALTOS**
COMPUTER SYSTEMS **2338A WALSH AVENUE • SANTA CLARA • CA 95050 • (408) 244-5766**

*Text continued from page 38:*
MOTHER-OF or FATHER-OF assertion that gives a parent for Robert. Here we end up with:

(MOTHER-OF FAY ROBERT)

Then we take that parent (eg: FAY) and find a MOTHER-OF assertion with that parent in the child position, giving:

(MOTHER-OF LENA FAY)

The individual in the mother position of that assertion is the desired grandmother. To incorporate this kind of knowledge in the system, the language is augmented with procedures that explain how to *derive* certain facts if they are not in the data base. There are two GRAND-MOTHER-OF derivation procedures; one that checks for mothers of fathers, and one that checks for mothers of mothers. They might be expressed as:

```
(TO-DERIVE (GRANDMOTHER-OF ?X ?Y)
           (FIND (MOTHER-OF ?Z Y))
           (FIND (MOTHER-OF X Z)))

(TO-DERIVE (GRANDMOTHER-OF ?X ?Y)
           (FIND (FATHER-OF ?Z Y))
           (FIND (MOTHER-OF X Z)))
```

The first procedure looks for the mother of the person in the third slot (eg: the grandchild), and then her mother; the second procedure for the father of that person, and then his mother. We have added a little more complexity to the simple patterns described earlier. These patterns have *variables* associated with the question marks. The first pattern in these procedures expresses, in effect, what the procedure can do. It says "If you want to determine if someone is the grandmother of someone else, try the following." In order for the rest of the procedure to know who these people are, it must *bind* the names to variables. RETRIEVE has to be extended again. In addition to checking the data base for already known facts, it checks a library of procedures for those whose patterns match the request, trying them one at a time. When we execute the RETRIEVE function, trying to find the grandmother of Robert, the pattern:

(GRANDMOTHER-OF ? ROBERT)

is matched against the head pattern in the TO-DERIVE construct:

(GRANDMOTHER-OF ?X ?Y)

The match is successful. Y will get the value ROBERT, and X the value ? (really no value at all, just a place holder). The first line causes the system to find an assertion that has MOTHER-OF in the first position and ROBERT, the value of Y, in the last line. Whatever is found in the second position is assigned to the variable Z. For our particular data base, the assertion:

(MOTHER-OF FAY ROBERT)

will be found and Z will get the value FAY. When the next line is executed, a MOTHER-OF assertion is looked for with FAY in the third position, and anything at all in the middle. (Remember X has the value ?.) The assertion it will find is:

(MOTHER-OF LENA FAY)

What we have just done is *derived* the fact:

(GRANDMOTHER-OF LENA ROBERT)

Here is a procedure to determine whether or not one individual is the uncle of another:


A Beautiful Way To Interface
IQ 140
SOROC's first and foremost concern, to design outstanding remote video displays, has resulted in the development of the IQ 140. This unit reflects exquisite appearance and performance capabilities unequaled by others on the market.
With the IQ 140, the operator is given full command over data being processed by means of a wide variety of edit, video, and mode control keys, etc.
The detachable keyboard, with its complement of 117 keys, is logically arranged into 6 sections plus main keyboard to aid in the overall convenience of operation. For example, a group of 8 keys for cursor control / 14 keys accommodate numeric entry / 16 special function keys allow access to 32 pre-programmed commands / 8 keys make up the extensive edit and clear section / 8 keys for video set up and mode control / and 8 keys control message and print.
Two Polling options available: 1) Polling compatible with Lear Siegler's ADM-2. 2) Polling discipline compatible with Burroughs.
IQ 120
The SOROC IQ 120 is the result of an industry-wide demand for a capable remote video display terminal which provides a multiple of features at a low affordable price. The IQ 120 terminal is a simple self-contained, operator / computer unit.
The IQ 120 offers such features as: 1920 character screen memory, lower case, RS232C extension, switch selectable transmission rates from 75 to 19,200 bps, cursor control, addressable cursor, erase functions and protect mode. Expansion options presently available are: block mode and hard copy capability with printer interface. The IQ 120 terminal incorporates a 12-inch, CRT formatted to display 24 lines with 80 characters per line.
Circle 341 on inquiry card.
SOROC TECHNOLOGY, INC.
165 FREEDOM AVE., ANAHEIM, CALIF. 92801
(714) 992-2860 / (800) 854-0147

**The system chains backwards through facts until it finds some simple ones it knows.**

```
(TO-DERIVE (UNCLE-OF ?X ?Y)
           (FIND (SIBLING ?Z X))
           (FIND (CHILD-OF Y Z)))
```

*"To show one person is the uncle of another, find a person that is a sibling of the first and a parent of the second."*

This procedure would work if we had SIBLING and CHILD-OF assertions in the data base. Since we don't, we must specify procedures that can determine these things from the information that is in the data base:

```
(TO-DERIVE (SIBLING ?X ?Y)
           (FIND (MOTHER-OF ?Z X))
           (FIND (MOTHER-OF Z Y)))
```

*"To determine if one person is the sibling of another, see if they have the same mother."*

```
(TO-DERIVE (CHILD-OF ?X ?Y)
           (FIND (MOTHER-OF Y X)))
```

*"To determine if one person is the child of another, see if the second is known to be the mother of the first."*

```
(TO-DERIVE (CHILD-OF ?X ?Y)
           (FIND (FATHER-OF Y X)))
```

*"To determine if one person is the child of another, see if the second is known to be the father of the first."*

There are now two different procedures for deciding CHILD-OF relations as was the case with the earlier GRANDMOTHER-OF relation. If the system doesn't already have the answer to the question in its data base, it will try one, and if that fails, it will try the other.

Our set of assertions does not happen to contain FATHER-OF assertions, so they too should be specified by procedures. We do have MOTHER-OF and HUSBAND-OF assertions. These are sufficient:

```
(TO-DERIVE (FATHER-OF ?X ?Y)
           (FIND (MOTHER-OF ?Z Y))
           (FIND (HUSBAND-OF X Z)))
```

*"To determine if one person is the father of another see if the second person's mother is the husband of the first."*

The control used by this system is often referred to as *backward chaining*. Determining if someone is the uncle of someone else may result in attempts to determine CHILD-OF relations that may then result in determining FATHER-OF and then HUSBAND-OF relations. The sys-

# MOVING DATA AT A SNAIL'S PACE BECAUSE YOU'RE FLOPPY BOUND?

*Let Corvus Systems put you back in the race!*


- **For TRS-80†, Apple‡, S-100 Bus*.**
- **Fully compatible hardware/software.**
- **10-million byte disk: IMI-7710.**
- **Proven Winchester technology.**
- **Z-80 based Corvus disk controller.**
- **Comprehensive disk diagnostics.**
- **Up to 4 disks per system.**
- **System $5350, add-on disk $2990.**
- **30 day delivery**

**Corvus offers a complete systems solution to the mass storage problem of micro computers. In a package smaller than a briefcase, we provide an intelligent controller, disk, and personality module. Call or write today for additional information.**

**Get up to speed with Corvus.**

†TRS-80 is a registered trademark of Radio Shack, a Tandy Co.
‡Apple is a registered trademark of APPLE Computers, Inc.
*S-100 pricing slightly higher.

**CORVUS SYSTEMS, Inc.**

900 S. Winchester Boulevard
San Jose, California 95128
408/725-0920

Circle 83 on inquiry card.

tem *chains backward* through facts until it finds some simple ones that it knows.

The TO-DERIVE procedures are similar in concept to *subroutines* in many other computer languages. The difference is that subroutines are usually called by *name*. If I want to compute a cosine I call the subroutine COS. Procedures in these languages are *invoked by a pattern* that indicates what they can accomplish. The procedure that determines if one person is the uncle of another has no name; it indicates by its pattern (UNCLE-OF ?X ?Y) that it is capable of determining whether or not one person is the uncle of another. This distinction is an important one. As shown, more than one procedure may have the same pattern. This will not disturb the system. It will try one, and if that fails, it will try others until it finds one that works. One TO-DERIVE procedure can serve several purposes. The UNCLE-OF procedure is capable of answering three different kinds of questions:

*"Is Harold the uncle of Robert?"*
*"Who are the nephews of Harold?"*
*"Who are the uncles of Robert?"*

**Better Pattern Matchers**

The ease with which concepts can be expressed in the language depends significantly on the sophistication of the pattern matcher. The pattern matcher described so far is of the simplest kind. Many things we would like to say are difficult or impossible to do with it. There is no such thing as an "ideal pattern matcher." One can always come up with more sophisticated ways to create patterns. This section is devoted to discussing two fairly well known extensions known as *unpack* and *multisets*.

Earlier we were concerned with a data base of assertions representing information about students taking courses at a school. The assertions were of the form:

```
(COURSES BARBARA (SPRING 1978)
         (PHYSICS-2
          ALGEBRAIC-TOPOLOGY
          AESTHETICS))
```

and it was impossible to phrase questions of the form Who took Algebraic Topology in the Spring of 1978?" The reason that this is impossible to indicate is that the atom ALGEBRAIC-TOPOLOGY can occur as any element of a list with zero or more atoms in this list, before and after it. The problem can be dealt with by the introduction of the *unpack operator*. This operator, represented by an exclamation point !, is placed before the question mark variable. A ? without a ! matches *exactly one* object. A !? combination will match *zero or more* objects. Here are some examples of patterns:

(FOO !? BAR) matches any list that begins with the atom FOO and ends with the atom BAR:

```
(FOO BLATZ BAR)
(FOO TOM LARRY BAR)
(FOO BAR)
```

(FOO !?) matches any list with FOO as the first element:

```
(FOO)
(FOO BAR)
(FOO BAR BLATZ)
```

(? !? FOO !?) matches any list that contains the atom FOO as the second or later member:

```
(XYZ FOO)
(XYZ ABC FOO TOM LARRY)
```

With the unpack operator the question "Who took Algebraic Topology in the Spring of 1978?" can be phrased:

```
(RETRIEVE ' (COURSES ? (SPRING 1978)
                     (!?ALGEBRAIC-TOPOLOGY !?)))
```

Of course, if we were using the unpack operator inside TO DERIVE procedures, the !? would be followed by a variable that gets bound to what it matches, just as the ? variables.

Another question we cannot ask with the simple pattern matcher is "Who took Algebraic Topology and Aesthetics in the Spring of 1978?" We cannot ask this question because whenever we have a list there is an intrinsic order to its elements. To be sure of covering all cases we would need two patterns:

```
(COURSES ? (SPRING 1978)
           (!? ALGEBRAIC-TOPOLOGY !?
            AESTHETICS !?))
```

as well as:

```
(COURSES ? (SPRING 1978)
           (!? AESTHETICS !?
            ALGEBRAIC-TOPOLOGY !?))
```

If there were three courses then six different patterns would be necessary. We need a more general solution. To handle the case where matches should be made regardless of the order of the elements, *multisets* are introduced. A multiset will be denoted by curly brackets { and }. A multiset is said to match a list if each of its elements match a corresponding element of the list (?s and !?s are allowed). Here are some examples of multisets:

{A B C} will match any list containing exactly the three elements A, B, C:

```
(A B C)
(C A B)
(B C A)
```

{A B ?} will match any three element list containing A and B:

```
(B C A)
(BAR A B)
(A X B)
```

{A B !?} will match any list containing A and B:

```
(B A)
(X B Y Z A V)
```

{B B !?} will match any list containing two or more occurrence of B:

(X B A B)
(B W S FOO B BAR)

The question "Who took Algebraic Topology and Aesthetics in the Spring of 1978?" can now be phrased:

```
(RETRIEVE ' (COURSES ? (SPRING 1978)
             {ALGEBRAIC-TOPOLOGY
              AESTHETICS !?})
```

**History**

The basic concepts of pattern-directed invocation originated in the PhD thesis of Carl Hewitt at MIT in 1969. The original Planner language that was the subject of his thesis was never implemented. A cut down version of Planner, roughly equivalent to our language with ADD, REMOVE, and RETRIEVE, was implemented in 1970 and called Microplanner.

Microplanner was used as a tool in subsequent research in artificial intelligence at MIT. The best known system to make use of Microplanner was the SHRDLU program of Terry Winograd. SHRDLU was a program about a simulated world consisting of a table, variously colored toy blocks, and a box. A person could type in English language questions and imperatives to which the system would take an appropriate action, such as: "What blocks are in the box?" or "Pick up the big red block." Assertions were used to store knowledge about the current state of the world, such as:

(IN PYRAMID BOX)

Procedures implemented simple reasoning involved with answering questions and constructing plans to carry out commands. Microplanner proved to be quite limited in its capabilities and spawned several immediate successors that embodied sophisticated improvements.

QA4, developed by Rulifson and associates at the Stanford Research Institute, introduced the notion of multiple *contexts*. Contexts are a way of having more than one data base inside the machine, each representing a different aspect of the problem at hand. One context might model (ie: contain assertions pertaining to) the state of the world at some point of time in the past, while another might model the current state of the world. Another common use of the context mechanism is to reason about *hypothetical worlds*, collections of assertions similar but not identical to the current one. A hypothetical world might represent what would happen *if* the machine took some action.

Conniver, developed by Sussman at MIT, introduced certain notions of control structure that seemed lacking in the original Microplanner. The system has a data base of facts and procedures that are capable of deducing facts that are not explicitly in the data base. When a call is made to RETRIEVE, it is entirely up to the system to choose which procedures to try, and in what order to try them. The simple minded scheme picks one procedure and tries it. If this does not work it picks another. There is no way in Microplanner that a program can have control over the order in which procedures are chosen. Conniver supplies facilities that allow the program to have access to possible choices and then order or otherwise process them.

AMORD, developed by deKleer and associates at MIT, keeps a trace, by means of *justifications*, of how each fact in the data base was derived. If a fact is determined to be no longer valid, all facts that derived from it, as determined from the justifications, are automatically removed by the system. This facility allows a program to conveniently change certain premises and automatically update the rest of the data base to reflect this change.

ETHER, developed by the author, allows the program writer to let many operations in the program be done in parallel. The program can maintain conflicting world models (ie: collections of assertions) that can be reasoned about concurrently.

**A General Information Storing Tool**

These languages have been developed explicitly as artificial intelligence research tools. We have not discussed in any detail the issues involved in modeling a situation in the world and reasoning about it. The examples given are meant to suggest the possibilities for pattern directed invocation as a more general tool for storing facts. The need to store facts (ie: to create data bases) comes up in all sorts of situations. As computation becomes cheaper, more and more stores of information will move from paper to electronic storage media. There are, of course, more efficient ways to store information than by representing them in list structure in a LISP environment. The disadvantage of some loss of efficiency seems to be far outweighed by the increased flexibility in accessing the information. ■

# Anyone Know the Real Time?

Steve Ciarcia
POB 582
Glastonbury CT 06033



I'm sure you've all heard the term *real-time*, such as a real-time operating system. But, how many really understand its meaning? A simple definition of a real-time system is: a system that operates in real time, that is, it responds to the need for action in a period of time proportional to the urgency of the need; first things are done first. In control applications the system can be depended on to provide the information necessary to base time-dependent decisions on information that is up to date as of the minute or the hour. Real time describes the processing of information in a sufficiently rapid manner that the results of the processing are immediately available to influence control of the process being monitored.

While there are particular architectural enhancements in high-speed process monitoring and control systems, basically any computer can be configured to perform some semblance of real-time operations. The essential criterion is that the computer be capable of performing a specific action at a particular time. The extent of real-time operation then becomes dependent upon execution speed. If a program that takes 1 second to analyze a data input and display it on the video display is to run in real time, it can only be called once per second. For continuous sampling this also means that the computer cannot be tied up doing any other task without provision being made for that program to be interrupted so that the analysis program can run. Most often, computers utilize hardware priority interrupts to provide this capability. A direct benefit of this approach is that all programs can execute asynchronously, since interrupt logic synchronizes the computer's action upon the occurrence of a real-time event. Further discussion of interrupts will continue later in this article.

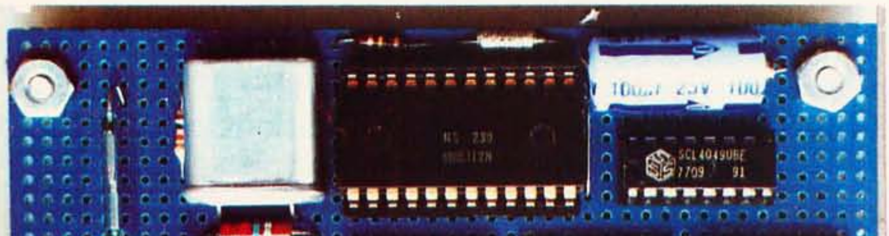
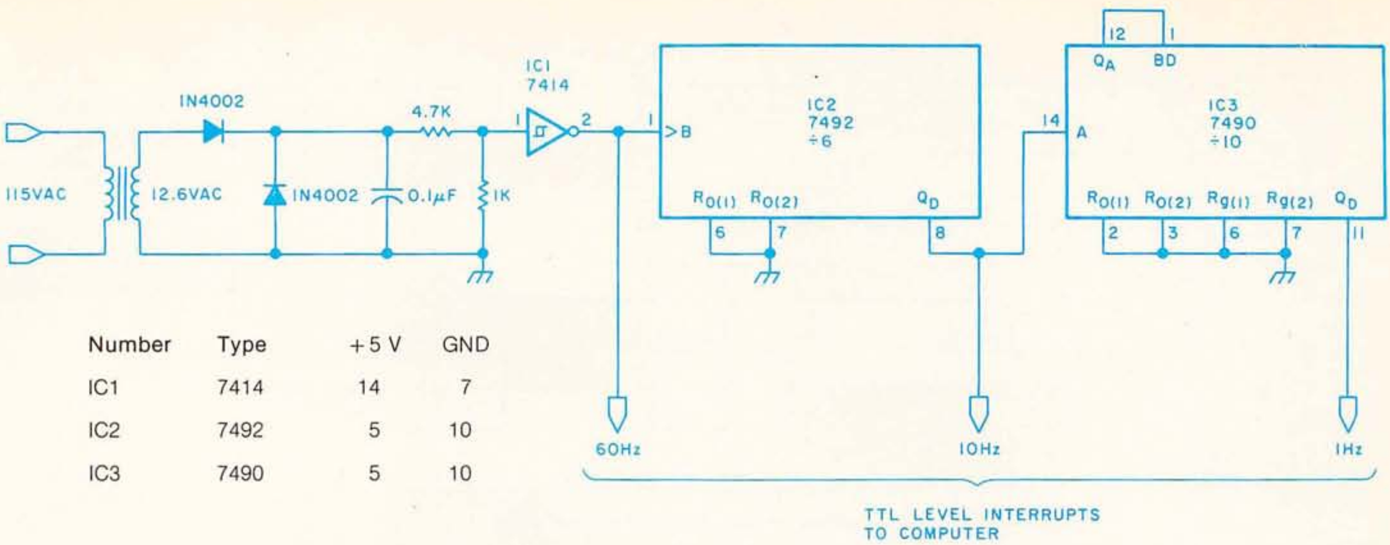


*Photo 1: A prototype board of real-time clock mounted on the back of an existing parallel I/O (input/output) board. Two reed switches on the left side of the board are for manual setting of the clock. The empty sockets are used for the particular application for which this board was designed, a home security system.*

A second, slightly less complex method of synchronizing computers to real-time events is through a technique of *status scanning* (or device polling). This software-intensive situation requires that all devices demanding real-time interaction set status flags to indicate ready conditions. The computer scans these flags periodically and performs the appropriate action. The flags are reset when the devices have been serviced. It is important to keep in mind that all the programs that the computer normally executes must be short enough to allow the computer to service every device. Also, care must be taken to design the system so that a second event cannot occur on an individual device before the computer has acknowledged the first event.

Most sophisticated real-time sys-




| Number | Type | +5 V | GND |
|---|---|---|---|
| IC1 | 7414 | 14 | 7 |
| IC2 | 7492 | 5 | 10 |
| IC3 | 7490 | 5 | 10 |

*Figure 1: A simple time-base generator for an interrupt-driven real-time clock.*

tems use a combination of these two methods. A clock circuit, such as that in figure 1, provides a time "tick" to the processor's nonmaskable interrupt line. This can be every 60th, 10th, or 1 second, as suggested in this schematic. When the computer acknowledges the interrupt, it first saves all registers from the program it was executing, and then services the real-time interrupt. Frequently the first action is to increment an internal counter which keeps track of elapsed time. Usually it will be a value equivalent to the total number of clock ticks, whether in seconds or milliseconds. Once this regular interval has been established, it is easy for the computer to scan all status flags from real-time devices. The addition of more real-time activities for the processor does not entail multiplying the number of interrupt lines, but rather it simply entails placing another status flag on the list of those to be checked on each clock tick.

The choice of a totally interrupt-driven real-time system, a combina tion scan and interrupt type, or a total scanning system is dependent upon the quantity of real-time operations and their frequency. An interrupt-driven system can process information faster than the same system configured for real-time scanning.

## Real Time Applications for Personal Computers

So far I have emphasized the system attributes, but nowhere have I discussed applications, particularly personal computing applications. Clock divisions down to milliseconds sound great and make interval timing extremely accurate, but I doubt that the majority of home computerists would want something that complex to integrate into their system. If my mail is any indication of this, they would prefer the design of a real-time clock which can be directly applied in home control applications. Automatically turning on the percolator at 6:45 AM would be far more stimulating than a high-speed data acquisition system which few would need.

## Build a Real-Time Clock

Essentially, the kind of real-time system which might appeal to personal computer users is one with a resolution of perhaps 1 minute rather than 1 ms. It should be read directly in hours and minutes like a 4- or 6-digit clock and not just total counts. A direct benefit of low resolution is reduced overhead. The computer does not have to acknowledge the clock update or scan status flags as often. It may not seem like much of a time savings, considering instruction execution speeds of 1 µs. However,

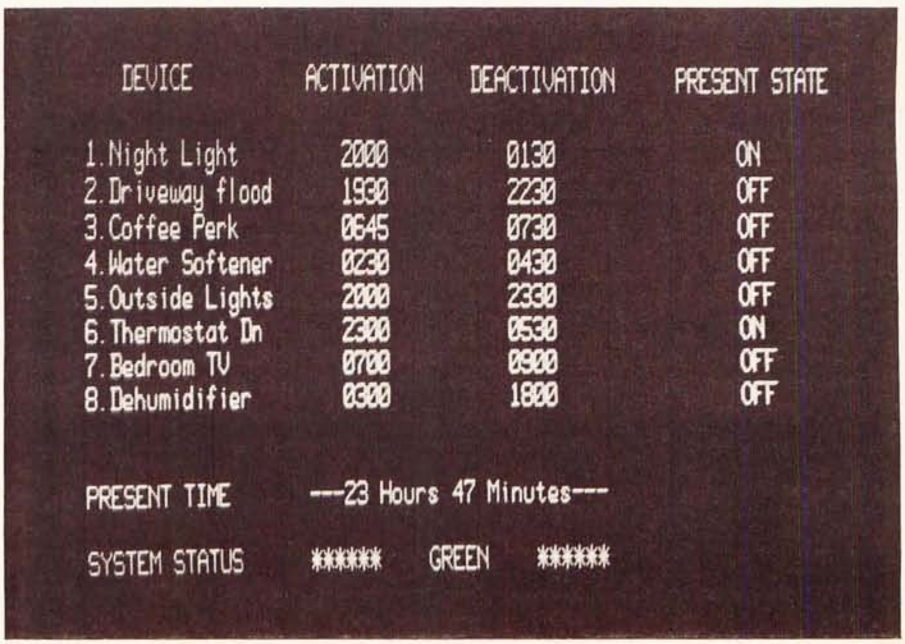



*Photo 2: A typical application of a real-time clock. This display is from my computer-controlled security system.*

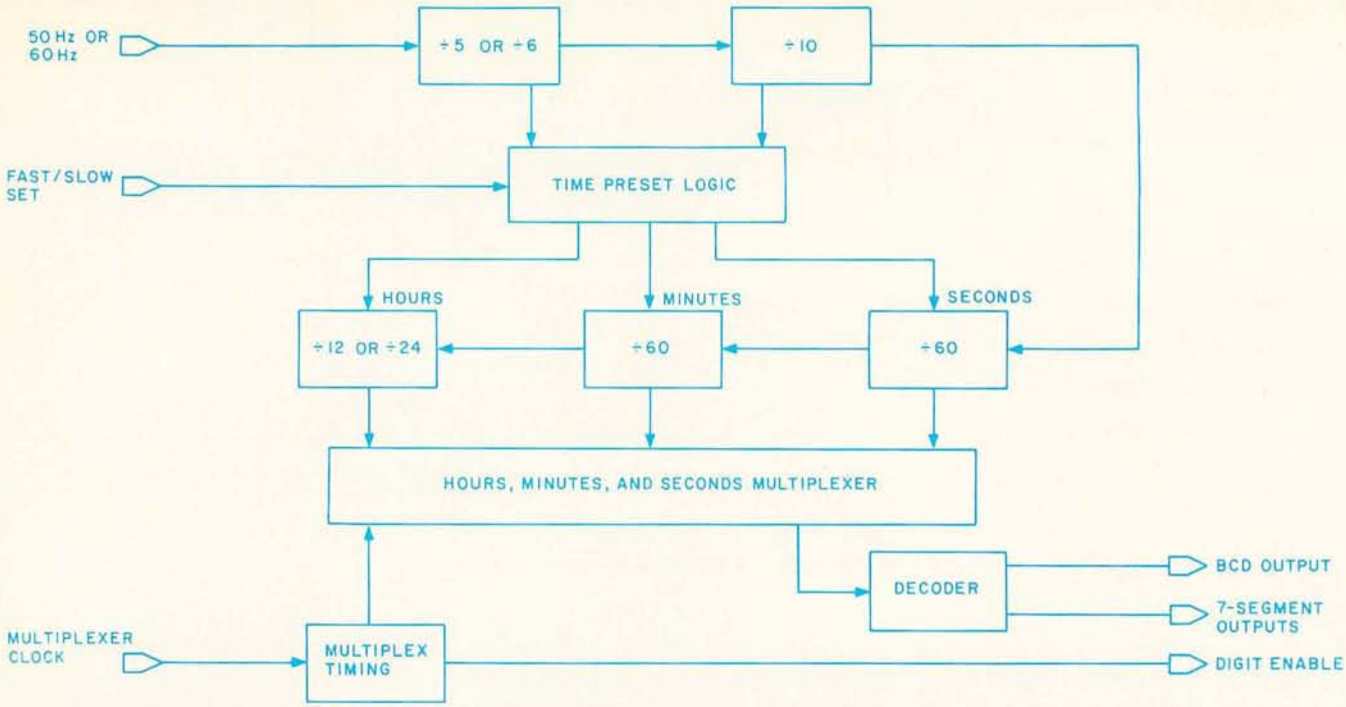
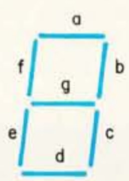



*Figure 2: The block diagram for a typical clock chip.*

TYPICAL 7-SEGMENT CLOCK DISPLAY




| | SEGMENTS | | | | | | | BCD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | f | g | 8 | 4 | 2 | 1 |
| 0 | × | × | × | × | × | × | | 0 | 0 | 0 | 0 |
| 1 | | × | × | | | | | 0 | 0 | 0 | 1 |
| 2 | × | × | | × | × | | × | 0 | 0 | 1 | 0 |
| 3 | × | × | × | × | | | × | 0 | 0 | 1 | 1 |
| 4 | | × | × | | | × | × | 0 | 1 | 0 | 0 |
| 5 | × | | × | × | | × | × | 0 | 1 | 0 | 1 |
| 6 | | | × | × | × | × | × | 0 | 1 | 1 | 0 |
| 7 | × | × | × | | | | | 0 | 1 | 1 | 1 |
| 8 | × | × | × | × | × | × | × | 1 | 0 | 0 | 0 |
| 9 | × | × | × | | | × | × | 1 | 0 | 0 | 1 |

*Figure 3: A comparison of output codes from 7-segment and BCD (binary coded decimal) clock chips.*

the interrupt routine could be 30 bytes and 100 μs long. If called every millisecond it would eat up 10% of the total cycle time—just to increment a counter! When it comes to real time, be careful not to byte (sic) off more than you can process.

The easiest way to provide an hourly and minute by minute input is to interface the computer to an MOS/LSI (metal oxide semiconductor/large scale integrated) clock device such as that found in most digital clocks or watches. The block diagram of a typical clock chip is shown in figure 2. This LSI device replaces about 22 TTL (transistor-transistor logic) chips once necessary to perform the same function, and consumes very little power, allowing battery standby operation. The circuit of figure 1 uses inexpensive TTL rather than CMOS (complementary metal oxide semiconductor) because battery backup is irrelevant if the computer cannot acknowledge interrupts in a powered down state. Figure 3 illustrates the logic of the BCD (binary coded decimal) and 7-segment output lines.

There are two approaches to the design of a clock interface. One approach is to let the clock circuit operate independently from the computer, attached in such a way that the computer is able to monitor this acti-

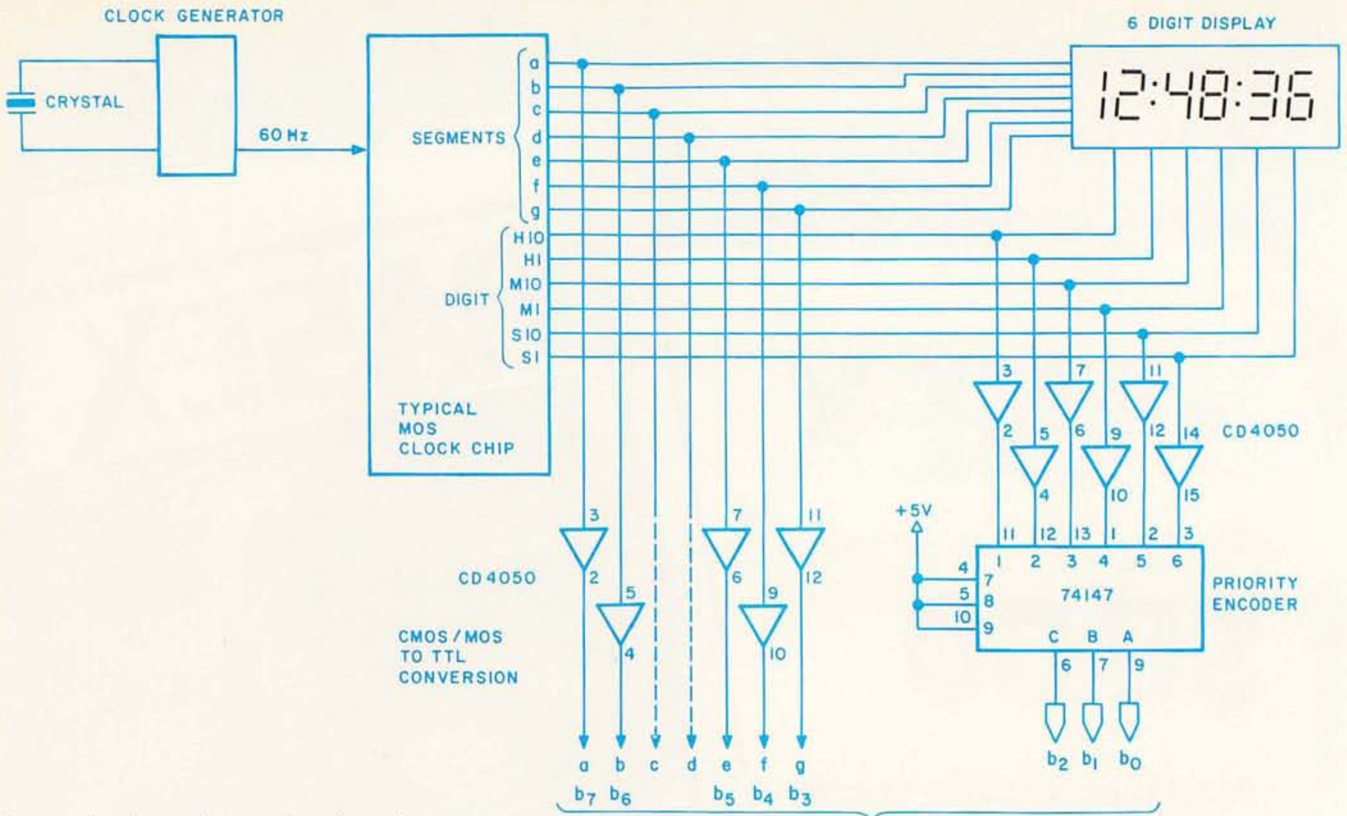



*Figure 4: A real-time interface implemented using a typical clock chip.*

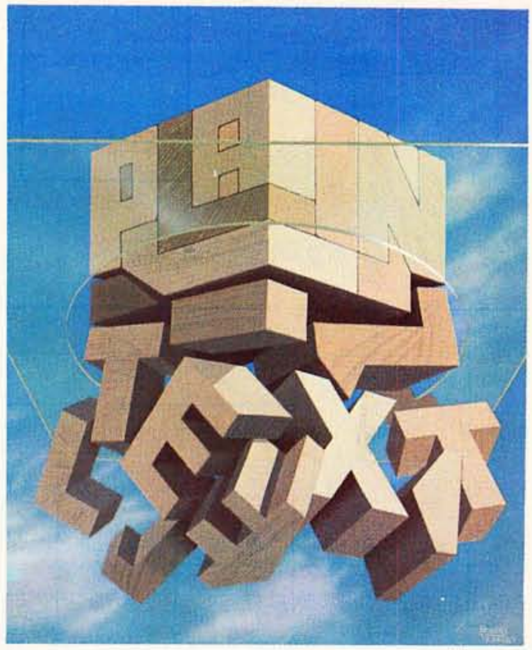
vity and extract a time value. The second approach, which I prefer, is to give the computer complete control over the information flow of the clock in a synchronous manner. This design makes the interface speed independent and allows it to be used directly with high-level languages.

Figure 4 shows the typical real-time clock interface. In this design the clock is configured in the usual manner to drive a 6-digit light emitting diode display. The clock runs independently with the display multiplexing rate (about 1 kHz) set by a resistor/capacitor combination attached to the chip. Five of the 7-segment drive lines are level shifted and buffered for TTL through a CD 4050, and the 6 digit lines are priority encoded to produce a 3-bit binary value for transmission to the computer of the energized digit-enable line. The 3-bit digit and 5-bit segment codes are combined to produce a single 8-bit byte interfaced to a parallel input port.

In operation, the computer program first looks at bits $b_0$ thru $b_2$ to determine which digit of the display to activate. Then it reads bits $b_3$ thru $b_7$ and compares them to a table to

250/750 BREAKING THE SOUND BARRIER ROBERT TINNEY

September 1977



250/750 THE TRAP DOOR ROBERT TINNEY

March 1979

# Byte Cover Prints -- Limited Editions.

The September '77 and March '79 covers of BYTE are now each available as a limited edition art print, personally signed and numbered by the artist, Robert Tinney.

These prints are strictly limited to a quantity of 750 for each cover, and no other editions, of any size, will ever be published. Each print is 18" x 22", printed on quality, coated stock, and signed and numbered in pencil at bottom.

The price of each print is $25. This includes 1) a signed and numbered print; 2) a Certificate of Authenticity, also signed personally by the artist and witnessed, attesting to the number of the edition (750), and the destruction of the printing plates; and 3) first class shipment in a heavy-duty mailing tube.

To order your limited edition art print, fill out and mail the order form below.

Send me ______ "Breaking the Sound Barrier" prints at $25 each, and ______ "Trap Door" prints at $25 each. I understand this price includes Certificate of Authenticity and first class shipment.

☐ I have enclosed check or money order to Robert Tinney Graphics.

☐ Charge this to my Master Charge or Visa

Card #________________ Expires:______

Ship my print(s) to:

Name________________________

Address________________________

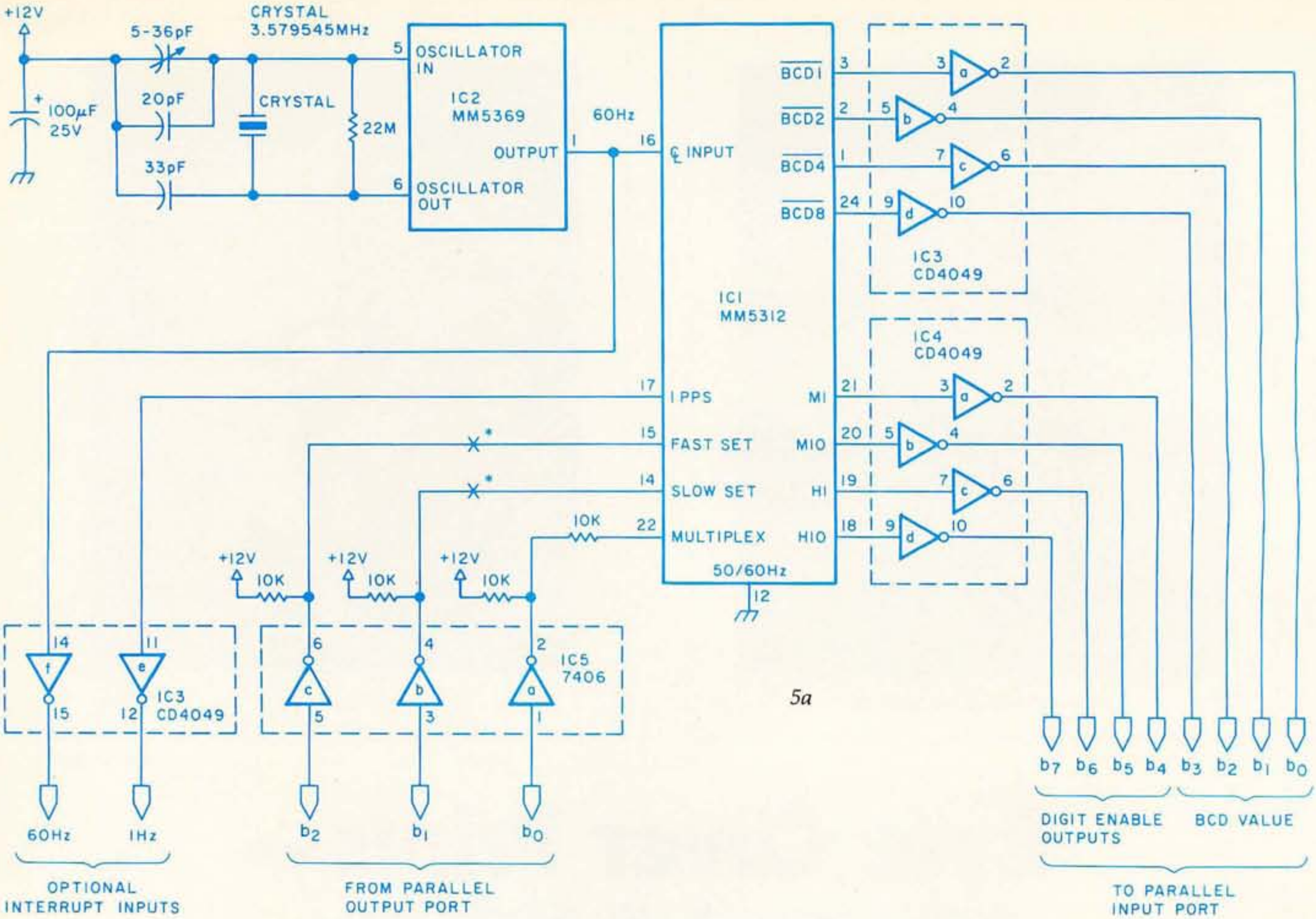
City________________ State______ Zip________

Send order to:

**robert tinney graphics**

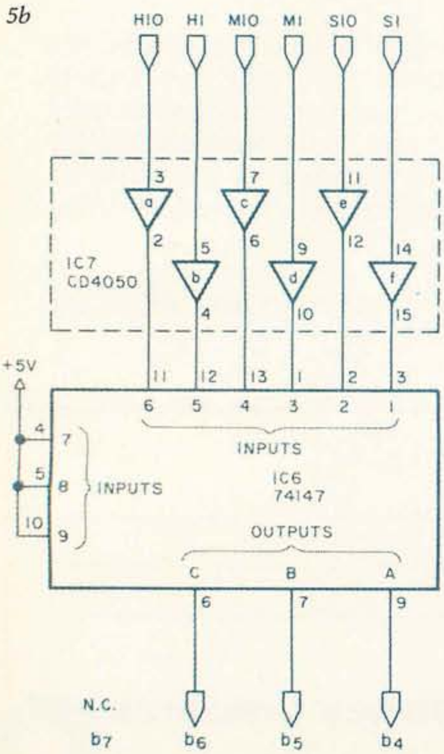
P.O. Box 45047 • Baton Rouge, LA 70895

Circle 369 on inquiry card.

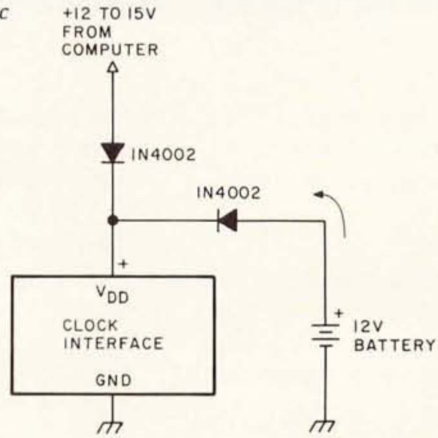



5a

5b




5c




| Number | Type | +5 V | GND | +12 V |
|---|---|---|---|---|
| IC1 | MM5312 | — | 23 | 13 |
| IC2 | MM5369 | — | 8 | 2 |
| IC3 | CD4049 | 1 | 8 | — |
| IC4 | CD4049 | 1 | 8 | — |
| IC5 | 7406 | 14 | 7 | — |
| IC6 | 74147 | 16 | 8 | — |
| IC7 | CD4050 | 1 | 8 | — |

*Figure 5: Design for a real-time clock which can be synchronously controlled by a BASIC or machine language routine. 5a shows the schematic diagram; asterisks indicate lines which should be opened to prevent loss of time data when the computer is powered down and the interface is used with battery backup. 5b shows an alternate configuration for a 6-digit clock when using an MM5311 integrated circuit. 5c shows the circuit for battery backup operation. The clock interface requires 12 mA from the battery during standby (indicated by the arrow).*

determine which character is being displayed. (Only 5 of the 7 segments are necessary to perform this comparison.) This process is repeated 5 more times as the chip sequences through the other digits. The final result is formatted into hours, minutes, and seconds. The entire operation takes about 10 ms and requires that the program be written in machine language.

If you can believe the claims of the manufacturers, there are now more computers in use that run BASIC rather than machine language as their primary mode of interactive communication. While it is still possible to manipulate individual bits and write machine language device control subroutines for these computers, their owners are obviously more familiar with high-level languages and would necessarily feel more comfortable with a clock design which could be controlled in BASIC as well as machine code. Figure 5 demonstrates such a design.

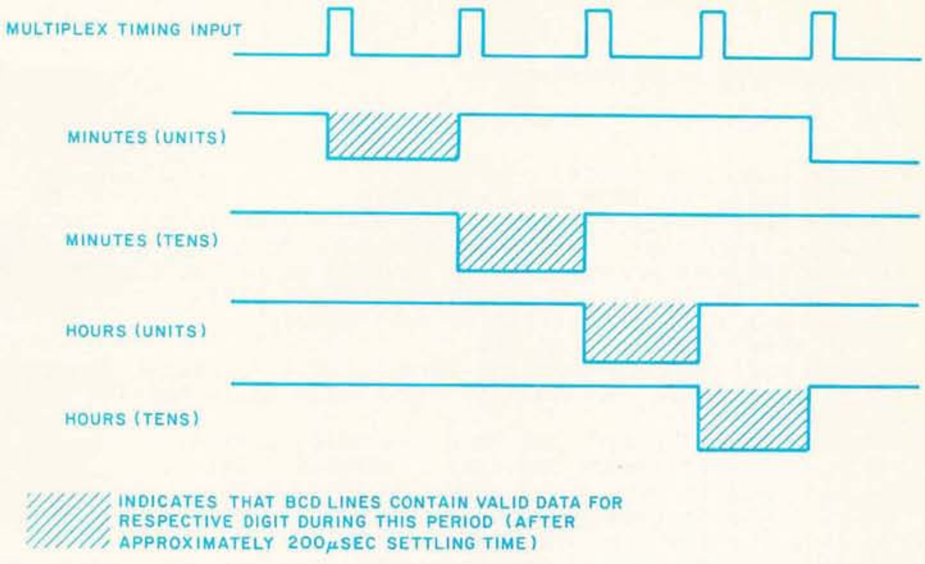



*Figure 6: Display multiplex timing sequence for the circuit in figure 5.*

This circuit, which can be manually or automatically preset, is fully static and allows the display output lines to be completely under program control. The basic 5-chip interface consists of a 4-digit BCD/7-segment output clock type MM5312, an MM5369 time-base generator, 2 MOS to TTL buffers to send data to the microprocessor, and 1 TTL-to-CMOS converter for processor control over the clock chip. Time is read by the computer as 4 binary coded

*Listing 1: Program for the real-time clock.*

```
LIST

100 REM REAL TIME CLOCK
110 REM COPYRIGHT 1979 STEVEN CIARCIA
120 REM THIS SIMPLE PROGRAM ALLOWS A COMPUTER TO TELL TIME BY
130 REM INTERFACING A DIGITAL CLOCK CHIP TO AN I/O PORT. (PORT 8 IN THIS EXAMPLE)
140 REM THE DISPLAY MUX LINE IS CONTROLLED BY THE COMPUTER. FIRST IT IS PULSED UNTIL
150 REM IT IS SET ON THE LEAST SIGNIFICANT DIGIT
160 OUT 8,1 :OUT 8,0 : T=INP(8) :D=T AND 16
170 IF D=16 THEN 200 ELSE 160
180 REM ONCE THE LSD POSITION IS SET THE 4 SUCESSIVE READINGS ARE TAKEN
190 REM THE INPUT PORT DATA IS ANDED WITH 15 TO OBTAIN THE BCD VALUE (REMEMBER,BASIC USES
    DECIMAL)
200 M1=T AND 15 :GOSUB 250 :REM   MINUTES (UNITS)
210 M2=T AND 15 :GOSUB 250 :REM   MINUTES (TENS)
220 H1=T AND 15 :GOSUB 250 :REM   HOURS (UNITS)
230 H2=T AND 15 :GOSUB 250 :REM   HOURS (TENS)
240 PRINT H2;H1;":";M2;M1 :GOTO 160
250 OUT 8,1 :OUT 8,0 :T=INP(8):RETURN :REM ADVANCE DISPLAY MUX

READY
```

decimal numbers. In a 4-digit clock like the one in figure 5, the data appears as a digit-enable output and an associated BCD value. The tens of minutes data is available when bit $b_5$ is high (bits $b_4$, $b_6$, and $b_7$ are low). It will appear as a BCD quantity in bits $b_0$ thru $b_3$. Unlike the circuit of figure 4, this unit is static and has no display to drive. It will stay on a particular digit until it is instructed to sequence to the next digit. This is accomplished by controlling the display-multiplexer input line of the clock.

Figure 6 shows how the multiplexer line is controlled in this application. Bit 0 of an output port (port 8 in my example) is used to pulse multiplexer input pin 22. At any time, 1 of the 4 digit-enable output lines will be low (at the chip), indicating that the multiplexer is set on that digit. The data on the BCD lines is for that digit. Reading the next digit is simply a case of pulsing bit $b_0$ again. There is no time constraint either. You can wait 10 minutes between digits if you wish (but the data won't mean much). It is best to read the 4 digits sequentially. The circuit is easily interfaced and exercised in BASIC as demonstrated in listing 1. The flow diagram of this program is shown in figure 7.

The addition of 2 more gates connected to output bits $b_1$ and $b_2$ facilitate automatic time preset. Figure 8 follows the logic of how such a program could be written. Two magnetic reed switches shown in photo 1 can be attached between pins 14 and 15, respectively, and ground to allow manual preset as well. I find that it is easier to just turn on the

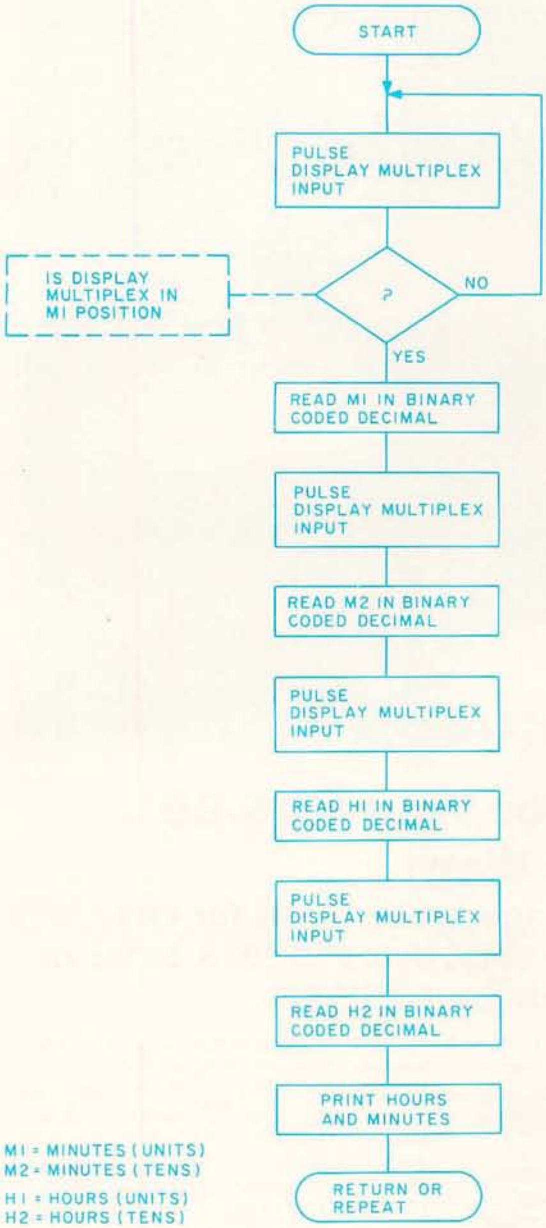



*Figure 7: Flowchart of the program given in listing 1.*

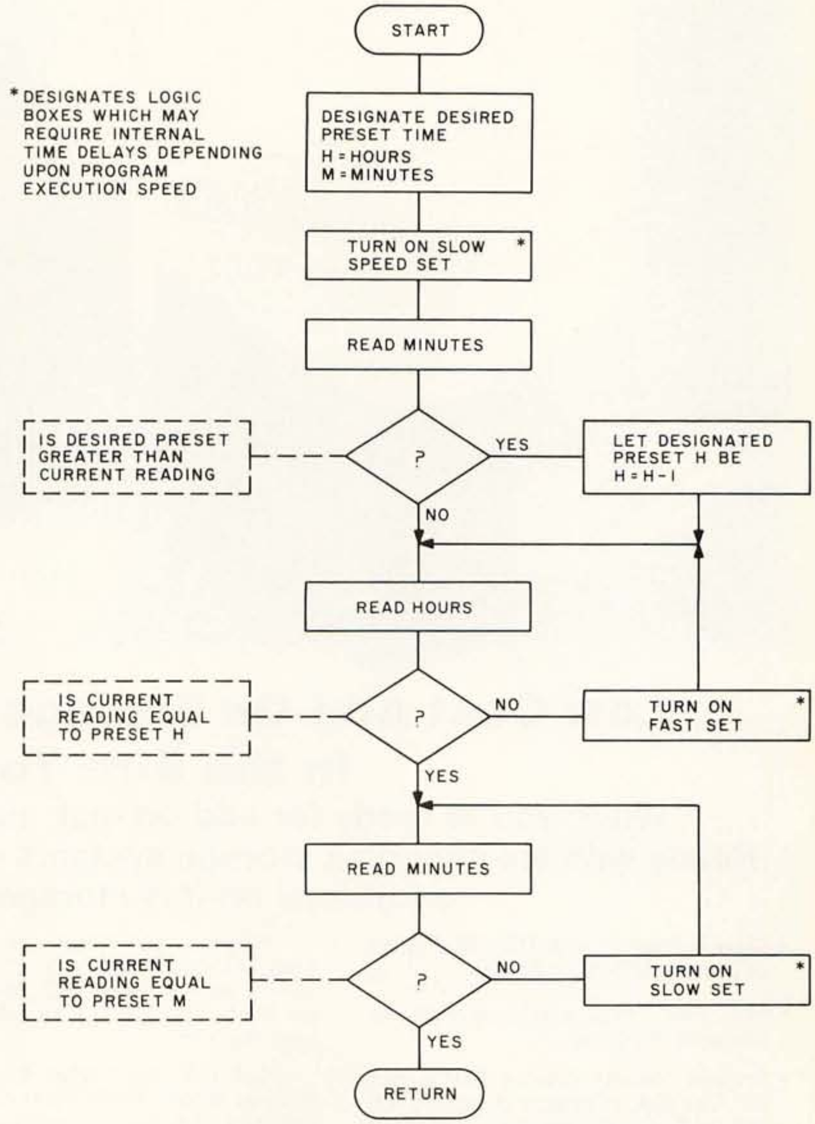



*Figure 8: Flowchart for the automatic reset routine.*

clock program in continuous display mode and adjust the clock as I read it. If a battery back-up capability is added, the 2 TTL automatic set gates should be disconnected. When the computer is powered up, random data can appear on bits $b_1$ and $b_2$, accidently causing it to enter the set mode. This is not a problem on the input. While a 4-digit, 24-hour clock is quite enough in my application (an example is shown in photo 2), there are those who need a second designation. Substituting an MM5311, the $s_1$ and $s_{10}$ digit-enable line can be added as 2 more parallel input bits and treated exactly as the present circuit, or binary encoded to reduce input bits, as shown in figure 5b. This method will require a slight software change but should be an equally viable approach. The present program in listing 1 executes in approximately 50 ms when used with Micro Com 8 K Zapple BASIC, but it works equally well with a machine language routine.

Whatever your final configuration, I am sure you will find that accurately timed control outputs are a definite advantage on any system. And there is no reason for the hardware of any interface to constrain the operator's choice of software interaction if it is not dictated by the frequency of events themselves. ■

---

*Next month the topic of "Ciarcia's Circuit Cellar" will be various joystick interfaces.*

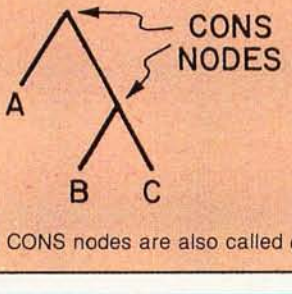
---

Circle 305 on inquiry card.

# LISP Notes

John Allen
Signetics
811 E Acques Ave
Mail Stop 38
Sunnyvale CA 94086

- **Symbolic Expressions** (or S-expressions) are the primitive data items of LISP. The usual interpretation of these expressions is a binary tree where LISP atoms appear at the tips of the tree, and the internal nodes, called *CONS nodes*, have 2 branches. The left branch is called the *CAR branch*; the right branch is called the *CDR branch*. For example:




These CONS nodes are also called *dotted pairs* because the linear notation for these trees, called *dot notation*, represents the nodes as dots. For example, the tree above would be written as (A . (B . C)) in dot notation. The LISP functions *car* and *cdr* select the CAR and CDR branches respectively. The function *cons* constructs a new binary tree from 2 fragments.

- **M-Expressions** of an external notation for LISP, while a special kind of S-expression, called *list notation*, are used for both the programming notation and the data notation. All articles in this month's BYTE use list notation for their data items. To emphasize the distinction between the idea of a list and its implementation as a dotted pair, the functions *first*, *rest*, and *concat* will sometimes be used instead of car, cdr, and cons, even though the functions are identical in traditional implementation.

Within the LISP language are several powerful and distinctive features. One, called *lambda notation*, gives LISP the ability to describe and manipulate functions as data objects. We use a simplified form of this concept in the LISP operators *DEF* and *DEFINE*. Another LISP distinction involves its concept of a *scope rule*: basically a rule to apply when finding the value of a nonlocal variable from within a function call. The default rule in LISP (and in APL) is called the *dynamic scope*, meaning "use the latest binding of a variable" (ie: the binding which was available when the function was called). ALGOL and Pascal use a rule called *static scope* which says, "use the value which was current at the time the function was defined."■

MICRO WORLD

TOLL-FREE 1-800-528-1418

LOWEST PRICES • FAST DELIVERY • DEPENDABLE SERVICE

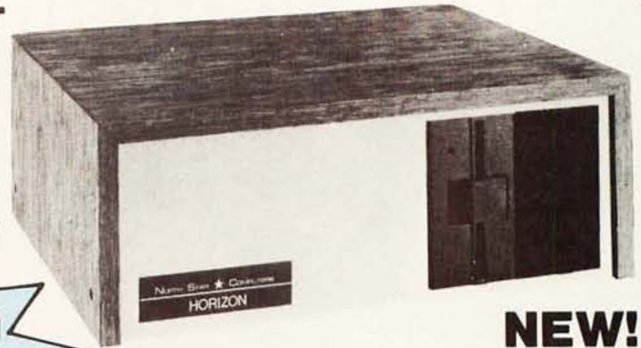
## DOUBLE DENSITY HORIZON I KIT

**double the storage at the same price!**

**Today's best buy. 180K bytes per disk. The chosen computer for two MicroWorld systems . . . Autoscribe — The Paperwork Manager™ and Bookkeeper — The Office Accountant™. Single density still runs on your new Horizon, or you can copy and convert all North Star software and programs to double density.**

- *exclusive* application software
- *add'l 16K memory* (kit), $349
- *add'l disk drive* (kit), $349

$1349 (reg. $1599)



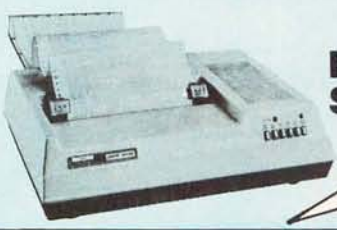

**NEW!**

**Call for low assembled prices. Double density also available on North Star disk sub-system, $599 kit.**

RUN SUPERIOR MICROPOLIS BASIC

## MICROPOLIS DISK DRIVE FOR TRS-80

$499 Introductory Price (reg. $599)



## HEATH DATA SYSTEMS PRINTER

$895

## SOROC IQ 120 VIDEO TERMINAL

$795 (reg. $995)

Immediate Shipment!

## TI 810 PRINTER

$1695 (reg. $1895)

## MicroWorld Specials

| | |
|---|---|
| Integral Data Printer .... | $ 749 |
| Add for Tractors ........ | $ 150 |
| Mime Terminal ......... | $ 742 |
| Hazeltine 1410 Terminal . | $ 765 |
| Hazeltine 1500 Terminal . | $1097 |
| Teletype Model 43 Printer ............... | $1019 |
| Dataproducts M-200 Printer ............... | $2921 |
| Vector Graphics MZ .... | $3225 |
| Exidy 16K Sorcerer ..... | $1035 |
| Data General microNOVA ......... | $CALL |
| Qume Sprint 5 RO Printer (55cps) ........ | $2895 |
| DEC LS 120 Printer .... | $CALL |
| Centronics 779 Printer (incl. tractors) ........ | $1095 |
| Micropolis 1042 Mod I Drive .............. | $CALL |
| TI 59 Calculator ........ | $ 219 |
| Novation CAT Modem .. | $ 199 |
| TI 820 Terminal ......... | $2199 |
| North Star Horizon Software ............ | $CALL |
| Dataproducts B-300 Printer ............... | $5346 |
| IPSI 1620 Diablo RO .... | $2705 |
| Hitachi 9" Monitor ...... | $ 184 |
| Cromemco System III ... | $5299 |
| Imsai VDP-80 .......... | $CALL |

MICRO WORLD™

Bringing information technology to your doorstep

1425 W. 12th Pl. • Tempe, AZ 85281 • 602-894-1193

VISA No extra charge for credit card orders master charge

Circle 233 on inquiry card.

**Freight collect, F.O.B. Tempe**
**No other handling charges**

Formerly Byte Shop Mail Order

# CALL TOLL-FREE 1-800-528-1418

# A Model of the Brain for Robot Control

## Part 3: A Comparison of the Brain and our Model

James Albus
Project Manager
United States Dept of Commerce
National Bureau of Standards
Washington DC 20234

In parts 1 and 2 we have shown how a neurological model called the Cerebellar Model Arithmetic Computer (CMAC) can compute functions, recognize patterns, and decompose goals. We have also shown how a crosscoupled hierarchy of CMACs (see figure 1) can memorize trajectories, generate goal directed purposive behavior, and store an internal model of the external world in the form of predicted sensory data. In this third article we will attempt to show how this structure and its capabilities can give rise to perceptual and cognitive phenomena.

The fact that the mathematical details of the CMAC model were derived from the cerebellum, a portion of the brain particularly regular in structure and hence uniquely suitable for detailed neurophysiological analysis, does not mean that the results are inapplicable to other regions of the brain as well. The basic structure of a large output cell (sometimes called a principal, relay, or projection neuron) served by a cluster of local interneurons is quite typical throughout the brain. Such clusters commonly receive input from a large number of nonspecific neural fibers similar to the mossy fibers in the cerebellum. In many instances they also receive specific inputs which are more or less analogous to climbing fibers. As we might expect, there are many differences in size and shape of the corresponding cell types from one region of the brain to another. These reflect differences in types of computations being performed and information being processed, as well as differences in the evolutionary history of various regions in the brain. Nevertheless, there are clear regularities in organization and similarities in function from one region to another. This suggests that, at least to a first approximation, the basic processes are similar.

The implication is that the general model of information processing defined by CMAC (the concept of a set of principal neurons together with their associated interneurons transforming an input vector **S** into an output vector **P** in accordance with a mathematically definable relationship H) may be useful in analyzing the properties of many different cortical regions and subcortical nuclei. This is particularly true since the accuracy, resolution, rate of learning, and degree of generalization of the CMAC H function can be chosen to mimic the neuronal characteristics of different areas in the brain.

**About the Author:**

*Dr James S Albus worked for NASA from 1957 to 1972 designing optical and electronic subsystems for over 15 spacecraft, and for one year managed the NASA Artificial Intelligence Program. Since 1973 he has been with the National Bureau of Standards where he has received several awards for his work in advanced computer control systems for industrial robots. He has written a survey article on robot systems for the February 1967 issue of* Scientific American *and his Cerebellar Model Arithmetic Computer won the Industrial Research Magazine IR-100 award as one of the 100 most significant new products of 1975.*

### Hierarchical Control

The idea that the central nervous system, which generates behavior in biological organisms, is hierarchically structured is an old one, dating back considerably more than a century. The analogy is often made to a military command structure, wherein many hundreds of operational units and thousands, even millions of individual soldiers are coordinated in the execution of complex tasks or goals. In this analogy each computing center in the behavior-generating hierarchy is like a military command post, receiving commands from immediate superiors and issuing sequences of subcommands which carry out those commands to subordinates.

Feedback is provided to each level by a sensory-processing hierarchy which ascends parallel to the behavior-generating hierarchy, and which operates on a data stream derived from sensory units which monitor the external environment as well as from lower level command centers which report on the progress being made in carrying out their subcommands. Feedback is processed at many levels in this ascending hierarchy by intelligence analysis centers that extract data relevant to the command and control functions being performed by the behavior-generating module at that level.

Each of these intelligence analysis centers makes predictions based on the results expected (ie: casualties, rewards, sensory data patterns) as a consequence of actions currently being taken. The intelligence centers then interpret the sensory data they receive in the context of these predictions. For example, in military in-

The ideas presented in this article represent the views of the author and not those of the Department of Commerce or the National Bureau of Standards.

telligence analysis a loss of 60 men in an operation where losses had been predicted at 600 implies an unexpectedly easy success, and perhaps indicates a weakness in the enemy position which should be further exploited. In the brain, the observation of 60 nerve impulses on an axon where 600 has been anticipated may imply an unexpectedly weak branch in a tree, upon which the placing of any weight will result in a fatal fall from the treetop.

The response of each command post (or data analysis center) in the hierarchy to its input depends on how it has been trained. Basic training teaches each soldier how to do things the "army way" (ie: what each command means and how it should be carried out). Each operational unit in the military has a field manual which defines the proper, or ideal response of that unit to every foreseeable battlefield situation. Each field manual is essentially a set of IF/THEN production rules or case statements, corresponding to a set of CMAC functions, $\mathbf{P} = H(\mathbf{S})$ or $\mathbf{Q} = G(\mathbf{D})$. At the lowest level in the military analogy these rules define the proper procedures for maintaining and operating weapons, as well as the proper behavioral patterns for surviving and carrying out assignments under battlefield conditions. At higher levels they define the proper tactics for executing various kinds of maneuvers. At the highest level, they define the proper strategy for deployment of resources and achievement of objectives.

In the case where each unit carries out its assignment "according to the book," the overall operation runs smoothly and the goal is achieved on schedule as expected. To the extent that various units do not follow their ideal trajectories, either because of improper training or because of unforeseen difficulties in the environment, the operation will deviate from the expected or planned schedule. Alternate tactics may be required. If a change in tactics still does not produce success, new strategies may be required. Of course, there is always the possibility that failure will occur, despite every effort. The goal will not be achieved or, worse yet, the organism may suffer a catastrophic setback.

There is considerable anatomical, neurophysiological, and behavioral evidence that the analogy between the brain and a military hierarchy is quite accurate. However, in saying this, it is important to keep in mind that the highly schematic hierarchy shown in figure 1 is a grossly oversimplified diagram of the vast interconnected hierarchical network which is the brain. Every motor neuron in the nervous system can be thought of as being controlled by its own hierarchy which interleaves and overlaps extensively with the hierarchies of nearby synergistic motor neurons. Each sensory-motor system has its own set of overlapping hierarchies which become increasingly interrelated and interconnected with each other at the higher levels. Thus, the entire brain may have the topological shape of an inverted paraboloid as shown in figure 2.

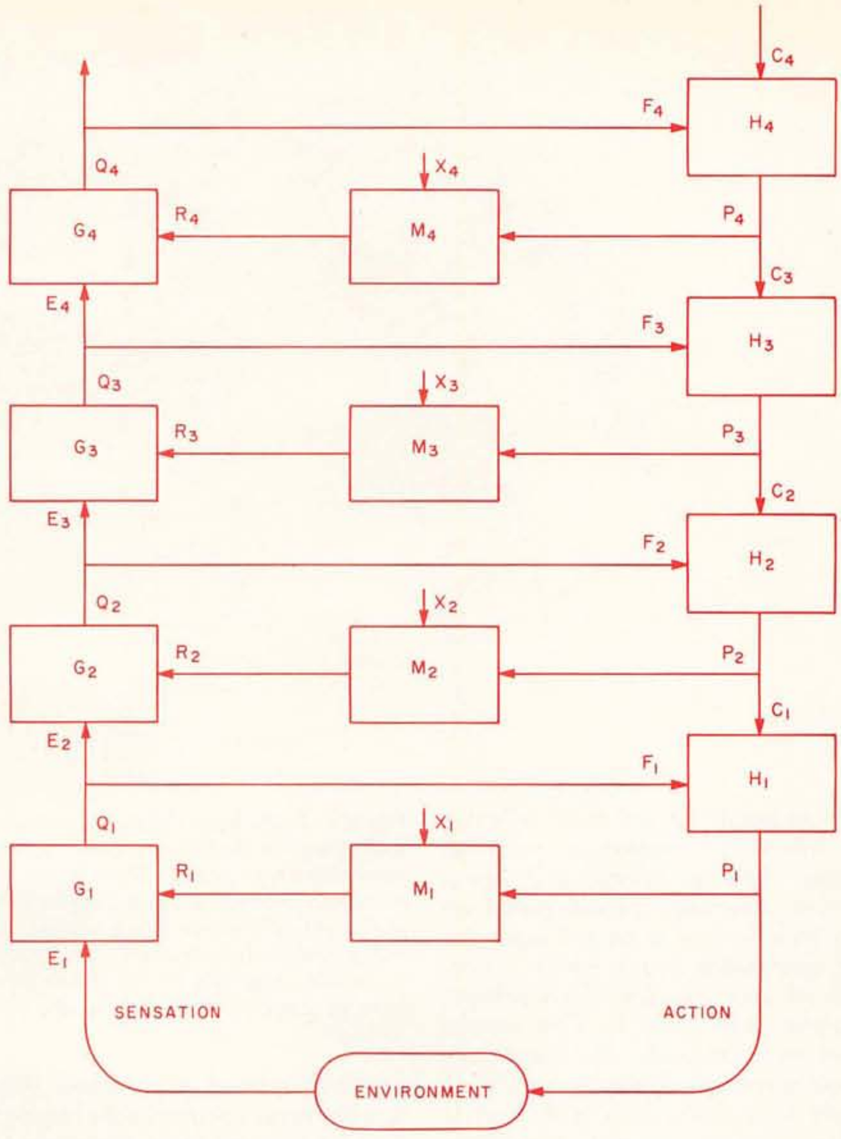



*Figure 1: A crosscoupled, processing-generating hierarchy. The H modules decompose input goals* **C** *into output subgoals* **P** *using feedback* **F**. *The M modules recall expected sensory data* **R** *which is compared with observed sensory experiences* **E**. *The G modules recognize sensory patterns* **Q** *and compute feedback errors* **F**.

## Triune Brain Hypothesis

There is in fact some evidence to suggest that the human brain is topologically similar to three (or more) concentric paraboloid hierarchies as illustrated in figure 3. Paul MacLean and others have hypothesized a triune brain wherein the inner core is a primitive structure (ie: the reptilian brain) which provides vital functions

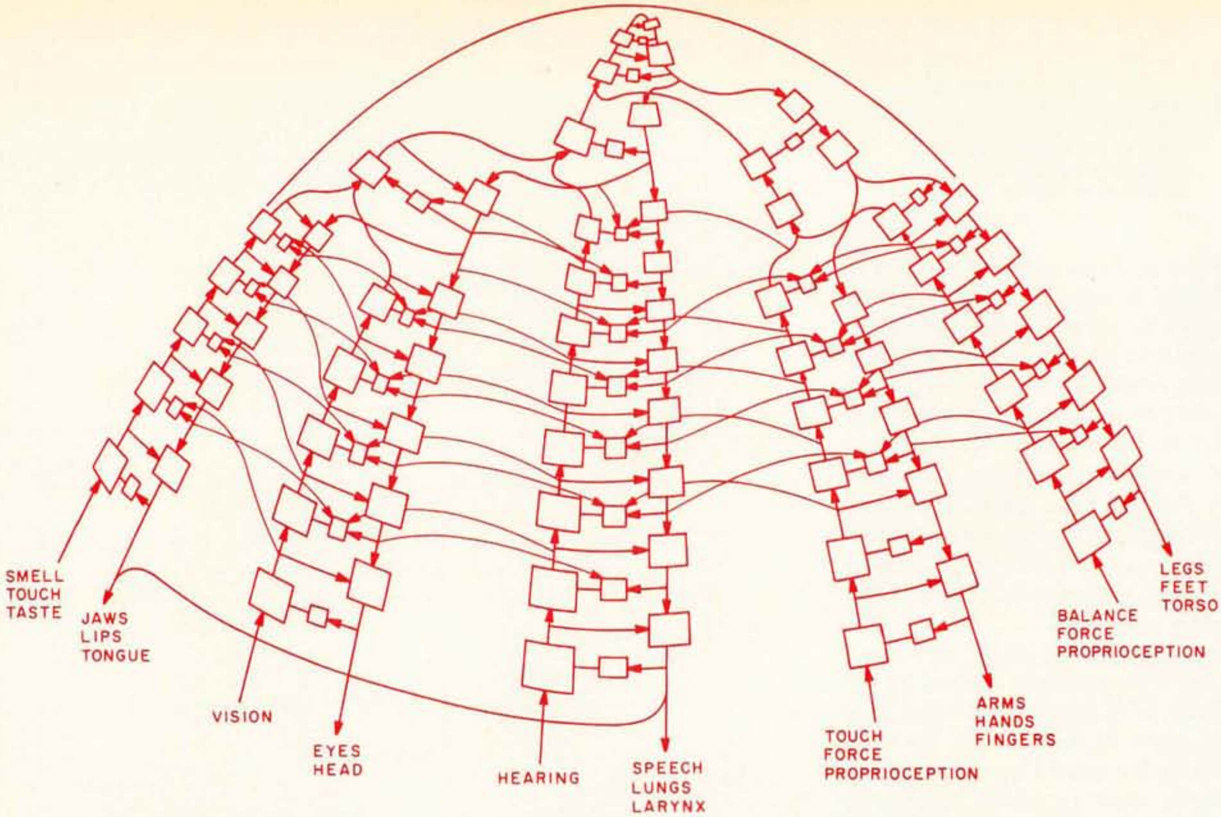



*Figure 2. In the brain different processing-generating hierarchies represent different sensory-motor systems. These become increasingly interrelated at the higher levels and eventually merge into a unified command and control structure. This enables a complex organism to coordinate its actions in pursuit of high level goals.*

such as breathing and basic reflexive or instinctive responses such as eating, fighting, fleeing, and reproductive activities. Superimposed on this inner core is a second layer (ie: the mammalian brain) which is capable of more sophisticated sensory analysis and control. This second layer tends to inhibit the simple and direct responses of the first so as to apply them more selectively and to delay responses until opportune moments. This second brain thus provides the patient waiting behavior necessary for effective hunting of prey. On top of this is yet a third layer (ie: the primate brain) which possesses the capacity to manipulate the other two layers in extremely subtle ways; to imagine and plan, to scheme and connive, to generate and recognize signs and symbols, to speak and understand what is spoken.

The outer layers employ much more sophisticated sensory analysis and control algorithms that detect greater subtleties and make more complex decisions than the inner more primitive layers are capable of performing. Under normal conditions the outer layers modify, modulate, and sometimes even reverse the sense of the more primitive responses of the inner layers. However, during periods of stress, the highly sophisticated outer layers may encounter computational overload and become confused or panicked. When this happens, the inner core hierarchy may be released from inhibition and execute one of the primitive survival procedures stored in it (ie: fight, flee, or freeze). A similar takeover by the inner hierarchy may occur if the more delicate circuitry of the outer is disrupted by physical injury or other trauma. Thus the brain uses its redundancy to increase reliability in a hostile environment.

Of course, all three layers of the behavior-generating hierarchy come together at the bottom level in the motor neuron — the final common pathway.

## Motor-Generating Hierarchies in the Brain

In the military hierarchy analogy, the motor neurons are the foot soldiers. They produce the action. Their firing rates define the output trajectory of the behavior-generating hierarchy. A CMAC representing a spinal motor neuron and its associated interneurons receive feedback **F** from stretch receptors via the dorsal roots, as well as from other motor neurons reporting ongoing activity in related muscles. The command vector **C** to this lowest level comes from the vestibular system, which provides inertial reference signals necessary for posture and balance, as well as from the reticular formation and basal ganglia (and in primates, also directly from the motor cortex).

There is nothing analogous to climbing fibers for the motor neurons, but this is not surprising since there is evidence that little or no learning takes place at this first level in the behavior-generating hierarchy.

Evidence for second, third, and fourth levels in the behavior-generating hierarchy comes from experiments with animals and observations of injured humans where the spinal cord is severed at different levels. If, as is shown in figure 4, the cord is severed from the brain along the line A-A, most of the basic motor patterns such as the flexor reflex and the reflexes that control the basic rhythm and patterns of locomotion remain intact. However, coordinated activation of these patterns to stand up and support the body against gravity requires that the regions below B-B be intact.

The stringing together of different postures to permit walking and turning movements requires the regions

**Color.** VP-590 add-on Color Board allows program control of 8 brilliant colors for graphics, color games. Plus 4 selectable background colors. Includes sockets for 2 auxiliary keypads (VP-580). $69*

**Sound.** VP-595 Simple Sound Board provides 256 tone frequencies. Great for supplementing graphics with sound effects or music. Set tone and duration with easy instructions. $24*

**Music.** VP-550 Super Sound Board turns your VIP into a music synthesizer. 2 sound channels. Program control of frequency, time and amplitude envelope (voice) independently in each channel. Program directly from sheet music! Sync provision for controlling multiple VIPs, multitrack recording or other synthesizers. $49*

**Memory.** VP-570 RAM Expansion Board adds 4K bytes of memory. Jumper locates RAM in any 4K block of up to 32K of memory. On-board memory protect switch. $95*

**EPROM Programmer.** VP-565 EPROM Programmer Board comes complete with software to program, copy and verify 5-volt 2716 EPROMs—comparable to units costing much more than the VP-565 and VIP put together! Programming voltages generated on board. ZIF PROM socket included. $99*

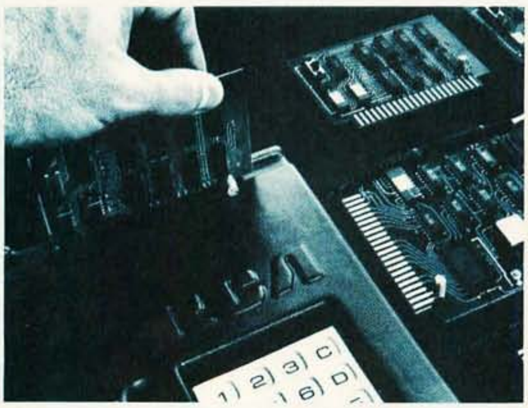
**EPROM Interface.** VP-560 EPROM Interface Board locates two 5-volt 2716 EPROMs (4K bytes total) anywhere in 32K of memory. VIP RAM can be re-allocated. $34*

**ASCII Keyboard.**** Fully encoded, 128-character ASCII encoded alpha-numeric keyboard. 58 light touch keys including 2 user defined keys! Selectable upper and lower case. Handsomely styled. Under $50*

**Tiny BASIC.**** VP-700 Expanded Tiny BASIC Board puts this high-level language on your VIP. BASIC stored in 4K of ROM. Ready for immediate use—no loading necessary. This expanded BASIC includes the standard Tiny BASIC commands plus 12 additional—including color and sound control! Requires external ASCII encoded alpha-numeric keyboard. $39*

**Auxiliary Keypads.** Program your VIP for 2-player interaction games! 16-key keypad VP-580 with cable ($15*) connects to sockets provided on VP-590 Color Board or VP 585 Keyboard Interface Card ($10*).

# COSMAC VIP lets you add computer power a board at a time.



With these new easy-to-buy options, the versatile RCA COSMAC VIP (CDP18S711) means even more excitement. More challenges in graphics, games and control functions. For everyone, from youngster to serious hobbyist. And the basic VIP computer system starts at just $249* assembled and ready to operate.

**Simple but powerful—not just a toy.**

Built around an RCA COSMAC microprocessor, the VIP includes 2K of RAM. ROM monitor. Audio tone with a built-in speaker. Plus 8-bit input and 8-bit output port to interface relays, sensors or other peripherals. It's easy to program and operate. Powerful CHIP-8 interpretive language gets you into programming the first evening. Complete documentation provided.

**Take the first step now.**

Check your local computer store or electronics parts house. Or contact RCA VIP Marketing, New Holland Avenue, Lancaster, PA 17604. Phone (717) 291-5848.

*Suggested retail price. CDP18S711 does not include video monitor or cassette recorder.
**Available 1st Quarter, 1979.

**See the RCA VIP at the 3rd Annual National Small Computer Show at the New York Coliseum, August 23-26, booth 4211.**

**The fun way into computers.**

RCA

Circle 322 on inquiry card.

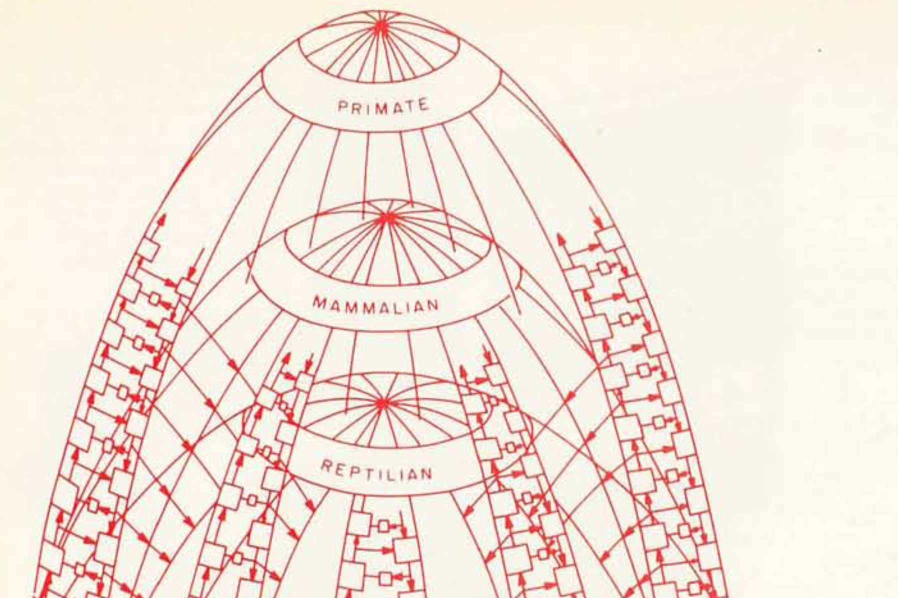



*Figure 3: The human brain is hypothesized to be a composite structure consisting of at least three layers: (1) a reptilian brain which provides basic reflexes and instinctive responses; (2) a mammalian brain which is more sophisticated and capable of delayed responses; and (3) a primate brain which can imagine, plan and manipulate abstract symbols. The outer layers inhibit and modulate the more primative tendencies of the inner layers.*

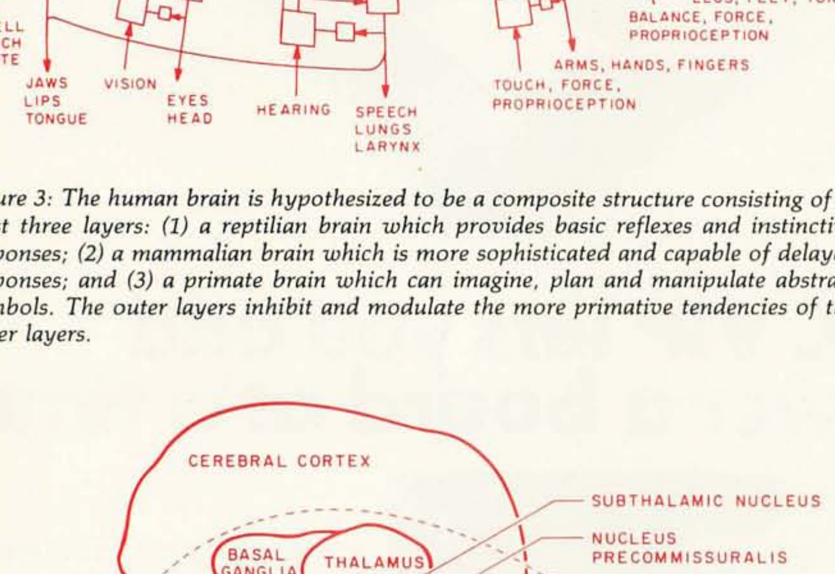



*Figure 4. The hierarchy of motor control that exists in the extrapyramidal motor system. Basic reflexes remain even if the brain stem is cut at A-A. Coordination of these reflexes for standing is possible if the cut is at B-B. The sequential coordination required for walking requires the area below C-C to be operable. Simple tasks can be executed if the region below D-D is intact. Lengthy tasks and complex goals require the cerebral cortex.*

below C-C to be undamaged. In particular it is known that the rotational movements of the head and eyes are generated in the interstitial nucleus; raising and lowering of the head in the prestitial nucleus; and flexing movements of the head and body in the nucleus precommissuralis. Stimulation of the subthalamic nuclei can cause rhythmic motions including walking. A cat with its brain sectioned along C-C can walk almost normally. However, it cannot vary its walking patterns to avoid obstacles.

Animals whose brains are cut along the line D-D can walk, avoid obstacles, eat, fight, and carry on normal sexual activities. However, they lack purposiveness. They cannot execute lengthy tasks or goals. Humans with brain disease in the basal ganglia may perform an apparently normal pattern of movements for a few seconds and then abruptly switch to a different pattern, and then another. One form of this disease is called St Vitus' dance.

Higher levels of the behavior-generating hierarchy become increasingly difficult to identify and localize, but there is much to indicate that many additional levels exist in the cerebral cortex. For example, the motor cortex appears to be responsible for initiating commands for complex tasks. The ability to organize lengthy sequences of tasks, such as the ability to arrange words into a coherent thought or to recall the memory of a lengthy past experience, seems to reside in the posterior temporal lobe. Interactions between emotions and intentional behavior appear to take place in the mediobasal cortex, and long term plans and goals are believed to derive from activity in the frontal cortex. Hierarchies of different systems (ie: vision, hearing, manipulation, locomotion, etc) merge together in the association areas.

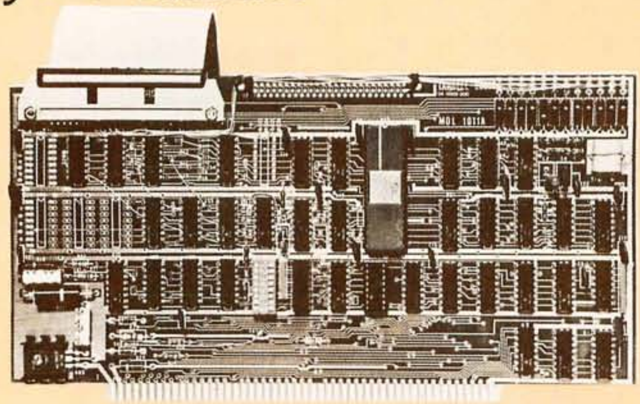
## Sensory-Processing Hierarchies in the Brain

It is a well established fact that hierarchies of sensory-processing modules exist in the brain. In a famous series of experiments, Hubel and Wiesel demonstrated four clearly distinguishable hierarchical levels in the visual system. Similar sensory-processing hierarchies have been extensively studied in the auditory

# The TARBELL Connection

In an effort to offer products that meet the continually changing demands of the microcomputer industry, TARBELL ELECTRONICS is pleased to offer immediate delivery of these quality components and operating software. All TARBELL products are available from computer store dealers everywhere.

## Tarbell Floppy Disk Interface



- Plugs directly into your IMSAI or ALTAIR and handles up to 4 standard single drives in daisy-chain.
- Operates at standard 250K per second on normal disk format capacity of 256K bytes.
- Works with modified CP/M Operating System and BASIC-E Compiler.
- Hardware includes 4 extra IC slots, built-in phantom bootstrap and on-board crystal clock. Uses WD 1771 LSI chip.
- Full 6-month warranty and extensive documentation.
- Kit $190 . . . . . . . Assembled $265.

Specify drive for assembled units. Complete disk subsystems with operating software available. Please inquire for details.

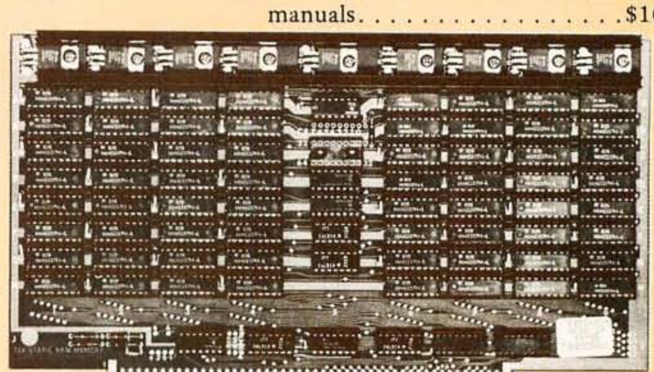
## Tarbell Disk BASIC

- Runs on 8080, 8085 or Z80
- Searches a file quickly for a string.
- Up to 64 files open at once.
- Random Access.
- Assignment of I/O.
- Alphanumeric line labels allowed.
- Read and Write string or numeric data.
- Unlimited length of variable names and strings.
- Procedures with independent variables.
- Number system 10 digits BCD integer or floating point.
- Chain to another program.
- Cause programs to be appended onto programs already in memory.
- Cause interpreter to enter edit mode using 15 single character edit commands.

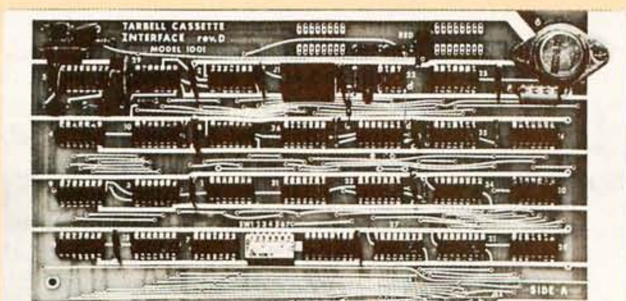
Occupies 24K of RAM. Tarbell BASIC on CP/M Disk . . . . . . . . . . . . . .$48.
Source on paper or CP/M Disk . . .$25.
CP/M and BASIC-E on disk with manuals. . . . . . . . . . . . . . . . .$100.

## Tarbell 32K RAM Memory

- 32K Static Memory
- S-100 Bus Connector
- 9 regulators provide excellent heat distribution.
- Extended addressing (bank switching.)
- Phantom line.
- Low power requirement.
- 20-Page operating manual.
- Full 1-year warranty.
- Assembled and tested full price only $625
- 16K version also available, assembled and tested only $390.



## Tarbell Cassette Interface



- Plugs directly into your IMSAI or ALTAIR.
- Fastest transfer rate: 187 (standard) to 540 bytes/second.
- Extremely reliable—Phase encoded (self-clocking).
- 4 extra status lines, and 4 extra control lines.
- 37-page manual included.
- Device code selectable by DIP-switch.
- Capable of generating Kansas City tapes.
- No modification required on audio cassette recorder.
- Complete kit $120 . . . . . . . . . . . . . . . Assembled $175.
- Manual may be purchased separately . . . . . . . . . . . . . $8.
- Full 6-month warranty on kit and assembled units.

## Tarbell Cassette BASIC

Includes most features of ALTAIR Extended BASIC, plus these added features:

- Assignment of I/O.
- Alphanumeric line labels.
- Unlimited length of variable names and strings.
- Number system 10 digits BCD integer or floating point.
- Procedures with independent variables.
- Read and Write string data.
- Multi-file capability.

Full price with complete documentation . . . . . . . . . . . .$48.

Prepaid, COD, or cash only. California residents please add 6% sales tax.

*ALTAIR is a trademark/tradename of Pertec Computer Corporation*
*CP/M is a trademark/tradename of Digital Research*

Tarbell Electronics
950 DOVLEN PLACE • SUITE B
CARSON, CALIFORNIA 90746
(213)538-4251 • (213) 538-2254

Circle 360 on inquiry card.

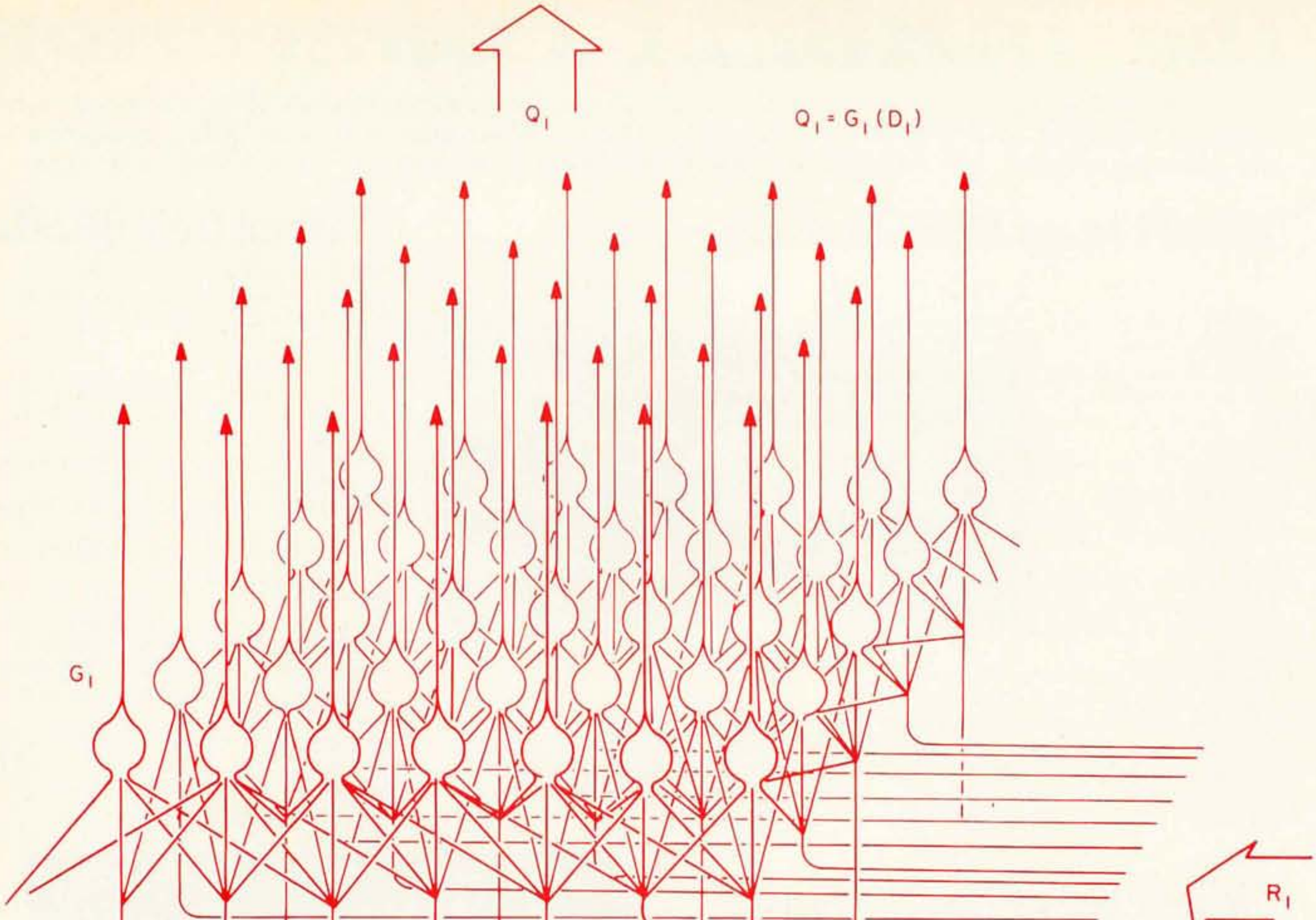



*Figure 5: A two-dimensional array of sensory-processing Cerebellar Model Arithmetic Computers such as might exist in the visual system. The observed sensory image* $\mathbf{E_1}$ *plus the prediction vector* $\mathbf{R_1}$ *enters and is recognized by the operator* $G_1$ *as a pattern. The vector* $\mathbf{R_1}$ *may select one of many filter functions or provide an expected image or map to be compared against the observed image.*

system and also the proprioceptive and kinesthetic pathways. Cross-coupling from these ascending hierarchies of sensory-processing modules to the motor-generating hierarchies provides the many different levels of sensory feedback information required at the various stages of the task or goal decomposition process. At each level, output vectors from the previous level of the sensory-processing hierarchy provide inputs to the next higher level, as well as feedback to the same level of the behavior-generating hierarchy.

In the case of vision, the two-dimensional nature of input from the surface of the retina causes the computational modules in the visual processing system to be organized in sheets. This implies that a CMAC model of a typical level in the visual processing hierarchy would resemble the structure shown in figure 5. In this structure the sensory input $\mathbf{D_1}$ might consist of a pattern of sensory variables $\mathbf{E_1}$ defining light intensity (perhaps in a particular color band) together with predicted variables $\mathbf{R_1}$ which select a particular filter function. The output $\mathbf{Q_1} = G_1 (\mathbf{D_1})$ then might define a pattern of edges or line segments. This output forms part of the input $\mathbf{E_2}$ to the second level. Output from the second level, $\mathbf{Q_2} = G_2 (\mathbf{D_2})$, might define patterns of connected regions or segments.

Recent work by David Marr at the Massachusetts Insititute of Technology and Jay Tennenbaum at SRI International suggests that the output vectors $\mathbf{Q_i}$ at various levels may define more than one type of feature. For example, a single level in the visual processing system might contain a depth image (derived from stereo disparity, light gradients, local edge-interaction cues, etc), a velocity image (derived from motion detectors), and an outline drawing image (derived from edge detectors, line, and corner finders) in addition to brightness, color, and texture images of the visual field. These and many other kinds of information appear to

exist in registration at several different levels of the visual information processing hierarchy so as to make possible the extremely sophisticated visual recognition tasks which our brains routinely perform. These different types of images interact, sometimes reinforcing each other so as to confirm a recognition, and sometimes contradicting each other so as to reject one possible interpretation of the visual input in favor of another.

### Crosscoupling

Cross links from the descending hierarchies of motor-generating modules provide the many different levels of contextual and predictive information required at various stages of the pattern recognition or sensory analysis process. In the visual hierarchy, as well as in all other sensory-processing hierarchies, context variables $\mathbf{R}_i$ may define expected values of the $\mathbf{E}_i$ vectors. This implies that the addresses $\mathbf{P}_i$ and $\mathbf{X}_i$ have stored data from previous experiences when what is currently recalled as $\mathbf{R}_i$ was experienced as $\mathbf{E}_i$. In this case the recalled context $\mathbf{R}_i$ is essentially a stored image, or map, which is accessed by an associative address created by the behavior-generating hierarchy being in a state more or less similar to that which existed when the remembered experience (ie: the map) was stored.

This implies that the sensory data processing hierarchy is a multilevel map (or template) matching process, and that in order to generate these maps the behavior-generating side of the crosscoupled hierarchy must be put into a state (or pulled along a trajectory) similar to that which existed when the template was recorded.

When this occurs, the interaction around the loop formed by the $G_i$, $H_i$, and $M_i$ modules at each level is similar to a phase-lock loop, or a relaxation process. The data $\mathbf{E}_i$ enters the module $G_i$ which recognizes it to be in a certain class $\mathbf{Q}_i$ with perhaps an error of $\mathbf{F}_i$. The recognition $\mathbf{Q}_i$ triggers an appropriate goal decomposition (or subgoal selection) function in the $H_{i+1}$ (or higher) modules which generates a command (or hypothesis) $\mathbf{C}_i$. This command, modified by the error $\mathbf{F}_i$, generates a subcommand (or subhypothesis) $\mathbf{P}_i$ and hence a predicted data vector $\mathbf{R}_i$. The prediction $\mathbf{R}_i$ may confirm the preliminary recognition $\mathbf{Q}_i$ and pull the context $\mathbf{P}_i$ into a more exact prediction via the feedback loop involving $\mathbf{F}_i$. Alternatively the prediction $\mathbf{R}_i$ may cause $G_i$ to alter or abandon the recognition $\mathbf{Q}_i$ in favor of another recognition $\mathbf{Q}'_i$.

### Loops and Rhythms

Obviously such looping interactions involve timing and phase relationships which may themselves have information content. Many sensory data patterns, especially in the auditory, visual, and kinesthetic pathways, are time dependent and involve some form of rhythmic or harmonic temporal patterns as well as spatial relationships. For example, activities such as walking, running, dancing, singing, speaking, and gesturing all have a distinctly rhythmic and sometimes strictly periodic character.

As was discussed in part 1 of this series, temporal patterns at various levels correspond to trajectories with different time rates of change, and hence (assuming approximately the same information content stored as trajectories at each level) different periods or complete rhythmical patterns. For example, at the lowest level of the auditory system, brain cells are excited by mechanical and electrical stimuli with frequencies ranging from about 20 Hz to 20,000 Hz. These sensory inputs thus have periodicities from 0.00005 to 0.05 seconds.

The highest frequency a nerve axon can transmit is about 500 Hz, but the brain handles higher frequencies in a manner somewhat reminiscent of the cerebellum's encoding of precise position. It encodes pieces of information about the phase of a wavefront on a number of different fibers. This means that by knowing which fibers are firing in which combinations at which instants, one can compute not only what is the fundamental pitch of the temporal pattern but what are all of its overtones. Thus, the CMAC G function at the lowest level (or really the loop comprised of the lowest level G, H, and M modules) can compute the Fourier transform, or the autocorrelation function, and presumably even the Bessel function describing the modes of vibration of the cochlear membrane.

Assume for example, that the G, H, and M modules in figure 6 constitute a phase-lock loop such that the input PATTERN is a signal f(t) and the

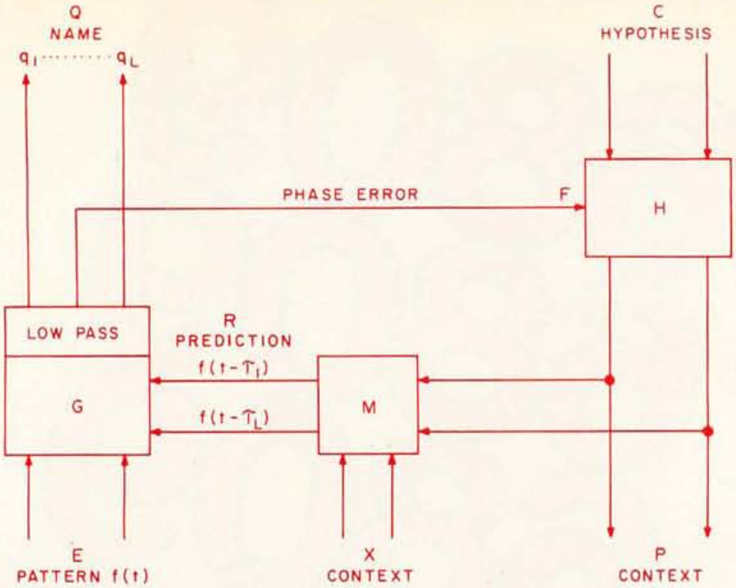



*Figure 6. A phase-lock loop consisting of a G, H, and M module. If the H and M modules produce a set of signals with nearly the same periodicity as the incoming signal* E, *the G function can compute a phase error signal* F *which pulls the* R *prediction into lock with the* E *observation. The G module can then also compute an autocorrelation function which gives a perception of pitch.*

PREDICTION is another signal $f(t-\tau)$. If the processing module G computes the product of the PATTERN • PREDICTION, then the output NAME is $f(t) \cdot f(t-\tau)$. When $\tau$ corresponds to 1/4 of the period of the input f(t), a low pass filter applied to the output will produce a phase ERROR signal which, when applied to the H module, can enable the PREDICTION signal $f(t-\tau)$ to track and lock on to the input PATTERN f(t). If the loop consists of a multiplicity of pathways with different delays ($\tau > 0$), the output, when processed through low pass filters, will produce an autocorrelation function:

$$\phi_{ff}(\tau) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} f(t) \cdot f(t-\tau)\, dt$$

such that:

$$Q = \begin{array}{l} q_1 = \phi_{ff}(\tau_1) \\ q_2 = \phi_{ff}(\tau_2) \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ q_L = \phi_{ff}(\tau_L) \end{array}$$

where:

$$0 < \tau_1 < \tau_2 \ \ldots \ < \tau_L$$

It has been shown that such an autocorrelation function produces a perception of pitch which is in good agreement with psychophysical data. In figure 6 the presence of an output on element $q_i$ would correspond to the perception of pitch at a frequency $\frac{1}{\tau_i}$

### Music and Language

Figure 7 suggests how a hierarchy of phase-lock loops might interact to recognize the variety of periodicities which provide the information content in spoken language and music. The coefficients that $q_i$ obtained from the lowest level loop form the input (together with other variables) to the second level.

If we assume that the sensory input to the first level consists of a pattern rich in information, such as music or speech, then as time progresses the trajectory of the input vector to the second level will also contain many periodicities. The principal difference from the standpoint of information theory is that the periodicity is now on the order of 0.05 seconds to 0.5 seconds. The trajectory input to the second level can, of course, be subjected to a quite similar mathematical analysis as were the trajectories of hair cell distortions and cochlear electrical stimulation which were input to the first level.

The principal difference is that at the second level and higher, information can be encoded for neural transmission by pulse-frequency rather than pulse-phase modulation. Also, some of the mechanisms by which time integrals are computed may be different. Nevertheless, processing by a CMAC G function can transform sections of the input trajectory into output vectors so as, in effect, to give them names. Characteristic patterns, or periodicities, at the second level are named notes, when the sensory stimulus is music. Where the stimulus is spoken language, they may be called phonemes.

The output of the second level forms part of the input to the third. The G function at the third level computes the names of strings of phonemes which it calls words, or strings of notes which it calls tunes. The G function at the fourth level computes names of strings of words which it calls sentences (or ideas), strings of tunes which it calls musical passages, etc. In music, the pattern in which the different periodicities match up as multiples and submultiples (ie: the beat, notes, various voices, melodies, and chord sequences) comprise the inner structure, harmony, or "meaning." The ability of the sensory processing-generating hierarchy of the listener to lock on to the periodicities and harmonies at many different levels (and hence many different periodic intervals) is the ability to "appreciate" or "understand" the music.

Similarly in speech the ability of the audio-processing hierarchy to lock on to periodicities at each level, and to detect or recognize and pass on to the next level the information bearing modulations or deviations in those periodicities, constitutes the ability to "understand" what is spoken. If the audio system locks on only at the first level, it detects phonetic sounds but not words. If it locks on the first two levels but no higher, it detects words but not meaningful phrases. If, however, the audio hierarchy locks on at the third, fourth, fifth, and higher levels, there is excited in the mind of the listener many of the same trajectories and sequences of interrelated and harmonious patterns (ie: goals, hypotheses, sensory experiences) as exist in the mind of the speaker.

This gives the speaker the ability to transmit messages and, even more important, to manipulate the mind of the listener to achieve his own goals. He can recruit help, enlist sympathy, give orders, and transmit all forms of sophisticated signals related to dominance, submission, and social interaction. Furthermore, by this mechanism he can induce into the highest levels of the sensory processing hierarchy of the listener recalled memories of his own experience. He can tell tales, relate stories, and thereby provide others with secondhand information as to what strategies and goal decomposition rules he personally has found to be successful.

## Origin of Language

One of the most basic features of language is that it is a form of behavior. That seems an obvious thing to say, but evidently it is not. Many experts feel that because language is connected with the intellect (ie: a higher function) it is quite divorced from mere motor behavior. However, there is no such thing as *mere* motor behavior. All behavior is the final output trajectory in the decomposition of high level goals. The intellect is *not* something distinct from behavior. It is the deep structure of behavior. It is the set of nonterminal trajectories which generate and coordinate what finally results in the phenomena of purposive or intentional action.

Language is certainly like other behavior in that it results from the coordinated contractions of muscles; in the chest, throat, and mouth. Like any other behavior such as walking, dancing, making a tool, or hunting for prey, language is both learned and goal directed.

The infant is born with only the most basic verbal reflexes. At first primitives are learned (coos, gurgles, cries, and phonetic sounds of various types), then strings of primitives (words), and finally strings of strings (phrases), etc. The sensory processing system stores (ie: records) sounds from the environment as $\mathbf{R}_i$ trajectories. Later the behavior-generating system learns to produce verbal outputs which mimic or duplicate these stored trajectories.

As with all behavior, the purpose of language is to obtain reward, to avoid punishment, and to achieve success in the social dominance hierarchy. The unique feature of language behavior is that it allows

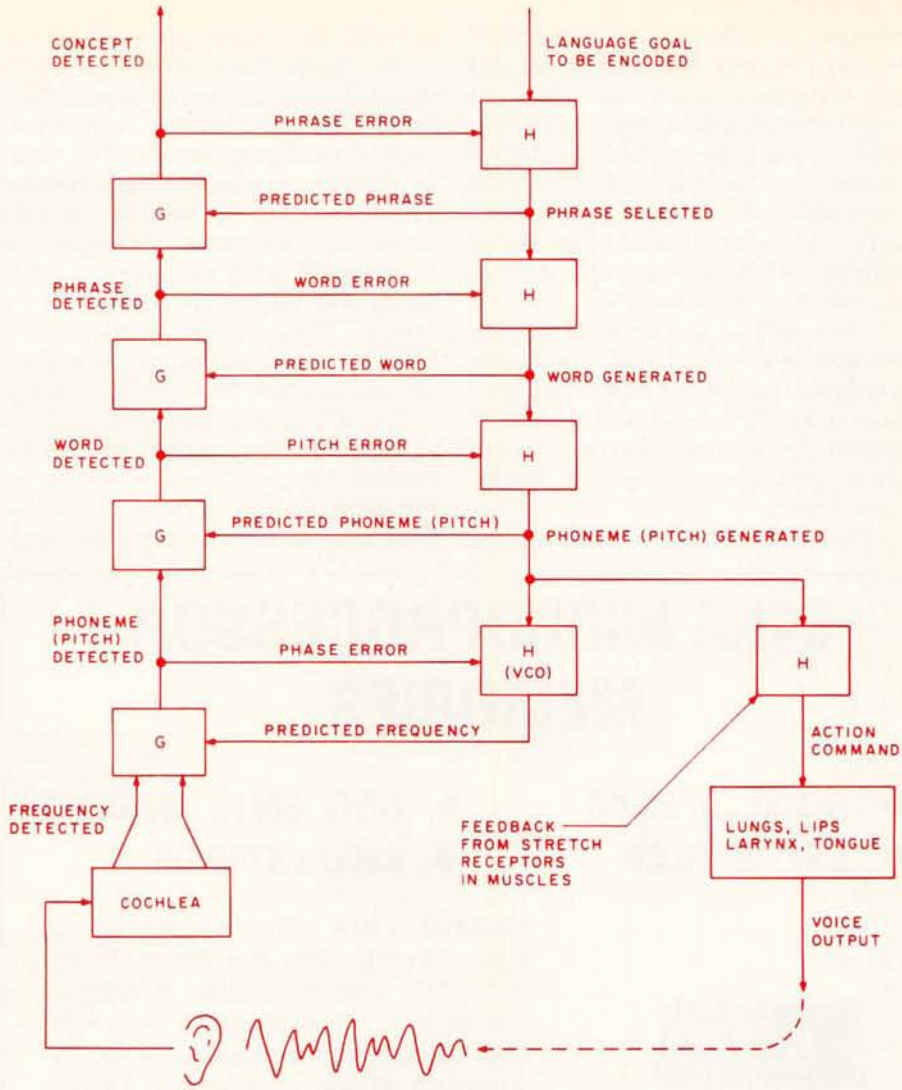



*Figure 7. A crosscoupled hierarchy in the hearing-speech system. The generating hierarchy decomposes language goals into strings of verbal output. When speech is being generated, the sensory processing hierarchy provides feedback to control intensity and modulation. When listening only, the generating hierarchy provides hypotheses and predictions for use in detecting, recognizing, following, and understanding the sensory input.*

communication between individuals to enlist help, to issue commands, to organize group behavior, and to receive feedback information from the sensory experiences of others.

## Writing

Certainly written language, at least, had its origins in goal-seeking activities. For example, the earliest writing in China began around 2000 BC as ideograms or symbols, engraved on bones and shells for the purpose of asking questions of heaven. Each stroke or series of strokes asks a certain question or seeks guidance for a particular branch point in the behavioral trajectory of the life of the asker.

The earliest of all known writing is the Uruk tablets discovered in the Mideast and dated about 3100 BC. This writing appears to be almost exclusively a mechanism for recording business transactions and land sales. These written symbols are now thought to be pictorial lists of tokens used for keeping track of merchandise or livestock. The tokens themselves first appeared 5000 years earlier during the beginning of the Neolithic period in Mesopotamia when human behavior patterns related to hunting and gathering were being replaced by others related to animal husbandry, agriculture, and the village market place.

This token method of accounting apparently served its purpose well, for the system remained virtually unchanged for about 5 millennia until the early Bronze Age when cities and city-states became the most advanced social organizations, and commerce grew into a large scale and complex enterprise. Then the requirements for more efficient accounting procedures led to the pictorial listing of tokens by writing on tablets — an early form of double-entry bookkeeping.

Once skill in this form of writing became widespread and commonly practiced, only a few additional symbols and some rules of syntax were required to express decrees, record dates, and relate accounts of significant events.

Thus, the language skill of writing evolved in small increments over many generations from the goal directed manipulation of physical objects; first the objects themselves, then token objects, and finally images or symbols representing the tokens. The meaning of the symbols, as well as the rules of syntax, were obvious to anyone having an everyday familiarity with the manipulation rules for tokens. These in turn mimicked the rules for manipulation of the objects of merchandise. The manipulation of symbols in written language is a form of goal-seeking behavior which evolved from, and remains similar to, the manipulation of physical objects.

Skill in writing, as any other complex goal-seeking activity, is acquired through painstaking training, endless practice, and numerous corrections of mistakes by a teacher. It is learned in stages, the lowest level primitives first (forming letters), then strings of primitives (words), then strings of strings (sentences), and so on. Only when the rules of spelling, grammar, and composition are more or less mastered can the scribe express or encode a thought (ie: a high level trajectory) into a string of written symbols.

### Speech

The origin of speech is much less certain since it dates from an earlier period. In fact, if we include the sounds of whales, animals, birds, and even insects as a form of speech, spoken language predates the origin of humanity itself. Surely any behavior pattern which communicates a threat, signals submission, expresses fear or acceptance, is a form of language whether it be audible speech or sign language, whether it be expressed by a mouse or a human. By this definition, some speech is very simple — a single facial expression, gesture, chirp, growl, or squeak for each emotional state encoded or intent expressed. Throughout the animal kingdom however, there exists a great variety of modes of expression and many different levels of complexity. Clearly sounds such as the growls, whines, barks, and howls of the wolf express an extremely complex variety of social communications. One can easily feel caught up in a primitive community sing-along when listening to a recording of a wolf-pack chorus.

As we ascend the ladder of behavioral complexity, we find a corresponding increase in the ability to communicate complex messages. In most cases this appears to be not so much an increased vocal capacity as an increased complexity of deep structure underlying overt behavior. This implies that the ability to speak derives, first of all, from having something to say (ie: from having internal trajectories of sufficient complexity that to attach facial expressions, gestures, and audible sounds to them results in complex and subtle messages).

### Primitive Human Speech

The most ancient forms of human speech that survive today are the tribal dances of the few remaining


Circle 355 on inquiry card.

stone-age peoples. In such rites, information on vital subjects such as hunting (including the habits, ferocity, and vulnerable areas of the prey), the proper techniques of stalking, using weapons, etc, are conveyed by dance, symbolic gestures, pantomime, songs, and shouts, as the hunters relate (indeed reenact) the exploits of the hunt. The storytellers replay the behavioral trajectories of their own actual hunting experience and attach verbal symbols and gestures to the portions which cannot be literally acted out.

Even in modern cultures, the majority of everyday speech consists of relating experiences ("...he did this, and I said that...," etc). This is simply the straightforward encoding of behavioral trajectories, or the recalled sensory experiences addressed by those behavioral trajectories, into a string of language tokens or symbols such as gestures, vocal cord, tongue, and lip manipulations. Thus, in the final analysis, all language is a form of goal-directed manipulation of tokens and symbols. The ultimate result is a manipulation of the minds, and hence the actions, of other members of the society. Language is a tool by which a speaker can arouse or implant in the listener a great variety of behavioral goals, hypotheses, and belief structures. By the use of these means, a speaker can command, instruct, threaten, entertain, or chastise other persons in his group to his own benefit and for his own ends.

The implication for research in language understanding is that there is much to be learned from the relationship between language and other forms of behavior. How, for example, can behavioral goals and trajectories be encoded into strings of language symbols for making requests, issuing commands, and relating sensory experiences? How can patterns of trajectories be encoded and transmitted by one processing-generating hierarchy so as to be received and reconstructed by another?

Clearly, language recognition depends on many of the same mechanisms by which the rhythms, periodicities, and harmonic patterns of music, song, and poetry are recognized, tracked, and predicted at many different levels. Consider that children are fascinated by rhythmical sounds, rhymes, and the repetition of familiar stories. Why do adolescents find it so rewarding to hear the same popular song over and over? Is it not the predictability, the lock-on which can be achieved due to a correspondence between the stored internal model and the observed sensory data stream? And why are the rhythmic movements of dancing and marching to music so compelling? Is it not the correlations and harmonic relationships between trajectories in the behavior-generating and sensory-processing hierarchies?

Music is a relatively simple domain for the study of the time dependent interactions between stored models and input data, and the study of music recognition by computer in an almost completely unexplored field. Thus, it is a fertile area for computer hobbyists and other researchers with limited resources.

Part 4 will discuss some operations of the highest hierarchical level such as will, emotion, and creativity. ■

# Exploring TRS-80 Graphics

George H Yeager
223 Riverside Dr
St Albans WV 25177

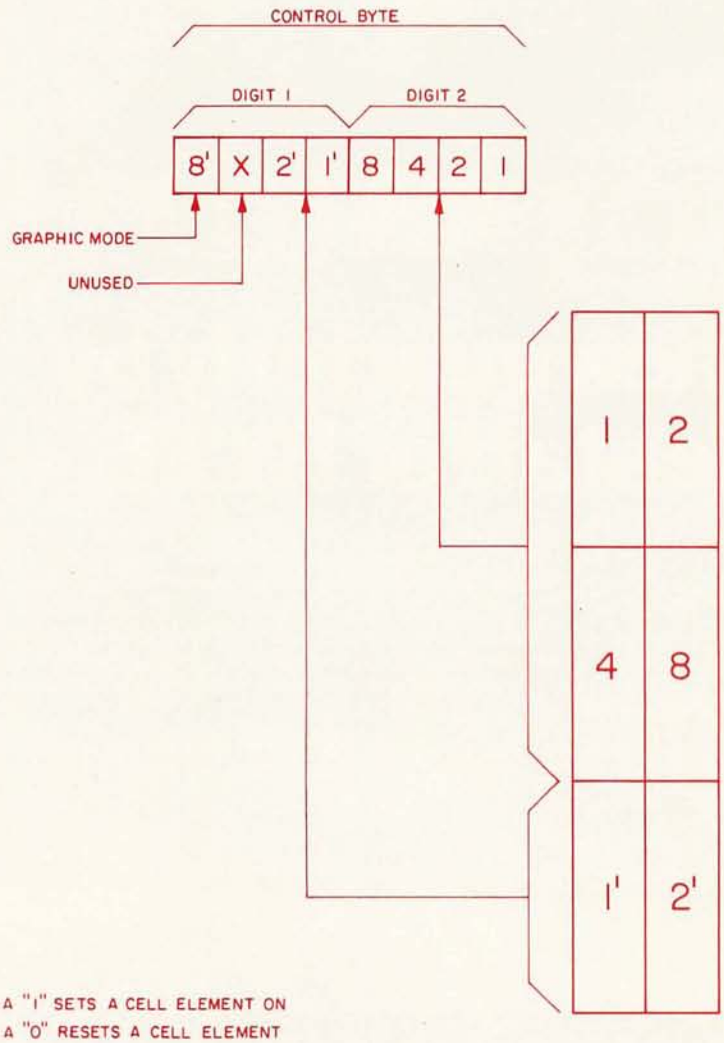



*Figure 1: Cellular division of a graphics cell on the video display screen. The control byte is divided into 2 hexadecimal digits. Individual bits of digit 1 are marked with primes (1'), and bits are designated by their corresponding power of 2 (rather than sequentially). Bit 8' controls the graphics mode. Bit 4' (marked X) is not used.*

Radio Shack seems to hide the neat little jewels of information a hobbyist needs to make a treasure of the TRS-80. One jewel is how to use the computer's graphics capability once you squeeze into the world of machine language by use of the T-BUG monitor. Beyond the excellent *Level 1 User's Handbook,* there has been little information until recently.

Between sessions of disassembling the undocumented control routines for keyboard, video, and cassette, I employed a "crystal ball" to unravel the mystery of machine language graphics control. (TRS-80 owners must be resourceful.) Here is what I found.

First, video display is in main memory address space and resides between hexadecimal locations 3C00 to 3FFF. Address 3C00 corresponds to the upper left corner of the monitor screen and 3FFF to the lower right corner. Anything placed in this block of memory will appear on the display at a specific cell (section of display grid) as a dot-matrix alphanumeric character or as a 6 element graphic character (the TRS-80 hardware does that).

The Radio Shack video display work sheet shows the location of each of the 1024 cells in the video display format. There are 64 cells per line and 16 lines on the page. Figure 1 shows how each cell is divided into six elements for graphics. The bottom two elements are always dark in the alphanumeric mode, providing line spacing.

To activate the graphics mode for a

| | | | |
|---|---|---|---|
| | | Enterprise | |
| 4500 | CD | Call show three; we came here with | *Enterprise* |
| 4501 | OE | H L pointing to symbol table and D E | |
| 4502 | 45 | pointing to graphic cell memory location. | |
| 4503 | 3E | Get constant to point to next line start | |
| 4504 | 3D | | |
| 4505 | 83 | Add it to cell location | |
| 4506 | 5F | Save new LSB line start ADR | |
| 4507 | 30 | Jump if no carry | |
| 4508 | 01 | | |
| 4509 | 24 | Increment H if L carried | |
| 450A | CD | Call show three | |
| 450B | OE | | |
| 450C | 45 | | |
| 450D | C9 | Return to caller | |
| 450E | 01 | | *show three* |
| 450F | 03 | Set B, C to three characters | |
| 4510 | OO | | |
| 4511 | ED | Transfer three characters | |
| 4512 | BO | | |
| 4513 | C9 | Return | |
| 4514 | AO | Graphic symbol table | |
| 4515 | AB | For Enterprise | |
| 4516 | A1 | For Enterprise | |
| 4517 | 8A | For Enterprise | |
| 4518 | 83 | For Enterprise | |
| 4519 | 8B | For Enterprise | |

*Listing 1: Demonstration routine for TRS-80 graphics in Z-80 machine language, for use with T-BUG or other monitor. This displays the starship* Enterprise. *Call this as a subroutine after preserving necessary registers. In the subroutine, registers H and L hold the output table pointer. Registers D and E contain the upper left corner location of graphic symbol within the display memory. Registers A, B, C, D, E, H, and L will be altered. This is meant only as a demonstration; it may not be general enough for other use.*

specific cell on the screen, data with a value of hexadecimal 80 or above must be placed into the memory location with which it corresponds. The most significant bit of the byte sets the graphics mode; placing a value of 7F or lower in a location activates the alphanumeric mode for the related cell.

Looking at figure 1, note that bits 1′ and 2′ of digit 1 control the bottom two elements in the cell. (These read as "one prime" and "two prime"; primes indicate digit 1.) Note also that bits 1, 2, 4, and 8 of digit 2 control the top four cell elements. In the graphics mode, bit 4′ is a "don't care" (ie, it is not used). If the cell element control bit is set to a 1, the element will be lit on the screen. If the element control bit is reset to 0, the element will not be lit.

The element control bits are identified in figure 1 by their decimal weight. The sum of the bits set to 1 in each section of the cell can be converted to hexadecimal to determine the code for each digit in the graphic control byte. Figure 2 (on page 84) shows all graphic characters and the proper generation codes, so that manipulation may be made easier.

The system is simple and flexible, allowing many shapes to be generated with one byte of code. It is unfortunate that the cell shape is unsymmetrical, thus complicating rotation and transformation of graphic displays. However, the mystery is now solved. A whole new world of more finely detailed and faster displays is available for TRS-80 fans.

I have provided a small demonstration program shown as listing 1. Running it under T-BUG will give an idea of the capabilities provided by machine language control of the TRS-80 graphics. Good luck, and let me know what you find out from your crystal ball.

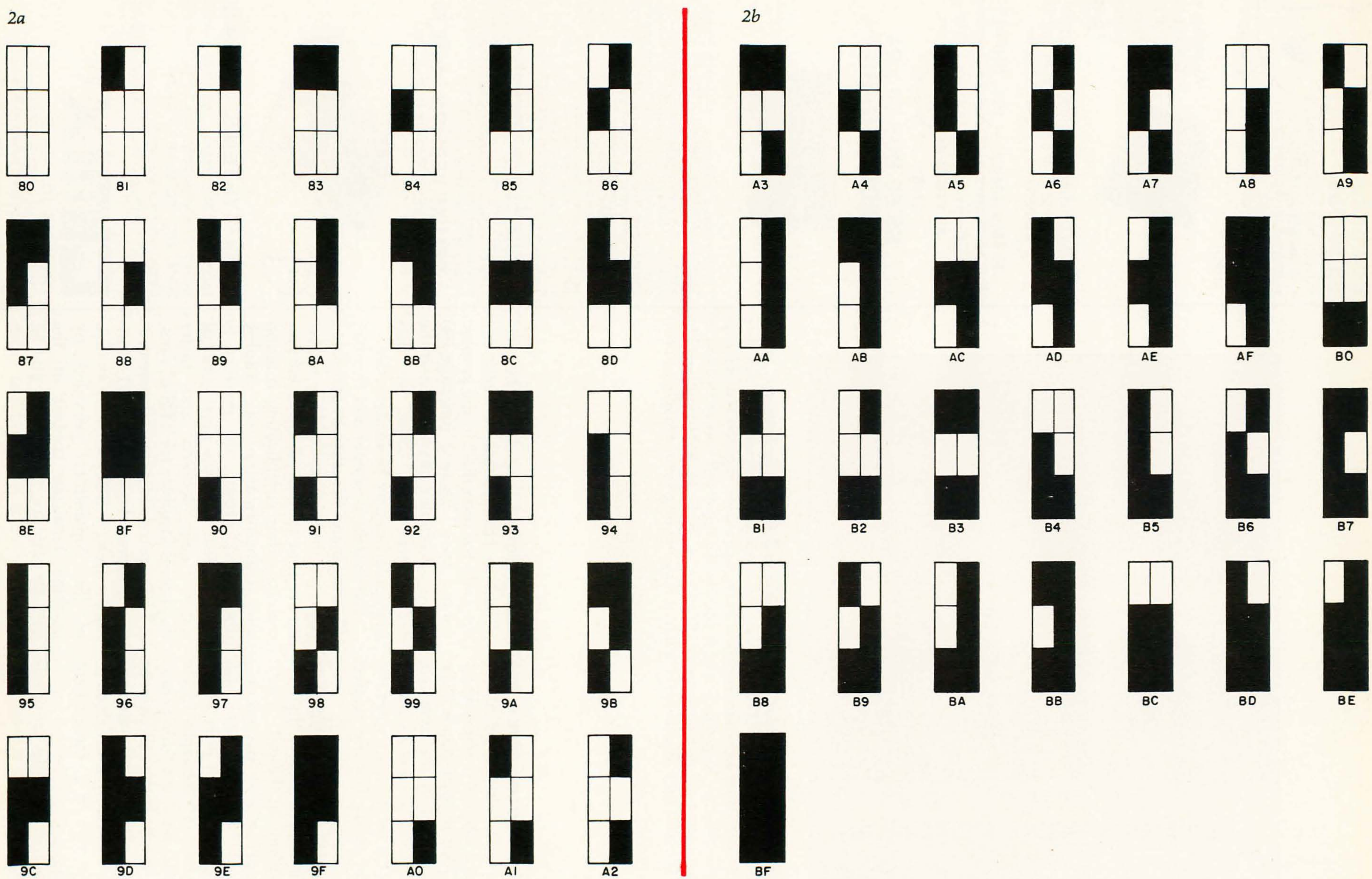



*Figure 2: Graphic patterns produced by values from hexadecimal 80 to BF. If you wish to use POKE graphics in Level II BASIC, you need to convert the values from hexadecimal to decimal.*

# ...And the future

**THE BYTE BOOK OF COMPUTER MUSIC** combines the best computer music articles from past issues of BYTE Magazine with exciting new material—all written for the computer experimenter interested in this fascinating field.

You will enjoy Hal Chamberlin's "A Sampling of Techniques for Computer Performance of Music", which shows how you can create four-part melodies on your computer. For the budget minded, "A $19 Music Interface" contains practical tutorial information—and organ fans will enjoy reading "Electronic Organ Chips For Use in Computer Music Synthesis".

New material includes "Polyphony Made Easy" and "A Terrain Reader". The first describes a handy circuit that allows you to enter more than one note at a time into your computer from a musical keyboard. The "Terrain Reader" is a remarkable program that creates random music based on land terrain maps.

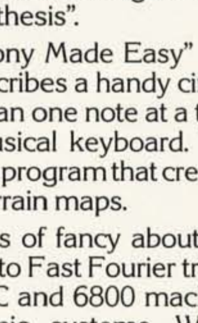
Other articles range from flights of fancy about the reproductive systems of pianos to Fast Fourier transform programs written in BASIC and 6800 machine language, multi-computer music systems, Walsh Functions, and much more.

For the first time, material difficult to obtain has been collected into one convenient, easy to read book. An ardent do-it-yourselfer or armchair musicologist will find this book to be a useful addition to the library.

NEW

ISBN 0-931718-11-2
Editor: Christopher P. Morgan
Pages: approx. 128
Price: **$10.00**

**SUPERWUMPUS** is an exciting computer game incorporating the original structure of the WUMPUS game along with added features to make it even more fascinating. The original game was described in the book **What To Do After You Hit Return,** published by the People's Computer Company. Programmed in both 6800 assembly language and BASIC, SUPERWUMPUS is not only addictively fun, but also provides a splendid tutorial on setting up unusual data structures (the tunnel and cave system of SUPERWUMPUS forms a dodecahedron). This is a **PAPERBYTE™** book.




ISBN 0-931718-03-1
Author: Jack Emmerichs
Pages: 56
Price: **$6.00**

**TINY ASSEMBLER 6800, Version 3.1** is an enhancement of Jack Emmerichs' successful Tiny Assembler. The original version (3.0) was described first in the April and May 1977 issues of BYTE magazine, and later in the **PAPERBYTE™** book **TINY ASSEMBLER 6800 Version 3.0.**


In September 1977, BYTE magazine published an article entitled, "Expanding The Tiny Assembler". This provided a detailed description of the enhancements incorporated into Version 3.1, such as the addition of a "begin" statement, a "virtual symbol table", and a larger subset of the Motorola 6800 assembly language.

All the above articles, plus an updated version of the user's guide, the source, object and PAPERBYTE™ bar code formats of both Version 3.0 and 3.1 make this book the most complete documentation possible for Jack Emmerichs' Tiny Assembler.

ISBN 0-931718-08-2
Author: Jack Emmerichs
Pages: 80
Price: **$9.00**

A walk through this book brings you into **Ciarcia's Circuit Cellar** for a detailed look at the marvelous projects which let you do useful things with your microcomputer. A collection of more than a year's worth of the popular series in BYTE magazine, **Ciarcia's Circuit Cellar** includes the six winners of BYTE's On-going Monitor Box (BOMB) award, voted by the readers themselves as the best articles of the month: **Control the World** (September 1977), **Memory Mapped IO** (November 1977), **Program Your Next EROM in BASIC** (March 1978), **Tune In and Turn On** (April 1978), **Talk To Me** (June 1978), and **Let Your Fingers Do the Talking** (August 1978).

Each article is a complete tutorial giving all the details needed to construct each project. Using amusing anecdotes to introduce the articles and an easy-going style, Steve presents each project so that even a neophyte need not be afraid to try it.

NEW

ISBN 0-931718-07-4
Author: Steve Ciarcia
Pages: approx. 128
Price: **$8.00**

# is right now!

**BASEX,** a new compact, compiled language for microcomputers, has many of the best features of BASIC and the 8080 assembly language—and it can be run on any of the 8080 style microprocessors: 8080, Z-80, or 8085. This is a **PAPERBYTE™** book.

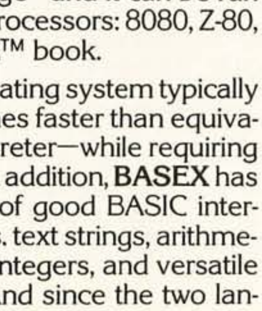
Subroutines in the **BASEX** operating system typically execute programs up to five times faster than equivalent programs in a BASIC interpreter—while requiring about half the memory space. In addition, **BASEX** has most of the powerful features of good BASIC interpreters including array variables. text strings, arithmetic operations on signed 16 bit integers, and versatile IO communication functions. And since the two languages, BASEX and BASIC, are so similar, it is possible to easily translate programs using integer arithmetic data from BASIC into BASEX.

The author, Paul Warme, has also included a BASEX Loader program which is capable of relocating programs anywhere in memory.

NEW

ISBN 0-931718-05-8
Author: Paul Warme
Pages: 88
Price: **$8.00**

**PROGRAMMING TECHNIQUES** is a series of **BYTE BOOKS** concerned with the art and science of computer programming. It is a collection of the best articles from BYTE magazine and new material collected just for this series. Each volume of the series provides the personal computer user with background information to write and maintain programs effectively.




The first volume in the Programming Techniques series is entitled **PROGRAM DESIGN.** It discusses in detail the theory of program design. The purpose of the book is to provide the personal computer user with the techniques needed to design efficient, effective, maintainable programs. Included is information concerning structured program design, modular programming techniques, program logic design, and examples of some of the more common traps the casual as well as the experienced programmer may fall into. In addition, details on various aspects of the actual program functions, such as hashed tables and binary tree processing, are included.

ISBN 0-931718-12-0
Editor: Blaise W. Liffick
Pages: 96
Price: **$6.00**

**SIMULATION** is the second volume in the Programming Techniques series. The chapters deal with various aspects of specific types of simulation. Both theoretical and practical applications are included. Particularly stressed is simulation of motion, including wave motion and flying objects. The realm of artificial intelligence is explored, along with simulating robot motion with the microcomputer. Finally, tips on how to simulate electronic circuits on the computer are detailed.

ISBN 0-931718-13-9
Editor: Blaise W. Liffick
Pages: approx. 80
Price: **$6.00**
Publication: Winter 1979

**RA6800ML: AN M6800 RELOCATABLE MACRO ASSEMBLER** is a two pass assembler for the Motorola 6800 microprocessor. It is designed to run on a minimum system of 16 K bytes of memory, a system console (such as a Teletype terminal), a system monitor (such as Motorola MIKBUG read only memory program or the ICOM Floppy Disk Operating System), and some form of mass file storage (dual cassette recorders or a floppy disk).

The Assembler can produce a program listing, a sorted Symbol Table listing and relocatable object code. The object code is loaded and linked with other assembled modules using the **Linking Loader LINK68.** (Refer to **PAPERBYTE™** publication **LINK68: AN M6800 LINKING LOADER** for details.)

There is a complete description of the 6800 Assembly language and its components, including outlines of the instruction and address formats, pseudo instructions and macro facilities. Each major routine of the Assembler is described in detail, complete with flow charts and a cross reference showing all calling and called-by routines, pointers, flags, and temporary variables.

In addition, details on interfacing and using the Assembler, error messages generated by the Assembler, the Assembler and sample IO driver source code listings, and **PAPERBYTE™** bar code representation of the Assembler's relocatable object file are all included.

This book provides the necessary background for coding programs in the 6800 assembly language, and for understanding the innermost operations of the Assembler.

ISBN 0-931718-10-4
Author: Jack E. Hemenway
Pages: 184
Price: **$25.00**

to order books see next page .....

# BYTE News . . . .

BUBBLE MEMORY ARRIVES FOR PERSONAL COMPUTERS. Rockwell International has introduced a bubble memory board for a personal computer system. The board contains 128 K bytes of storage and plugs directly into the expansion bus for the AIM-6502 processor (which is the same as the KIM-1 bus). Rockwell also supplies a controller card which allows the bubble memory to function as a floppy disk replacement. The controller will control up to 16 memory boards for a total of 2 M bytes of bubble memory. However, before you rush out to buy it, be aware that each bubble memory board costs $2500 and the controller board costs $1000.

Intel and National will also soon become manufacturers of bubble memory. Texas Instruments and Rockwell are currently supplying bubble memories. Texas Instruments and Rockwell devices contain 256 K bits. The Intel device, which will be in volume production in early 1980, will contain 1 M bits, while the National device will contain 256 K bits. Texas Instruments and Rockwell have been producing limited quantities of the bubble memory devices and they do not expect to begin volume production until 1980. Furthermore, one Japanese manufacturer, Fujitsu, appears to be near bubble memory introduction.

MORE LARGE COMPANIES RUMORED ABOUT TO ENTER PERSONAL COMPUTER MARKET. Rumors continue that RCA, Hewlett-Packard and Zenith are seriously considering entering the personal computer market. Each is known to have a personal computer system development project in progress. Other companies seriously investigating the market include IBM and Bell Labs, each of which is known to have personal computer projects at the research facilities.

Several Japanese companies also introduced personal computer systems at the June NCC show in New York, Matsushita introduced its JD-700 to sell for $5,000 to $6,000. It has a 2 K byte read only memory, two minifloppies, and a printer, and it uses Extended BASIC. Sord introduced the M200 ($6,000 to $7,000), which uses a Z-80 with 64 K memory, up to four minifloppy drives, and BASIC, FORTRAN, or COBOL. Ai Electronics showed its APC-20 ($7,500) which is Z-80 based, has two 5 inch drives and hardware arithmetic, and has software options which include FORTRAN, BASIC, COBOL, PL/3 and CP/M.

DIGITIZED HI-FI ON THE HORIZON. An industry group called the "Digital Audio Disk Council" was formed in late 1978 to establish guidelines and standards for pulse code modulation (PCM) recordings. The council includes 35 companies and is an international group. The standard is expected to be adopted in one to two years.

It is expected that pulse code modulation recordings will be the next generation of super hi-fi disks. The technique provides wider frequency response and greater dynamic range, and virtually eliminates distortion and noise. The record will also include an address code for random access of selections. Applications to published software products may well impact the small computer field.

INTEL RETIRES THE 1103. Intel has finally retired the 1103 dynamic memory which houses 1 K bits. This was Intel's first successful MOS memory product and it was a pioneer in the field of IC-MOS memories. Intel has made 35 million of these units since its introduction in 1971.

TI INTRODUCES SPEAKING TRANSLATOR. At the June Consumer Electronics show, Texas Instruments introduced a hand-held language translator which displays and speaks the translated words through the use of a speech synthesizer circuit. This is a significant advance over the Craig and Lexicon units introduced six months earlier, which only display translated words. The unit will cost $250, plus $50 for plug-in language modules. English, Spanish, French and German modules will be available, with Russian, Japanese and Chinese to follow later. The unit displays 1000 words, 500 of which can be spoken. Craig has also increased their module vocabularies to 2,400 words.

UPI NEWS WIRE NOW AVAILABLE TO PERSONAL COMPUTER USERS. United Press International (UPI), one of the prime sources of news used by newspapers throughout the country, has made their service accessible to personal computer users. The UPI wire can be dialed as a local number in most US cities. UPI will charge $15 per hour during business hours, and $2.75 during other times.

IBM DEVELOPS ULTRA-HIGH SPEED LOGIC. The IBM Research Center at Yorktown Heights NY has disclosed their development of logic circuits with switching speeds of 13 picoseconds. Based on

Josephson junction technology, the devices are still in an experimental form. The new circuits are called "Current Injection Logic" and they generate thousands of times less heat than previous types of logic. As a result, higher circuit densities will be possible.

MINIATURE FLOPPY DISKS IN DEVELOPMENT. At present we have 8 inch (20.3 cm) and 5.25 inch (13.3 cm) floppy disks. A new, smaller disk is now well into development and has been proposed for international standardization. Commonly referred to as the *Eurodisk,* it is a square package that measures 4.12 inches (10.5 cm), will store 400 K bytes, use 50 tracks per side, and have a 300 K bps data transfer rate. The standard 5.25 inch (13.3 cm) floppy disk holds 125 K bytes on 40 tracks and has a 125 K bps data transfer rate (double these figures for double density). Olivetti is also expected to announce a very low cost 2.55 inch (6.5 cm) disk which will store 8 K bytes. It will take several seconds to read or write, there is no provision for random file access, it will be thicker, and will not use a jacket. It is rumored to be intended for use in a personal computer that is now nearing introduction. Rumors also continue that IBM will use the 3.25 inch (8.3 cm) disks, currently used in their dictating units, in some of their low end computer systems such as the 5110.

FLAT DISPLAY PANELS SHOWN. At the May meeting of the Society for Information Display, several Japanese companies demonstrated prototype flat panel displays that are now in an advanced stage of development. Ise Electronics showed a 240 character (40 characters by 6 lines) vacuum-fluorescent display that was 250 mm wide by 100 mm high and 14.5 mm thick. It operated off of low voltage and was low power. Hitachi exhibited an 80 character LCD panel which was 280 mm by 50 mm by 23 mm, and operated from 5 VDC and dissipated only 100 mw. NEC showed a storage type LCD panel of 120 characters, and Fujitsu demonstrated a 1560 character plasma display panel.

VIEWDATA AND TELETEXT NEWS. Both the Viewdata and Teletext home data-base access systems will be introduced to the US market by the mid 1980s. Viewdata is a system that connects the home to a central computer via telephone lines. The user can call up data to appear on a modified television. General Telephone and Electronics presently has a Viewdata research development project. Trial systems are already in operation in England and West Germany.

Teletext transmits data on a television signal, fitting the data into the blank space between picture frames. Micro-TV, a Philadelphia-based company has been doing this for over two years, while KSL-TV, Salt Lake City, has done the same for one year. Texas Instruments is supplying the decoders for the KSL test.

The Electronic Industries Association is currently evaluating Teletext. Some companies believe that by the late 1980s the home system will include Viewdata, Teletext, video disk, and a personal computer system to control them. In fact, Apple Computer already offers a service, in conjunction with Dow Jones and Co, which permits Apple owners to display stock market information by dialing a phone number.

Viewdata and Teletext are viewed as complementary services to help bring advanced household management, home environmental control, teaching, and entertainment into the home. Some experts feel that it will be realized in as little as three years.

Oak Industries of Crystal Lake IL recently demonstrated their Teletext system. Called "Videotext," it allows cable television operators to pipe data to subscribers via a microprocessor-based decoder. Each decoder has its own address which allows the cable company to monitor all units. This means that they will know immediately if a set is stolen. The cable company will also be able to cut off non-paying subscribers, thereby rendering stolen units useless.

A Miami-based company, Knight-Ridder Newspapers Inc, has formed a subsidiary named Viewdata Corporation of America, which will undertake a two year, $1.3M test. The Hong-Kong Telephone Company also expects to implement a Viewdata system next year.

The Canadian government and telephone companies are currently testing systems which transmit data over both telephone lines and television signals. One system, constructed by Bell Canada, presently has 25 units in a network, linking together Toronto, Montreal and Ottawa. The units were built by Bell Northern Research. Bell Canada expects to have 1,500 to 2,000 units installed in homes next year. Several others are conducting tests.

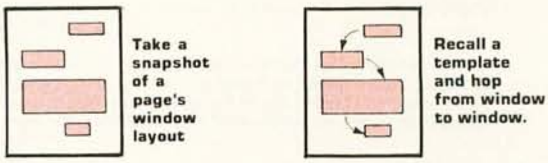
**Sol Libes**
**ACGNJ**
**1776 Raritan Road**
**Scotch Plains NJ 07076**

MAIL: I receive a large number of letters each month as a result of this column. If you wish a response, please include a stamped, self-addressed envelope.

**Wordsmith** is the video text editing system you've been waiting for. Its power, flexibility and simplicity help you carve any text editing task down to size—in a way you can understand. We wanted a system that allows you to think in traditional ways about text layout, yet at the same time makes the traditionally tedious operations such as cut and paste simple and fast. We think we've done it. We want you to decide for yourself

## Flexibility

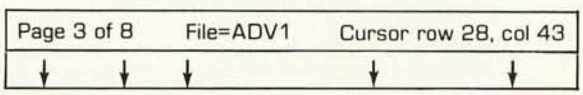
- **Logical/Physical Page Distinction.** Define your own hardcopy size. Wordsmith remembers the difference between the screen size and the hardcopy page size.
- **Modular Hardcopy Driver.** Drive a Qume® Sprint-5 or TTY-like device directly now, Diablo, NEC and other hardcopy devices soon.
- **Pure Text.** Wordsmith files are pure text with no control characters mixed in. This universal format keeps you as compatible with the world as possible. What you see on the screen is what you get as hardcopy.
- **Page Templates.** Snapshots of the block layout of a page can be saved as named disk files, then later recalled and superimposed on the current page. Use such "templates" for standard multicolumn layouts, common letter formats, and fixed-field forms. A single keystroke dispatches you quickly from block to block as you fill in your page.




- **File Switching.** Moving from document to document to examine, copy, move and change text is like rolling off a log. You're not confined to one disk file at a time anymore.

## Simplicity

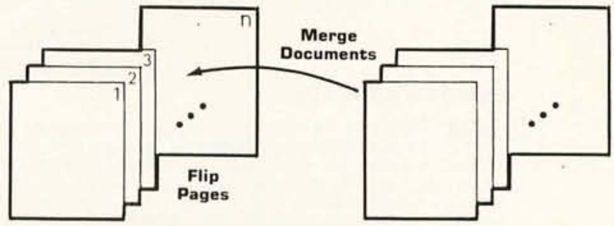
- **Auto Word Break.** Forget the right margin. Wordsmith notices when you won't be able to complete the current word and moves it to the next line for you as you continue typing.
- **Understandable Commands.** The most frequently used commands are single keystrokes. The rest are easily remembered abbreviations.
- **Informative Status Lines.** The top two screen lines constantly display page number information, document name, cursor position, tab stops and status/error phrases. You're always in touch with your document.



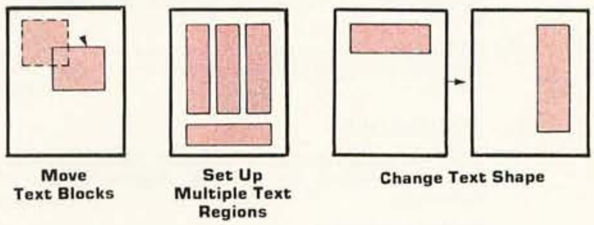

- **Protection Against Catastrophic Errors.** It's nearly impossible to ruin your document with a single bad command. Wordsmith's page oriented design and double-checking user interface help you do what you mean!

## Power

- **Page Oriented Philosophy.** A document is a collection of pages. The screen displays one entire page at a time. Simple random access page flipping commands take you quickly to any page in the document. Equally efficient commands allow you to insert, delete, copy and move pages both within one document and across documents.




- **Extensive Block Manipulation Capabilities.** Using "windows", portions of text, charts, etc., can be quickly and effortlessly moved around on the current page, or across pages. The shape and size of any window can be changed in real time, with the contained text automatically reformatting itself (heeding word and paragraph boundaries) to conform to the new shape.




- **Instantaneous Formatting.** Compacting (extraneous blank deletion) and right justifying are simple commands that tidy up a full page or window's worth of text in the blink of an eye. Random access cursor movement, line and character insert and delete, line and page split and join, and a host of other line and character level commands help you put text in its place quickly and accurately.

## The Wordsmith™ TEXT EDITOR

Defining the New Generation of Text Editing

**from Micro Diversions, Inc.**
**8455-D Tyco Rd.**
**Vienna, Va. 22180**
**(703) 827-0888**

- Direct CP/M® and North Star DOS compatibility
- Available for 40x86, 24x80 and 16x64 memory-mapped video boards
- Fully reentrant for efficient multi-programming environments (6K program space, 5K data area)
- 8080 and Z80 compatibility

**Ordering Information:**
$200
(Screensplitter™ Owners: $80)
Manual only: $15
Check, VISA, Mastercharge

1. CP/M or North Star DOS version?
2. TTY or QUME interface?
3. Brand and memory address of video display board?
4. Ship on single or double density, 5" or 8" diskette?

**Inquire about our custom keyboard.**

# Aw, cut it out!

Save these pages. It's our latest, up-to-the-minute list of super software.

Software with Manual / Manual Alone

## DIGITAL RESEARCH

☐ **CP/M* FDOS** — Diskette Operating System complete with Text Editor, Assembler, Debugger, File Manager and system utilities. Available for wide variety of disk systems including North Star, Helios II, Micropolis, iCOM (all systems) and Altair. Supports computers such as Sorcerer, Horizon, Sol System III, Versatile. Altair 8800, COMPAL-80, DYNABYTE DB8/2, and iCOM Attache. Specify desired configuration ..... **$145/$25**

☐ **MAC** — 8080 Macro Assembler. Full Intel macro definitions. Pseudo Ops include RPC, IRP, REPT, TITLE, PAGE, and MACLIB. Z-80 library included. Produces Intel absolute hex output plus symbols file for use by SID (see below) **$100/$15**

☐ **SID** — 8080 symbolic debugger. Full trace, pass count and break-point program testing system with back-trace and histogram utilities. When used with MAC, provides full symbolic display of memory labels and equated values ...... **$85/$15**

☐ **TEX** — Text formatter to create paginated, page-numbered and justified copy from source text files, directable to disk or printer ...... **$85/$15**

☐ **DESPOOL** — Program to permit simultaneous printing of data from disk while user executes another program from the console ...... **$50/$1**

## MICROSOFT

☐ **Disk Extended BASIC** — Version 5, ANSI compatible with long variable names, WHILE/WEND, chaining, variable length file records ...... **$300/$25**

☐ **BASIC Compiler** — NEW! Language compatible with Version 5 Microsoft interpreter and 3-10 times faster execution. Produces standard Microsoft relocatable binary output. Includes Macro-80. Also linkable to FORTRAN-80 or COBOL-80 code modules ...... **$350/$25**

☐ **FORTRAN-80** — ANSI '66 (except for COMPLEX) plus many extensions. Includes relocatable object complier, linking loader, library with manager. Also includes MACRO-80 (see below) ...... **$400/$25**

Structured Systems prices are discounted!!

☐ **COBOL-80** — ANSI '74 Relocatable object output. Format same as FORTRAN-80 and MACRO-80 modules. Complete ISAM, interactive ACCEPT/DISPLAY, COPY, EXTEND ...... **$625/$25**

☐ **MACRO-80** — 8080/Z80 Macro Assembler. Intel and Zilog mnemonics supported. Relocatable linkable output. Loader, Library Manager and Cross Reference List utilities included ...... **$149/$15**

☐ **EDIT-80** — Very fast random access text editor for text with or without line numbers. Global and intra-line commands supported. File compare utility included ...... **$89/$15**

All Microsoft prices are discounted!!

## XITAN (software requires Z80** CPU)

☐ **Z-TEL** — Text editing language. Expression evaluation iteration and conditional branching ability. Registers available for text and commands. Macro command strings can be saved on disk for re-use ...... **$69/$20**

☐ **ASM** Macro Assembler — Mnemonics per Intel with Z-80 extensions. Macro capabilities with absolute Intel hex or relocatable linkable output modules. New version 3 with added features ...... **$69/$20**

☐ **LINKER** — Link-edits and loads ASM modules ...... **$69/$20**

☐ **Z-BUG** debugger — Trace, break-point tester. Supports decimal, octal and hex modes. Dissassembler to ASM mnemonic set. Emulation technique permits full tracing and break-point support through ROM ...... **$89/$20**

☐ **TOP** Text Output Processor — Creates page-numbered, justified documents from source text files ...... **$69/$20**

☐ **A4 package** includes Z-TEL, ASM, LINKER, Z-BUG, TOP ...... **$299/$40**

## EIDOS SYSTEMS

☐ **KISS** — Keyed Index Sequential Search. Offers complete Multi-Keyed Index Sequential and Direct Access file management. Includes built-in utility functions for 16 or 32 bit arithmetic, string/integer conversion and string compare. Delivered as a relocatable linkable module in Microsoft format for use with FORTRAN-80 or COBOL-80, etc. ...... **$535/$23**

☐ **KBASIC** — Microsoft Disk Extended BASIC with all KISS facilities, integrated by implementation of nine additional commands in language. Package includes KISS.REL as described above, and a sample mail list program ...... **$995/$45**

## MICROPRO

☐ **Super-Sort I** — Sort, merge, extract utility as absolute executable program or linkable module in Microsoft format. Sorts fixed or variable records with data in binary, BCD, Packed Decimal, EBCDIC, ASCII, floating, fixed point, exponential, field justified, etc. etc. Even variable number of fields per record! ...... **$250/$25**

☐ **Super-Sort II** — Above available as absolute program only ...... **$200/$25**

☐ **Super-Sort III** — As II without SELECT/EXCLUDE ...... **$150/$25**

☐ **Word-Master** Text Editor — In one mode has super-set of CP/M's ED commands including global searching and replacing, forward and backwards in file. In video mode, provides full screen editor for users with serial addressable-cursor terminal ...... **$150/$25**

☐ **Word-Star** — Menu driven visual word processing system for use with standard terminals. Text formatting performed on screen. Facilities for text paginate, page number, justify, center, underscore and PRINT. Edit facilities include global search and replace, read/write to other text files, block move, etc. Requires CRT terminal with addressable cursor positioning. Word-Master users may upgrade for $395*** .. **$495/$25**

## SOFTWARE SYSTEMS

☐ **CBASIC-2** Disk Extended BASIC — Non-interactive BASIC with pseudo-code compiler and runtime interpreter. Supports full file control, chaining, integer and extended precision variables etc. ...... **$90/$15**

## STRUCTURED SYSTEMS GROUP

☐ **General Ledger** — Interactive and flexible system providing proof and report outputs. Customization of COA created interactively. Multiple branch accounting centers. Extensive checking performed at data entry for proof, COA correctness etc. Journal entries may be batched prior to posting. Closing procedure automatically backs up input files. All reports can be tailored as necessary. Requires CBASIC ...... **$899/$25**

☐ **Accounts Receivable** — Open item system with output for internal aged reports and customer-oriented statement and billing purposes. On-Line Enquiry permits information for Customer Service and Credit departments. Interface to General Ledger provided if both systems used. Requires CBASIC ...... **$699/$25**

☐ **Accounts Payable** — Provides aged statements of accounts by vendor with check writing for selected invoices. Can be used alone or with General Ledger and/or with NAD. Requires CBASIC ...... **$699/$25**

☐ **NAD** Name and Address selection system — interactive mail list creation and maintenance program with output as full reports with reference data or restricted information for mail labels. Transfer system for extraction and transfer of selected records to create new files. Requires CBASIC ..... **$79/$20**

*CP/M is a trade name of Digital Research
**Z80 is a trademark of Zilog, Inc.
***Good until August 31, 1979

EFFECTIVE JUNE 19, 1979

# The Nature of Robots

## Part 3: A Closer Look at Human Behavior

William T Powers
1138 Whitfield Rd
Northbrook IL 60062

In part 1 of this series, I demonstrated that the concept of *behavior* is not as clear as certain people would indicate. The patterns that we call behavior result from the convergence of many influences, only a part of which can be attributed to the organism that we say is behaving. Yet the behaving organism varies its own actions so that when the influence of these actions is added to all that is unpredictable, the result is recognizable as patterns of behavior.

In part 2 we observed that a control system controls its input, not its output. It acts on its environment to make its own sensory or perceptual signal match a reference signal received from elsewhere, and to automatically counteract the effects of disturbances. It does not have to sense the cause of the disturbance: it senses the quantity it is controlling, and reacts to deviations of that quantity (or the signal representing it) from a reference level that is set by the reference signal.

The reference signal acts just as an intention ought to act. It specifies some state of affairs that is to be achieved, and serves as a target toward which action always urges the perception of the controlled variable. Under normal circumstances the control system can make its perceptual signal track a changing reference signal, and still oppose the effects of disturbances.

There are two main rules of thumb:

- The reference signal reaching a good control system controls the perceptual signal in that system.
- The actions of the control system vary so as to oppose the effects of disturbances, even if the reference signal remains constant.

Let's see how this control system model applies to one small human subsystem: a spinal *reflex* arc (reflex just means "turned back on itself"). This will lead to some concepts that will be of use to the designers of robots.

### The Tendon Reflex

In the early 19th century, Sir Charles Bell established the fact that sensory nerves are separate from motor nerves, and described the "circle of nerves" found in a spinal reflex. A sensory nerve that is part of a spinal reflex arc (we will talk about one that is stimulated by the stretching of a tendon) sends its signal to the spinal cord, and the same cell that receives this signal emits a motor signal that reaches a muscle. When the muscle contracts, it has physical effects that stimulate the same sensory nerve. These are closed loops; the effects of sensory nerves that are stimulated by muscle action affect the same muscle action.

In all such loops that have been discovered, the sense of the feedback is negative. This is true of the tendon reflex. If signals from cells in the spinal cord cause a muscle to contract, the resulting stretch of the tendon stimulates sensors clustered around the tendon. The signals from these sensors reach the same cells in the spinal cord to *inhibit* their firing.

Apparently the materials are present for a control system, but before we discuss this, a digression is necessary.

### All or None or Some

One of the most unfortunate accidents to occur in neurology was the discovery that signals in nerves are carried by impulses. The effect was as if the discoverers of electricity had discovered the electron before they had formulated laws of current flow, and thus developed the whole theory of electricity on the basis of collisions between one electron and another electron. As soon as there were instruments to detect nerve signals it was known that the amplitude of an impulse generated by a nerve cell was independent of the source; there was a trigger effect, so that either an impulse was generated, or it was not.

As a result, almost all neurological research has focused on single impulses. The "all-or-none" principle became so firmly entrenched that by the time digital computers arrived on the scene, most people were led off the track. "Aha," they said, "if a nerve-cell has a threshold that is just high enough, 2 impulses will have to reach it simultaneously to fire it: behold, an AND gate!" Since inhibition (an impulse tending to *reduce* the sensitivity of a nerve cell to an impulse arriving by a different path) can occur, we clearly have the NOT operator, and with the addition of OR (a nerve cell that can be fired by an impulse from any of several paths), we have all of the ingredients for a generalized logic circuit.

There is no longer sufficient reason to believe that the nervous system works in this way. Those who tried to analyze nerve nets as logic devices had to make a lot of assumptions, such as synchronism or clocking, that are incompatible with experimental facts. This more modern under-

*Figure and listing numbering continued from part 2.*

**About the Author**

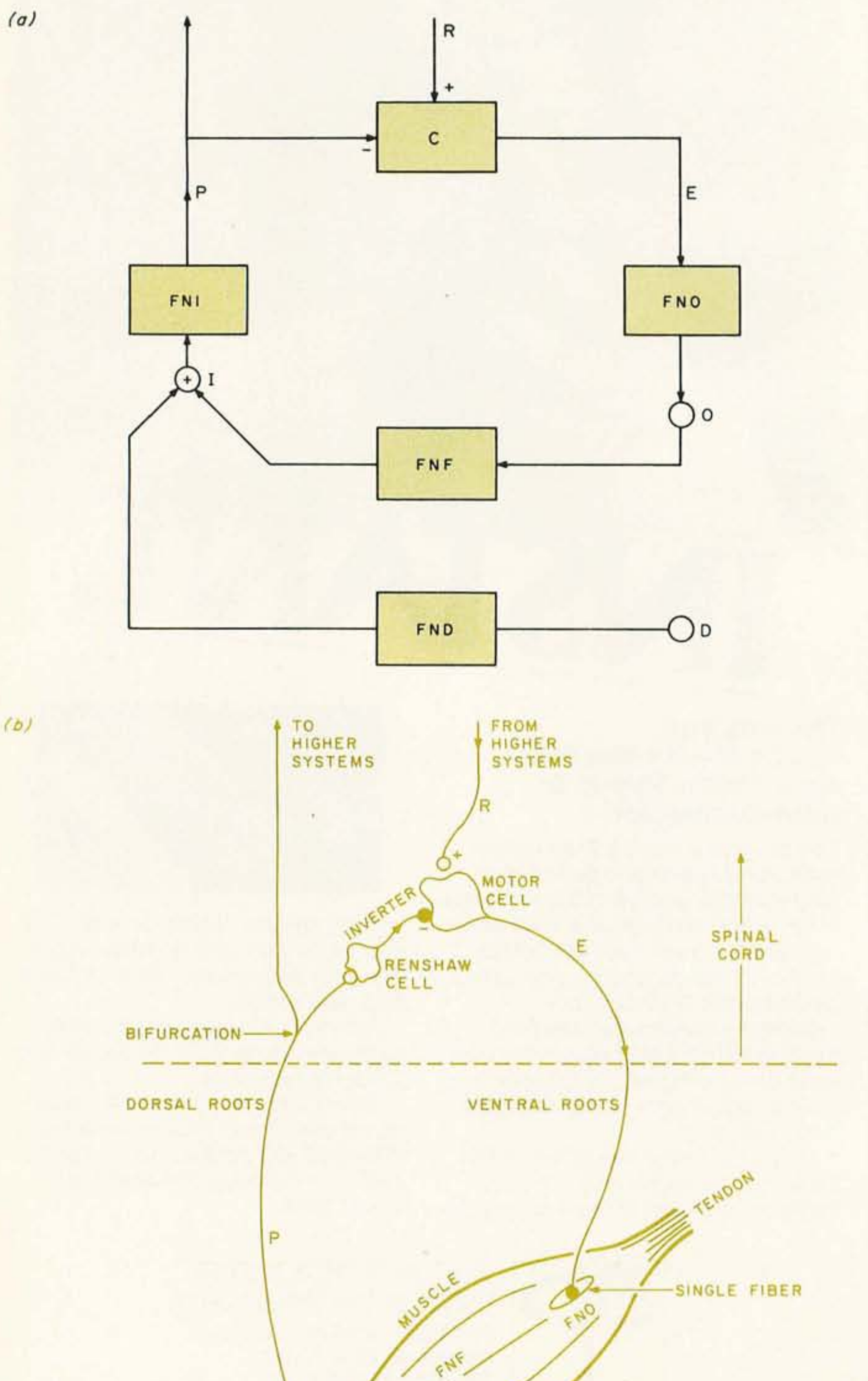
*William T Powers has been exploring the meaning of control theory for studies of human nature since 1953. He spent a number of years (to 1960) in medical physics, and then another 13 (to 1975) as Chief Systems Engineer for the Department of Astronomy at Northwestern University. His occupation has been designing electronic, optical, and mechanical systems for science.*

standing was reflected in Dr Ernest Kent's recent BYTE article series, "The Brains of Men and Machines" (January 1978 BYTE, figure 2, page 16). It now seems that single impulses are not a significant unit of information for most neurons. What counts is *frequency of firing*. The sum of frequencies of excitatory and inhibitory impulses reaching a given neuron has an effect on the rate of that neuron's firing so that the output frequency is a function of a set of input frequencies. Most neurons, in other words, compute *analog*, not digital, functions. As we all know, it is perfectly possible to build digital circuitry out of analog components. Digital integrated circuits are all constructed from analog transistors.

Therefore, when I begin to identify components of a control system, as I will do in a moment, the signals will be thought of as *continuously variable frequencies*, not as on/off binary quantities. The *functions* that combine some signals will be functions of continuous variables. While any one neuron behaves as a rather nonlinear device, a collection of neurons performing essentially the same function in parallel yield an overall pleasantly linear input/output relationship, especially if we consider the normal, rather than extreme range of frequencies (zero or saturation rates of firing).

The spinal reflex systems we will now examine involve several hundred — sometimes several thousand — control systems operating in parallel, although they will be drawn as simple control systems. A perceptual signal is really the mean rate of firing in a whole bundle of pathways, all starting from sensors that are measuring the same input(eg: stretch in a tendon). The signal that enters the muscle in this system is a bundle of signals, each exciting 1 or 2 small fibers out of the thousands that make up 1 muscle. Thus, we will be dealing with neural impulses in much the way electronic engineers deal with electrons. In the majority of cases, the number of impulses passing through a cross-section of a bundle of redundant pathways per unit time will be "the signal," just as the number of electrons passing through a cross-section of a conductor per unit time is called "the current."

*Figure 13: Figure 13a is the standard control-system diagram we have been using in this series. Figure 13b is a spinal reflex arc. FNI is the input function; P, the perceptual signal; C, the comparator; R, the reference signal; E, the error signal; FNO, the output function; O, the output quantity; FNF, the feedback function; I, the input quantity; FND, the disturbance function; and D; the disturbing quantity. Roots are bundles of nerve fibers entering or leaving the spinal cord. An actual spinal reflex arc may involve several hundred systems like the one in figure 13b, with as many motor cells all operating in parallel. Thus, a signal is a bundle of signals that carry similar information.*



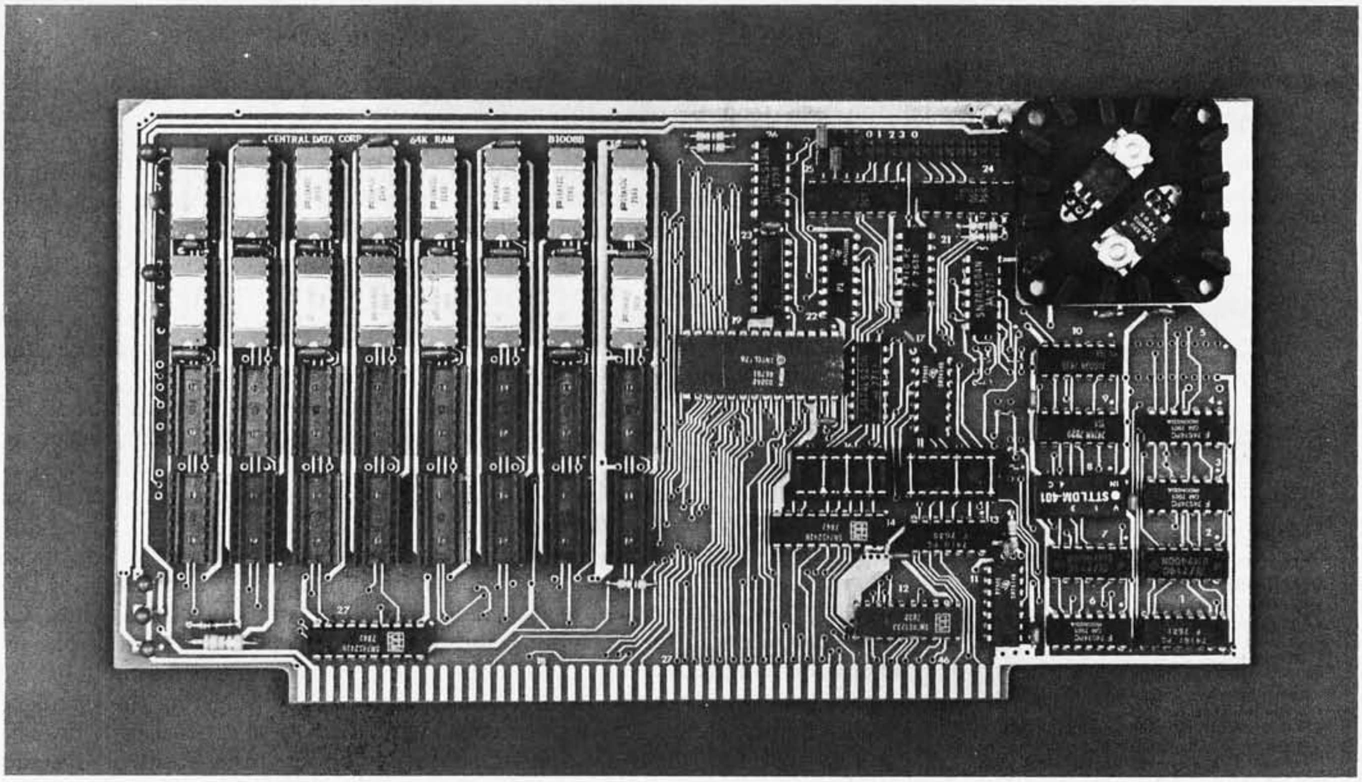

## Level-1 Control System

Figure 13b is a schematic diagram of the tendon reflex. Figure 13a is the diagram of a general control system that I have already shown and discussed earlier. Figure 13a has an input function FNI, a perceptual signal P, a comparator C, a reference signal R, an error signal E, an output quantity O, a feedback function FNF and an input quantity I completing a closed loop. Entering this loop at the same point as the input quantity are the effects of a disturbing quantity D, affected by the disturbance function FND.

Figure 13b contains the same components in the same relationships. The input function is a sensor which emits a signal P, the frequency of which depends continuously on the amount of stretch I of the tendon at the end of the muscle. This signal P travels to the spinal cord, and the local branch enters an *inverter* which is specialized to produce *inhibitory* effects on any neuron it reaches (these actually exist in the spinal cord as Renshaw cells). This inverted copy of the perceptual signal reaches the cell body of a motor neuron C, which also receives an excitatory input from a pathway descending from centers that are higher in the nervous system (the reference signal R).

The signal emitted by this motor neuron represents the excess of excitation over inhibition, and thus represents the difference between the reference and (inverted) perceptual signal: it is clearly the error signal E. The error signal enters the muscle, where it is converted into an average shortening of the contractile fibers in the muscle FNO. The output quantity O is the net stretch of the connective tissue that links the individual contractile fibers together. The feedback function FNF consists of the mechanical relationships that sum all these individual little forces into one force that will tend to stretch the tendon.

I have shown the disturbance as a string that pulls directly on the tendon. It is rather hard to disturb the tendon control system without dissecting the organism, a procedure that always leaves one wondering whether or not this is the original system. The reflex that is tested with a hammer just under the kneecap is a different one, a muscle-length control



32K Board Pictured Above

# New RAM Prices.

## From The Dynamic Memory Company.

| | |
|---|---|
| **16K—$249** | **32K—$375** |
| **48K—$500** | **64K—$625** |

**Ever since we started making these memory boards over a year ago we have continued to lower our prices to stay competitive. Due to your confidence in us, we are again able to lower our prices!** Our reliability has been proven by months of superior performance in thousands of installations. Our low-power boards are being used by quality-minded systems manufacturers across the country and overseas.

**4MHz boards now available.**

After receiving hundreds of requests, our engineering staff has come up with a new version of our board which runs on 4MHz Z-80 systems. It wasn't easy to come up with a high speed board which would operate as reliably as our 450ns version, but after months of careful design and testing, we did it. The price of the 250ns board is $10 per 16K additional.

**All of our features remain.**

Our boards didn't become great sellers only because of the price. We still offer you our deselect feature which allows our RAM to overlap with any fixed memory areas in your system. Also, the RAM area of our board is fully socketed so that you can expand the board yourself.

Other standard features include: plug selectable addressing on 16K boundaries (shorting plugs are placed over wire-wrap pins to address the board — located on the top of the board for easy changes), S-100 and Z-80 compatability and totally invisible refresh — no wait states.

**Fully assembled, tested, and guaranteed.**

All of our boards go through a rigorous testing procedure. They are then placed on burn-in running a series of memory tests to detect any other possible faults. After you receive the board, you are backed by us with a one year warrantee.

**Low power consumption keeps your computer from "losing its cool."**

The total power consumption of our 16K board is typically less than 4 watts (+8V @ 300ma, +16V @ 150ma and −16V @ 20ma). Boards with additional memory typically increase power consumption only 1 watt per 16K!

**Standard S-100 Interface.**

Our board is designed to interface with any standard S-100 CPU. All of the timing of the board is independent of the processor chip, and the board is set up for different processors by changing two plugs on the board.

**Contact your local dealer.**

To find out more about our RAM boards, contact your local dealer. If he is unable to help you, call or write us for a fast response. Central Data Corporation, 1207 North Hagan Street, Champaign, IL 61820. (217) 359-8010

**Central Data**

Circle 45 on inquiry card.

system. Artificially stretching the tendon will tend to relax the muscle, since the feedback is inhibitory.

In part 2 I described how control systems work. We now immediately know what this spinal reflex loop does. It maintains the perceptual signal P matching the reference signal R. Since P is a measure of tension in the tendon, we can say that this control system controls the sensed tension, and not the degree of contraction of the muscle. It also varies the amount of contraction in the fibers of the muscle to oppose any extraneous effects that tend to alter the tension in the tendon, either increasing or decreasing it.

We know that muscles are attached to bones, generally across a joint, and that when a muscle changes tension it often changes the angle at the joint that it spans. In this way movements are created and forces are applied to objects, or against gravitational and other forces. However, this little control system knows nothing of that. The only behavior it produces is sensed tension. It controls a neural signal which represents the net force being created by the muscle and any active disturbances. The control system does not know this — it has, after all, only the one kind of sensor. It knows only how much signal it is getting from the outside world, and not even what *kind* of signal this is. It is just an amount. It would need many other sensors and a very intelligent computer in order to know that this amount is measured in units of tension.

## First Level of Behavioral Control

Every muscle that is used in voluntary behavior (as opposed to internal or visceral) is involved in a control system like that in figure 13b. There are no exceptions. Thus, there is no way that any higher process in the brain can *directly* produce a muscle tension. The brain can produce a muscle tension only by providing a reference signal which specifies how much tension is to be *sensed*. This does not even determine how tense the muscle will be, for if there is a steady external disturbance working, the muscle will adjust its degree of contraction to compensate for the disturbance. Pull steadily on the tendon, and the muscle will completely relax, even with the presence of a nonzero reference signal. Inject Novocain into the perceptual pathway, and the muscle may go into a violent spasm because it is trying to create a perceptual signal. The brain cannot command the muscles to contract. It can only tell level-1 control systems how much tension to sense. It is up to those control systems to do what is necessary to create the demanded signal.

*Gray's Anatomy* names about 200 muscles, most of which occur in pairs, and many of which consist of numerous subdivisions capable of having different effects. There are perhaps 500 to 800 muscles which can be distinguished on the basis of different directions of effect. Thus, we own 500 to 800 level-1 control systems. Every human action must be performed by adjusting the reference signals for these control systems. The behavior of these control systems need not be simulated for the simple reason that this has been done to a sufficient degree in part 2 of this series.

There are actually more level-1 control systems than muscles. For example, every muscle also contains *length* sensors, which are involved in level-1 control systems that govern not force, but something related to the stretching of the muscle itself. Length and force can be controlled quite independently under suitable circumstances; however, we won't be getting into such details here. The main point is that we chew, scratch, talk, walk, run, and swim by using level-1 control systems, and by telling them not what to do, but what to sense.

## Higher Levels of Control

We have accounted for all outgoing signals from the brain that are concerned with overt actions (in the sense that all will act on level-1 control systems, although there may be, at level 1, control systems we haven't considered here). We have not, however, accounted for all incoming signals. The nervous system has hundreds of millions of sensory endings, most of which are not involved in level-1 control systems.

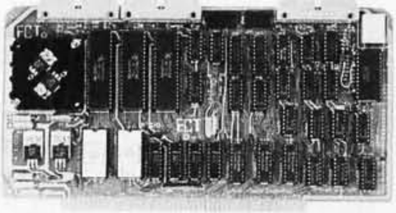
You'll notice that in figure 13b the perceptual signal branches. This is a real branch; all level-1 perceptual signals involved in these control systems branch, sending one branch

*SPECIALIZING IN*

# QUALITY MICROCOMPUTER HARDWARE

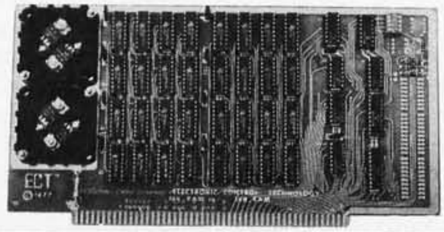
**INDUSTRIAL • EDUCATIONAL • SMALL BUSINESS • PERSONAL**

**BUILDING BLOCKS FOR MICROCOMPUTER SYSTEMS, CONTROL & TEST EQUIPMENT**

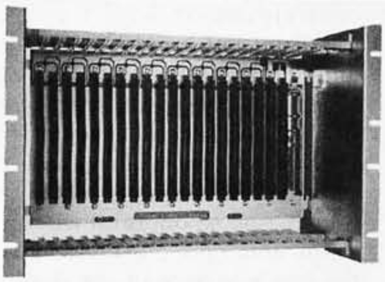


**$R^2$ I/O**

2K ROM
2K RAM
3 Serial Ports
1 Parallel Port



**16 K RAM**
FULLY STATIC
MEMORY



**ECT-100-F**
RACKMOUNT
CARD CAGES


**TT-10**
TABLE TOP
MAINFRAMES

**POWER SUPPLIES, CPU's, MEMORY, OEM VARIATIONS**

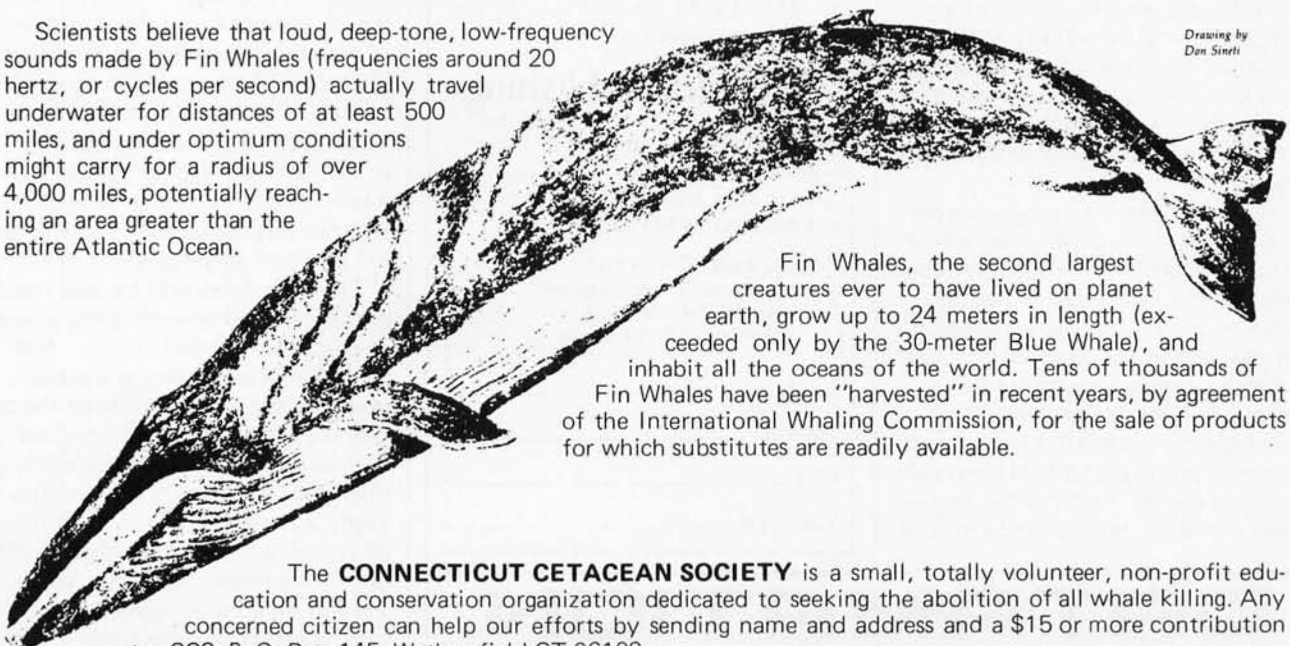
# ELECTRONIC CONTROL TECHNOLOGY

763 RAMSEY AVE.
HILLSIDE, N.J. 07205
**(201) 686-8080**

---

# SAVE THE WHALE

## *The Fin Whale is the world's greatest long-distance communicator.*



*Drawing by Don Sineti*

Scientists believe that loud, deep-tone, low-frequency sounds made by Fin Whales (frequencies around 20 hertz, or cycles per second) actually travel underwater for distances of at least 500 miles, and under optimum conditions might carry for a radius of over 4,000 miles, potentially reaching an area greater than the entire Atlantic Ocean.

Fin Whales, the second largest creatures ever to have lived on planet earth, grow up to 24 meters in length (exceeded only by the 30-meter Blue Whale), and inhabit all the oceans of the world. Tens of thousands of Fin Whales have been "harvested" in recent years, by agreement of the International Whaling Commission, for the sale of products for which substitutes are readily available.

The **CONNECTICUT CETACEAN SOCIETY** is a small, totally volunteer, non-profit education and conservation organization dedicated to seeking the abolition of all whale killing. Any concerned citizen can help our efforts by sending name and address and a $15 or more contribution to: CCS, P. O. Box 145, Wethersfield CT 06109.

upward. Many of the branches — enough to represent what is going on in all the muscles — continue upward to the next level of organization. The perceptual signals from level-1 input functions that are not parts of control systems do likewise. Thus, we can imagine a higher part of the nervous system that is completely surrounded, with regard to input and output, by level-1 systems and input functions.

The signals going downward from this higher part end up in control systems of the general type shown in figure 13b, controlling sensed tension and a few other simple variables. The signals going upward, the level-1 perceptual signals, all reach the next higher level of organization, which happens to be represented in the brain stem, the cerebellum, and one part of the cerebral cortex.

Imagine a second level of control systems. The input functions of this new layer will not be equipped with sensors; instead, they will receive the perceptual signals generated by level-1 input functions (or in the case of signals involved in level-1 control systems, copies of them, courtesy of the bifurcation of the dorsal roots). These signals, in subsets, are the real-time inputs to level-2 input functions, each of which generates one level-2 perceptual signal. We *define* a level-2 input function in terms of the way a single level-2 perceptual signal depends on some set of level-1 perceptual signals.

It is now clearly possible to construct a level-2 comparator, provide it with a reference signal, and make it generate a level-2 error signal. That error signal can then be wired to the input of a level-2 output function, and copies of the output of that FNO can be fanned out to serve as *reference signals for level-1 control systems*.

In fact, we can construct as many level-2 control systems as we like, until we run out of neurons that are located where the level-1 perceptual signals terminate and the level-1 reference signals originate. All outgoing signals that are further inward will be accounted for; they will be level-2 reference signals. (If you can figure out why they can't be level-1 reference signals, bypassing level 2, you are beginning to understand control theory. Hint: Level-1 reference signals are adjusted by level-2 systems: what happens if an *arbitrary* signal is added to the output of a level-2 system?)

Some level-1 perceptual signals may be combined to produce level-2 perceptual signals, without involving the new perceptual signals in any level-2 control system. Perceptual signals that *are* involved in level-2 control systems branch, just as their counterparts at level 1 do: one of the branches heads further inward and upward in the brain. We can now repeat the process of going from the first to the second level of control. Clearly, a third level of control systems can be constructed, then a fourth, and so on, until we run out of brain and find ourselves looking at the inside surface of the skull.

This is my model of the brain. It will be discussed in greater detail in the next article of this series. At present we will develop a clearer understanding of the relationship between one level of control and the next higher level of control through the use of BASIC. As you will see, the relationship has some rather amazing and challenging properties.

### Two-Level Control Hierarchy

We are going to model a very elementary 2-level control system. I won't attempt to model a *real* human system because it would get too complicated. The imaginary system will consist of 3 level-1 control systems, each controlling sensed force (just as in the tendon reflex system) and 3 level-2 systems, each controlling a separate *aspect* of the forces controlled by level-1 systems.

The 3 muscles will be laid out in a plane, one end of each being joined at a common central point, and the other being anchored to a point in the plane. If the angles between the muscles are equal, they will form a Y. We will assume that the common connection does not move; the muscles will apply a force there but, as in the case of flying a stick-controlled airplane, any movement will be negligible. This allows us to ignore some complex interactions between the muscles. Those interactions would not in-

NEECO PROUDLY ANNOUNCES THE COMPUTHINK

# MINIMAX

## "FULLY INTEGRATED COMPUTER SYSTEM"


MINIMAX SERIES COMPUTER

***THE MINIMAX SERIES WAS DESIGNED TO OFFER THE MARKET MINICOMPUTER CAPABILITIES AT MICROCOMPUTER PRICES. COMPARE THE CAPABILITIES & PRICE!***

**CONTACT NEECO FOR FULL SPECS - FREE MINIMAX MANUAL.**

## *MEET THE MINIMAX COMPUTER*

**THE MINIMAX SERIES COMPUTER WAS DESIGNED BY INDUSTRY PROFESSIONALS. COMPARE THE PRICE AND FEATURES TO ANY OTHER COMPUTER IN ITS CLASS!**

• THE MINIMAX SERIES COMPUTER IS AN INTEGRATED, COMPACT UNIT CONTAINING THE CPU, DUAL DENSITY DISK STORAGE, 12 INCH CRT, AND FULL STYLE KEYBOARD, WITH SEPARATE NUMERIC ENTRY PAD. ALL KEYS (INCLUDING CURSOR) WITH FULL REPEAT • HYBRID 2 MEGAHERTZ 6502 CPU • 108K SYSTEM RAM (48K USER) • FASTEST FLOPPY DISK ACCESS (24K LOADS IN 4.2 SECONDS) • 16K ROM CONTAINS COMPUTHINK BASIC (AN EXTENDED MICROSOFT BASIC) WITH EXTENDED PRECISION, DOS INCLUDES COMPLETE FILE I/O WITH FULL RANDOM ACCESS, COMPLETE MONITOR WITH DEBUG & TRACE, AND TINY 6502 ASSEMBLER • COMPLETE HIGH RESOLUTION GRAPHICS WITH INDIVIDUAL DOT (240 x 512) POINT SCREEN ADDRESSABILITY • FULL SCREEN TEXT EDITING WITH OVERWRITE, INSERTION OR DELETION • SPLIT SCREEN/WINDOW MODES • INDIVIDUAL FIELD EDITING WITH FIELD PROTECT AND AUTO SKIP TO NEXT FIELD • DISK STORAGE SYSTEM TRANSFERS 6K PER SECOND WITH AUTO VERIFY AND PARITY CHECK • 12 INCH CRT-64 CHARACTERS BY 30 LINES. UP TO THREE PROGRAMMABLE CHARACTER FONTS FOR LANGUAGES OR SPECIAL CHARACTERS • • • SWITCHABLE 110 OR 220V OPERATION • • • • HYBRID CPU IS MICROPROGRAMMABLE WITH 64 USER DEFINABLE OPCODES. CHOICE OF 800K OR 2.4 MEGABYTE DISK STORAGE • FULL SERIAL RS-232C PORT WITH PROGRAMMABLE BAUD RATES AND MODEM CONTROL SIGNAL • DEDICATED DISK PORT • PRINTER PORT SUPPORTS PARALLEL COMMERCIAL PRINTERS • 24 PIN I/O USER PORT • PAGEMATE DATABASE AVAILABLE • PLM COMPILER AVAILABLE • BUSINESS PACKAGES AVAILABLE • COMPLETE DIAGNOSTICS & SCHEMATICS INCLUDED • COMPLETE USER MANUAL INCLUDED

**MINIMAX I - .8 MEGABYTE ON LINE MINIFLOPPY STORAGE**
**MINIMAX II - 2.4 MEGABYTE ON LINE 8" FLOPPY STORAGE**

***SPECIAL DEALER PRICING AVAILABLE ON DEMONSTRATION MINIMAX AND SOFTWARE PACKAGES TO QUALIFIED SELECTED DEALERS. CONTACT NEECO FOR INFO.***

| | |
|---|---|
| **MINIMAX I -** | **$4495** |
| **MINIMAX II -** | **$5995** |

*THE MINIMAX WAS DESIGNED AND IS MANUFACTURED BY COMPUTHINK COMPUTER CORP. DISTRIBUTED IN EUROPE AND THE EASTERN U.S. BY NEECO.*

PRINTER NOT INCLUDED IN PRICE

IN ADDITION TO HARDWARE CAPABILITIES THAT ARE UNMATCHED IN THE INDUSTRY, THE MINIMAX COMPUTER SUPPORTS A COMPLETE DATA BASE SYSTEM (PAGEMATE), CONTAINING FULL STATISTICAL, SORTING, AND EDIT FUNCTIONS. A PLM COMPILER IS AVAILABLE, FULL BUSINESS SUPPORT SOFTWARE IS AVAILABLE AND MORE!—FULL DEALER SUPPORT IS AN IMPORTANT PART OF OUR MARKETING. CONTACT NEECO FOR FURTHER INFORMATION.

THE MINIMAX IS NOW AVAILABLE FOR QUANTITY DELIVERY. DOMESTIC DEALERSHIPS AVAILABLE. EUROPEAN DISTRIBUTORSHIPS/DEALERSHIPS AVAILABLE TO QUALIFYING COMPANIES WITH SUPPORT CAPABILITIES. SOFTWARE HOUSES AND OEM INQUIRIES INVITED. CONTACT NEECO.

**NEECO**

**NEW ENGLAND ELECTRONICS CO., INC.**
**679 HIGHLAND AVE., NEEDHAM, MA 02194**
**MON-FRI, 9:00-5:30, E.S.T.**

**(617) 449-1760**

TELEX 951021, ANSWERBACK "NEECO"

Circle 281 on inquiry card.

terfere with control, but would make the model very complicated. In simulating a control organization, it is always the simulation of the *environment* that creates complexities. The geometric interactions between the muscles are properties of the world in which these control systems live, not of the control systems proper.

There will be 3 level-1 control systems, 1 for each muscle. Each will sense the force being generated by its own muscle. Each will have a loop gain of 10, and a slowing factor of 0.07 (see part 2 for discussion of these properties).

There will also be 3 level-2 control systems. One will use the 3 muscles to control a force in the X direction (left and right), another will control a force in the Y direction (up and down), and the third will control the sum of the 3 forces, this sum corresponding to what physiologists call "muscle tone." We will see why there is such a thing as muscle tone (the steady mutually cancelling tension that is always there in muscles). Each level-2 control system will have a loop gain of 50, and a slowing factor of 0.01.

I hope that this arrangement looks a little amazing. Here we have 3 muscles spaced at roughly 120-degree intervals around a common point. No one muscle pulls in either the X or the Y direction. To pull in the X direction, all 3 muscles must alter their tensions. To pull in the Y direction, all 3 must alter their tensions. To vary the muscle tone all 3 must once more alter their tensions. We will be able to set reference values for these 3 variables at the same time, throw in a disturbance of arbitrary size and direction to boot, and there will be no interference among the systems that cannot be easily taken care of. Each level-2 force-controlling system will be able to keep its perceptual signal matched to any reference signal, while the others do the same thing at the same time.

It may add interest to know that the outputs from the level-2 systems to the level-1 systems will not be accurately weighted: the only choice will be whether or not a given level-2 output reaches a given level-1 comparator after multiplication by 1, 0, or −1. All 3 level-2 outputs will reach and be added together in all 3 level-1 comparators. The neat separation of X, Y, and tone control is not accomplished by carefully balancing the amount of output sent to each level-1 system. Only the crudest adjustment has to be made on the output side, essentially the choice between positive and negative feedback, with negative always being chosen.

We now come to what is perhaps the most fundamental concept of this theory of brain function. The organization which determines that an X vector, a Y vector, and a tone or scalar force will be controlled is found in the *input* functions, not in the output functions. The organization of behavior is determined by the *perceptual*, not the motor organization of the brain. By the time we finish this installment you will see exactly how that happens.

## Setting Up the Model

Let us start by looking at a typical control system of unspecified level in a hierarchy of control systems. This system will receive multiple input signals from lower-level systems and multiple reference signals from higher-level systems. It will emit just 1 output signal (we will assume that the only need for an explicit output function is to provide error amplification and to smooth; otherwise the error signal could be used directly as the output signal). Figure 14 shows this typical system.

## Perceptual Inputs from Lower Levels

The input function will now be a little too complicated to be represented as a BASIC function since we need a set of weighting factors so that each input can be assigned a weight before summing all of the inputs together. The easiest way to deal with weighting factors for a generalized system is to use a matrix that contains all of the factors for all of the levels. For the input function we designate the matrix as S (for sensory) and write it as:

S(L,J,K),

where: L = level
J = system at that level
K = weight of Kth signal from level L−1.

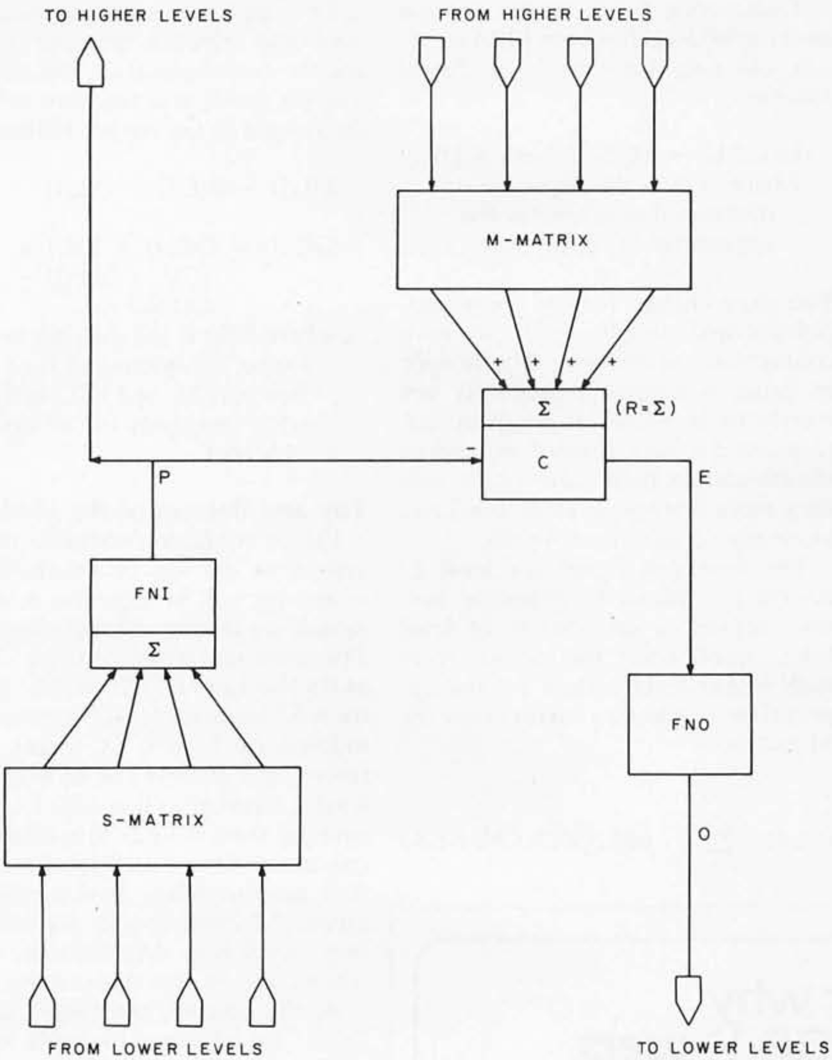



*Figure 14: A typical control system in the middle of a hierarchy of control systems. This system receives multiple reference signals, given a positive or a negative sign by an appropriate entry in the M matrix (no other weighting). The sum of these reference signals is the effective reference signal. The system also receives multiple input signals which are copies of perceptual signals in lower-order systems. These signals are given quantitative weightings by the S matrix and summed in the input function FNI of the system to create this system's perceptual signal P. A duplicate of the perceptual signal travels upward to higher-level systems.*

*The perceptual signal is subtracted from the effective reference signal (or vice versa), and the remainder is emitted by the comparator C as the error signal. The error signal is amplified and smoothed by the output function FNO with the result being emitted to lower-level systems as the output signal O.*

The perceptual signal for this Jth system at the Lth level will be designated P(L,J). The perceptual signal can thus be written as the sum of contributions (weighted) from some set of lower-level systems, a weighting of O in the S matrix meaning absence of a connection:

$$P(L,J)=\sum_{K=0}^{N(L-1)-1} S(L,J,K)\times P(L-1,K)$$

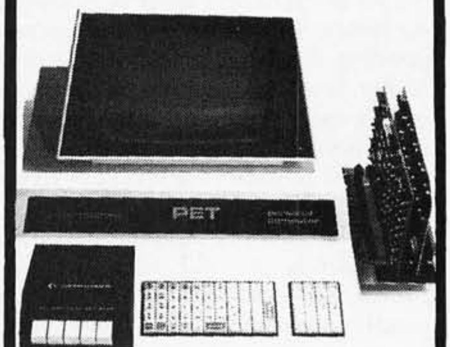
where N(L−1) is the number of systems in the next lower level.

## Reference Inputs from Higher Levels

A similar operation is performed to calculate the net reference signal R(L,J). A matrix M(L,J,K) is used to select a connection factor (1, 0, or −1) for each output of a higher-level system; the net reference signal is the sum of all the outputs of the higher-level systems, each multiplied by its appropriate factor. A 0, of course, means no connection.

The M matrix is filled by looking at the sign of the corresponding entry in the S matrix for the *next higher level*.





To understand how this correspondence is figured, think of the second index in the matrix as the *destination* of the signal, and the third index as the *source*.

Suppose that we wanted to fill in the M matrix for 1 level of systems. An entry will be $-1$ if the corresponding S matrix entry of the next higher level is negative, 0 if the S matrix entry is 0, and 1 if the S matrix entry is positive. But *which* is the entry in the S matrix for level L+1 corresponding to M(L,J,K)?

The answer is simple: M(L,J,K) corresponds to S(L+1,K,J). The source and destination indices are simply interchanged. If a higher-level system gives a negative weight (of any amount) to the perceptual signal from a given lower-level system, it sends a copy of its output to the comparator of *the same lower-level system* with a negative (inhibitory) sign. A negative connection factor means that the output of this higher-level system will subtract from the contributions of other higher-level systems to the lower-level *net reference signal*.

Thus, once the S matrix for the next higher level has been filled in, we can calculate the entries in the M matrix:

M(L,J,K) = SGN (S(L+1,K,J))
where SGN is the *Sign* function that generates the appropriate 1, 0, or $-1$.

You may choose to skip these procedures and simply spell out each connection one at a time. My thought in using a general solution is not merely to save lines of program, but to point the way toward expanding the simulation both horizontally (adding more systems at each level) and vertically (adding more levels).

The reference signal for level L, system J, is found by summing over the outputs of all systems of level L+1, multiplying the output from each higher-level system by the appropriate connection factor from the M matrix:

$$R(L,J) = \sum_{K=0}^{N(L+1)-1} M(L,J,K) \times O(L+1,K)$$

To complete this general model we need only calculate the error signal E and the output signal O. The required slowing factor and the error sensitivity are put in the output function.

E(L,J) = R(L,J) − P(L,J)

O(L,J) = O(L,J) + K(L) × (G(L) × E(L,J) − O(L,J))
where K(L) is the slowing factor for all systems of level L (see part 2), and G(L) is the error sensitivity for all systems of level L.

## Top and Bottom of the Model

We do not have a complete control system at the top of this hierarchy where we will be injecting reference signals for the highest complete level. Therefore we designate those signals as (in this case) O(3,I), output signals from 3 imaginary level-3 systems (us) indexed by I = 0 (X force), 1 (Y force), or 2 (tone). The M matrix for level 2 is set up so that M(2,I,I) is 1, I running from 0 to 2; this establishes connections from each level-3 output to 1 corresponding level-2 reference input. All other entries are left at 0 (my North Star BASIC zeros arrays when they are first dimensioned).

At the bottom, the output signals O(1,I) are supposed to create muscle tensions that affect 3 input quantities; the amount of stretch in the tendon attached to each muscle. To avoid treating a special case, we will designate these input quantities as "level 0 perceptual signals," P(0,I). The value of each input quantity is found by adding the magnitude of the corresponding output to the component of a disturbance that acts along the length of the associated muscle. The value of the input quantity P(0,I) represents the net stretch in a tendon created by the muscle contraction and this component of the disturbance as they act together.

The level-1 S matrix simply connects each input quantity, multiplied by 1, to its respective input function. Thus, we set S(0,I,I) = 1, for I = 0, 1, and 2. All other entries in this matrix are 0.

The geometry of the muscles is adjustable. Since setting up this geometry is the opening phase of the BASIC program, we will take a quick run through this program and discuss the muscle setup. See figure 15 to help

visualize how everything works. Figure 16 is the same system, more closely representing the organization of the brain.

### The Simulator

*Muscle angles.* After the dimension statements and the statements that set slowing factors and error sensitivities for each level have been called, the program calls a subroutine that asks for the angle at which each of the 3 muscles is to be set (in degrees). You can use 30, 150, and 270 degrees (for equal spacing). There is nothing to prevent the choice of any angles you like, although you should draw a diagram to determine the effect on the system. It is hard to create a force in a direction in which there is no component of force from any muscle.

*Sensory weightings.* Lines 9 to 15 organize the perceptions of this system, and thus organize its behavior. For values of I from 0 to 2, all 3 levels of sensory matrix are set up. You can now see how X and Y forces are sensed. The weights for level 2, system 0, correspond to the *cosine* of the angle between the positive X axis and the angle of each muscle. Those for level 2, system 1, correspond to the *sine* of the same angles. Each input function is weighting the perceptual signals from the muscles according to the *component* of force that is aligned with the direction being sensed. The tone system, level 2, system 2 adds the signals together to yield a total-force signal.

*Motor weightings.* Lines 19 to 23 use the already entered values of the S matrices to create the connection matrix M. The sign function selects the sign that will preserve negative feedback.

*Highest-level reference signals.* In line 24, the program calls a subroutine that asks for 3 reference signals: one designating the amount of X force, another designating the amount of Y force, and a third designating the sum of forces, or muscle tone. Positive or negative numbers are allowed. A real nervous system cannot handle negative frequencies, but the same effect can be created by suitable use of inverters so that one (positive) frequency means a positive quantity and another (also positive) frequency means a negative quantity. In reality there would be 6 systems of level 2 in this 4-quadrant system.

I have set up level 1 to behave realistically like a muscle control system; neither negative signals nor negative forces can be produced.

*Disturbance.* At line 25, the program calls a subroutine which asks for the amount and direction of a constant disturbance. A disturbance might be created by seizing the place where the 3 muscles join, moving it, and holding it in the new position. Despite the fact that the control systems are neither detecting nor controlling position, arbitrary movement of this junction in space will stretch or relax the muscles, creating changes of force due to the spring constants of the muscles. Therefore it is reasonable to suppose that a force disturbance can be created, one which projects into the direction of each muscle according to the cosine of the angle between the disturbance vector and the axis of the muscle.

*Calculating the behavior.* Lines 29 through 37 call a subroutine that actually does the calculation of signals in all 6 control systems. You will notice 3 nested FOR-NEXT loops. The outer 2 loops cause the lower-

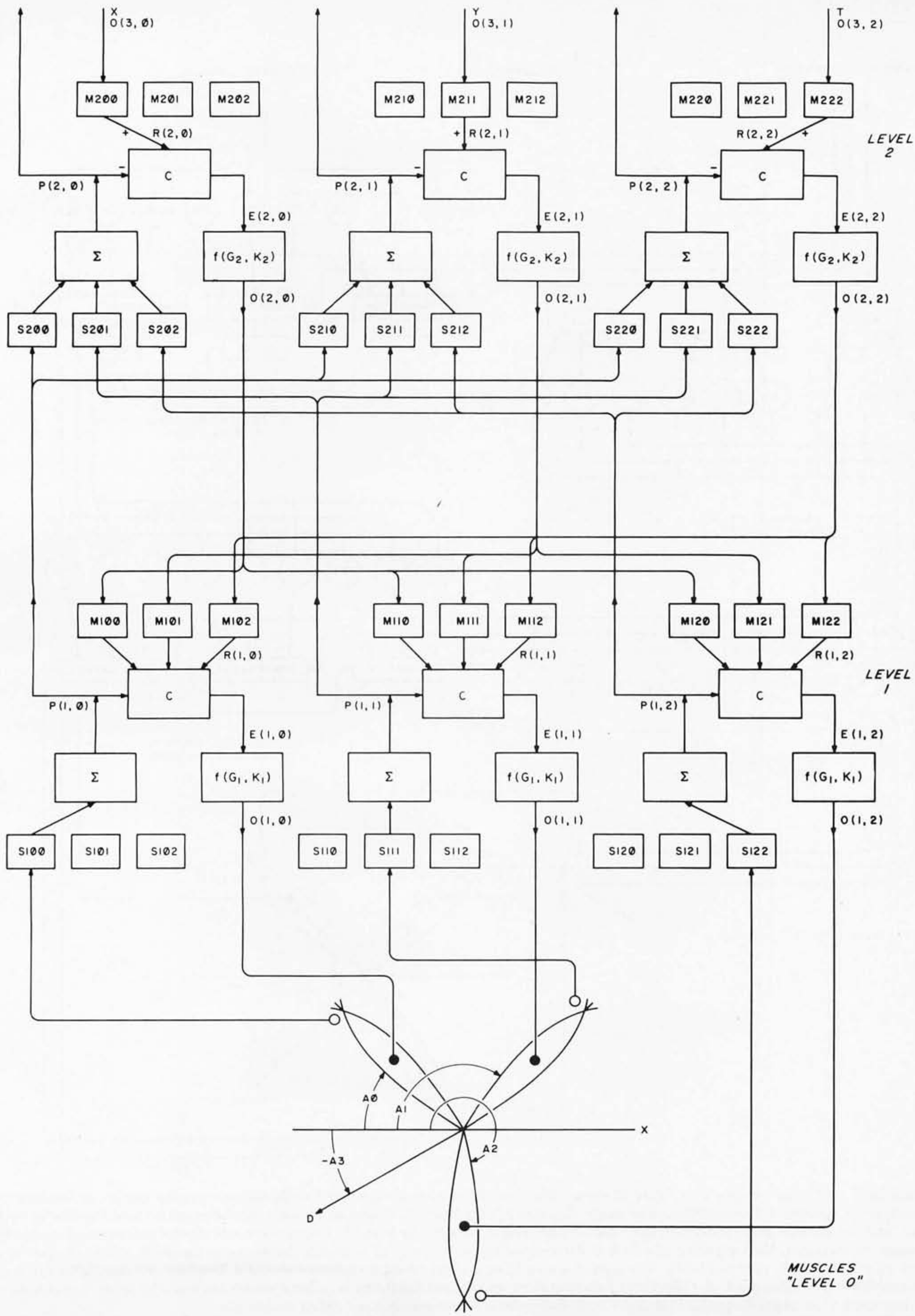
*Text continued on page 111*

*Figure 15: The 2-level hierarchy simulated in this article. Three level-1 systems each control the amount of tension in 1 muscle, as represented by the 3 level-1 perceptual signals. Copies of these 3 perceptual signals reach all 3 level-2 systems, where they are weighted and summed so as to represent the X component of muscle force (P(2,0)), the Y component of muscle force (P(2,1)), and total muscle force or muscle tone (P(2,2)).*

*Each second level system sends an amplified and smoothed version of its error signal as an output signal to all 3 lower-level systems. Each output signal splits into 3 identical branches, 1 for each level-1 system. When a branch reaches a level-1 comparator, it may be connected directly or through an inverter before being summed with other reference inputs. There is no other weighting of output signals. If necessary, an inverter is used to preserve negative feedback for a particular path.*

*Each level-1 system amplifies and smooths its error signal to make an output signal reaching just 1 muscle.*

*A higher-level system determines the reference signals for X, Y, and total force. These are specified by the operator of the simulator. All systems correct their own errors simultaneously.*


X
O(3, Ø)
Y
O(3, 1)
T
O(3, 2)
M2ØØ
M2Ø1
M2Ø2
M21Ø
M211
M212
M22Ø
M221
M222
R(2,Ø)
R(2,1)
R(2,2)
LEVEL 2
C
P(2, Ø)
P(2, 1)
P(2, 2)
E(2,Ø)
E(2,1)
E(2,2)
Σ
f(G2, K2)
O(2,Ø)
O(2,1)
O(2,2)
S2ØØ
S2Ø1
S2Ø2
S21Ø
S211
S212
S22Ø
S221
S222
M1ØØ
M1Ø1
M1Ø2
M11Ø
M111
M112
M12Ø
M121
M122
R(1,Ø)
R(1,1)
R(1,2)
LEVEL 1
P(1,Ø)
P(1,1)
P(1,2)
E(1,Ø)
E(1,1)
E(1,2)
f(G1, K1)
O(1,Ø)
O(1,1)
O(1,2)
S1ØØ
S1Ø1
S1Ø2
S11Ø
S111
S112
S12Ø
S121
S122
AØ
A1
A2
-A3
X
D
MUSCLES "LEVEL O"

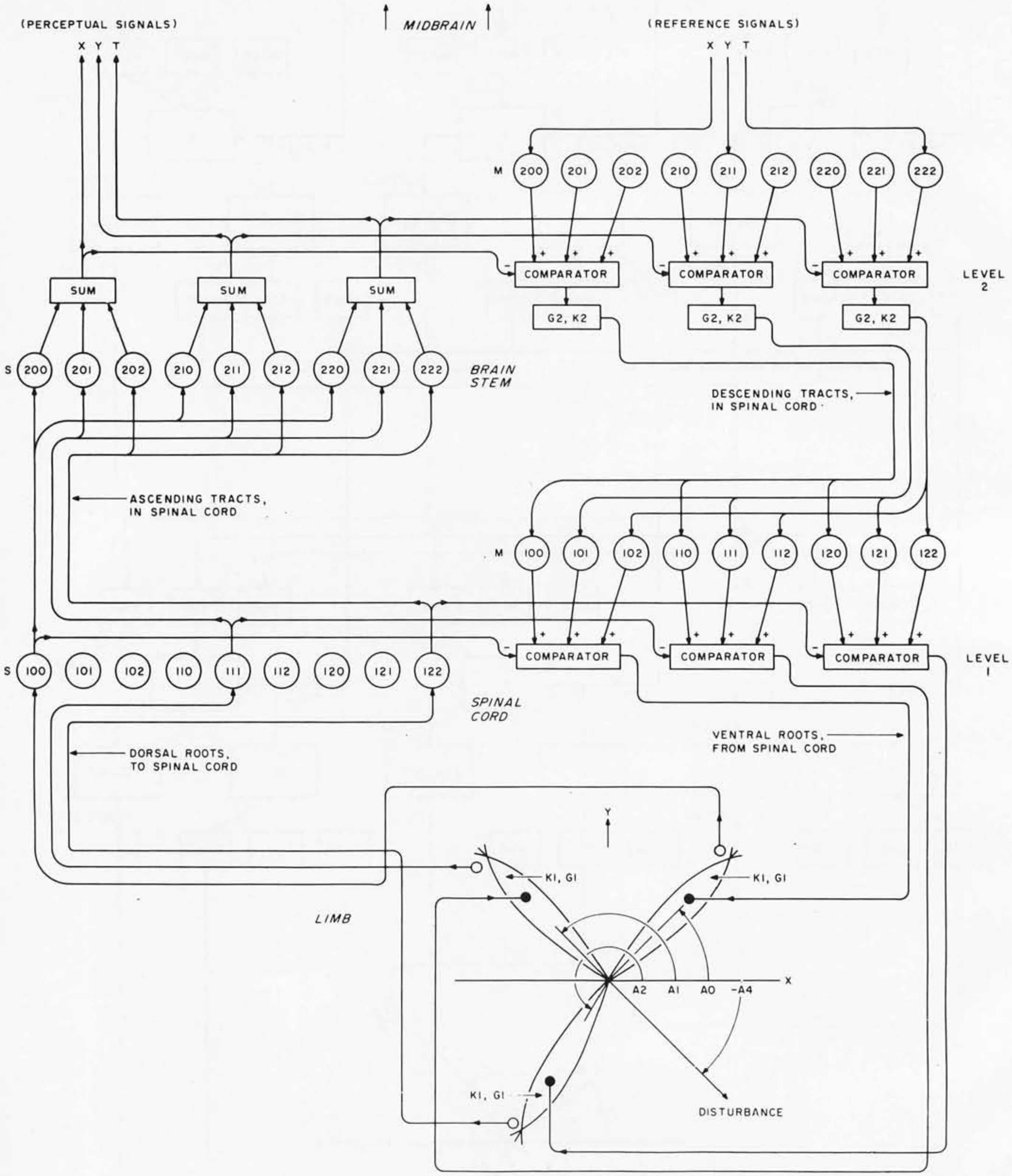
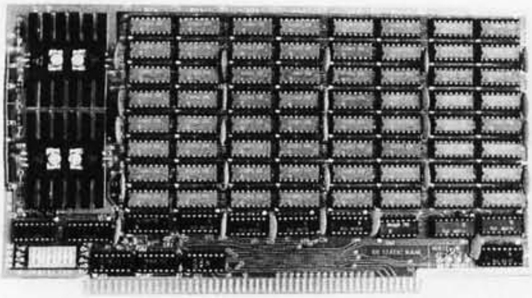



*Figure 16: Topological transform of figure 15 shows how control systems are arranged in the human nervous system, at least according to some cybernetic theoreticians. The major difference from figure 15 is that all sensory functions are lumped together at each level, and comparison and output functions are also lumped together. The S and M matrices are represented in a nervous system as synaptic connections, the weighting of which is determined by the number of branches (from one to hundreds) that form just as a nerve fiber reaches the next cell body. The sign of a weighting is determined by whether or not a Renshaw cell (specialized to produce inhibition) is interposed. A collection of comparators and output functions is called a motor nucleus. For level 2 and higher, the branches of perceptual signals that cross over and enter a motor nucleus are called collaterals.*

# Wondering which memory is best for you?

## base 2• *offers the following products to the S-100 market at the industry's lowest prices:*

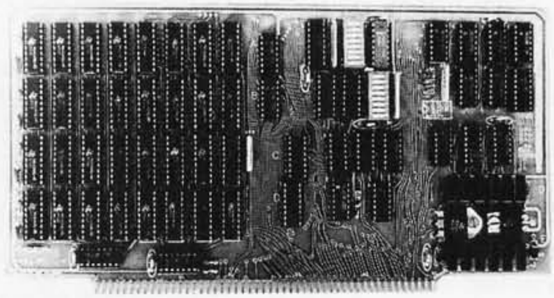


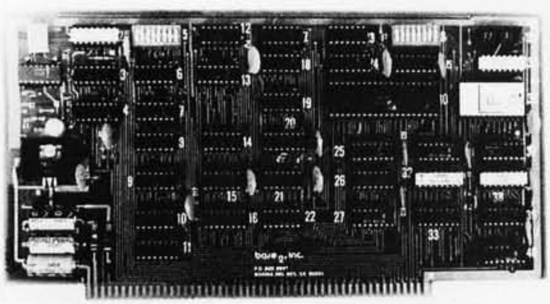
### *8K Static Memory Board*

This 8K board is available in two versions. The 8KS-B operates at 450ns for use with 8080 and 8080A microprocessor systems and Z-80 systems operating at 2MHz. The 8KS-Z operates at 250ns and is suitable for use with Z-80 systems operating at 4MHz. Both kits feature factory fresh 2102's (low power on 8KS-B) and includes sockets for all IC's. Support logic is low power Schottky to minimize power consumption. Address and data lines are fully buffered and 4K bank addressing is DIP switch selectable. Memory Protect/Unprotect, selectable wait states and battery backup are also designed into the board. Circuit boards are solder masked and silk-screened for ease of construction. These kits are the best memory value on the market! Available from stock . . . **8KS-B $125** (assembled and tested add $25.00)
**8KS-Z $145** (assembled and tested add $25.00)

### *16K Static Memory Board*

Base 2 can now offer the same price/performance in a 16K static RAM as in its popular 8K RAM. This kit includes 8K bank addressing with 4K boundary address setting on DIP switches. This low power unit provides on-board bank selection for unlimited expansion . . . No MUX board required. Using highest quality boards and components we expect this kit to be one of the most popular units on the market. Available in two speed ranges, the 16KS-B operates at 450ns while the 16KS-Z operates at 250ns.
**16KS-B $275.00** (assembled and tested)
**16KS-Z $300.00** (assembled and tested)



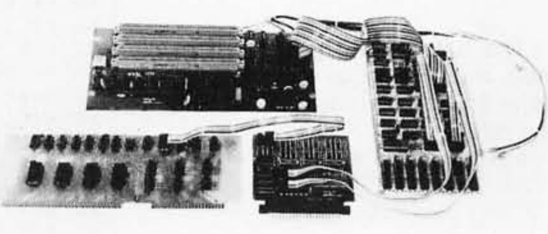


### *Z-80 CPU Board*

Our Z-80 card is also offered in two speed ranges. The CPZ-1 operates at 2MHz and the CPZ-2 operates at 4MHz. These cards offer the maximum in versatility at unbelievably low cost. A socket is included on the board for a 2708 EPROM which is addressable to any 4K boundary above 32K. The power-on jump feature can be selected to address any 4K boundary above 32K or the on-board 2708. An On-board run-stop flip-flop and optional generation of Memory Write allows the board to run with or without a front panel. The board can be selected to run in either the 8080 mode, to take advantage of existing software, or in the Z-80 mode for maximum efficiency. For use in existing systems, a wait state may be added to the M1 cycle, Memory request cycle, on-board ROM cycle, input cycle and output cycle. DMA grant tri-states all signals from the processor board. All this and more on top quality PC boards, fully socketed with fresh IC's. **CPZ-1 $110 CPZ-2 $125**

### *S-100 for Digital Group Systems*

This kit offers, at long last, the ability to take advantage of S-100 products within your existing Digital Group mainframe. Once installed, up to four S-100 boards can be used in addition to the existing boards in the D.G. system. The system includes an "intelligent" mother board, ribbon cables to link existing D.G. CPU to the DGS-100 board and a power wiring harness. The DGS-100 is designed to fit in the 5-3/4" x 12" empty area in the standard D.G. cabinet. It may seem expensive but there's a lot here! End your frustration! **DGS-100 $250.00**



# base 2. inc.

Send for more details on these products. Get on our mailing list for information on more soon to be announced products at factory-direct prices from BASE 2. Why pay more when you can get the best at these prices???

**P.O. Box 3548 • Fullerton, Calif. 92634**
**(714) 992-4344**
**CA residents add 6% tax**
**MC/BAC accepted • FOB — U.S. destination**

Circle 29 on inquiry card.

*Listing 3: North Star BASIC simulation of a 3-muscle system. The muscles have 3 operations they are to perform: movement in the X direction, movement in the Y direction, and tone control. A sample run of the simulator is shown in listing 4. The exclamation point is used as an abbreviation for the PRINT statement.*

```
1 DIM P(2,2),R(2,2),E(2,2),O(3,2),S(3,2,2),M(2,2,2),A(3),K(2)
2 DIM G(2)
3 G(1)=10\ K(1)=.07\ G(2)=50\ K(2)=.01
4 P=3.1415927/180
5 GOSUB 99\ REM        (SET UP MUSCLE GEOMETRY)
6 REM           *************************
7 REM           SET UP SENSORY WEIGHTINGS
8 REM           *************************
9 FOR I=0 TO 2
10 S(1,I,I)=1
11 S(2,0,I)= COS(A(I))
12 S(2,1,I)= SIN(A(I))
13 S(2,2,I)=1
14 S(3,I,I)=1
15 NEXT I
16 REM          *************************
17 REM          SET UP MOTOR WEIGHTINGS
18 REM          *************************
19 FOR L=1,TO 2
20 FOR I=0 TO 2
21 FOR J=0 TO 2
22 M(L,I,J)=SGN(S(L+1,J,I))
23 NEXT J\ NEXT I\ NEXT L
24 GOSUB 109\ REM    (SET UP REFERENCE SIGNALS)
25 GOSUB 116\ REM    (SET UP DISTURBANCE)
26 REM          *************************
27 REM          CALCULATE SYSTEM BEHAVIOR
28 REM          *************************
29 !\FOR Q=1 TO 5
30 FOR J3=0 TO 1
31 L=2\ GOSUB 50\ REM  CALCULATE SYSTEMS AT LEVEL L
32 FOR J2=0 TO 1
33 L=1\ GOSUB 50
34 FOR I=0 TO 2
35 P(0,I)=O(1,I)+D*COS(A(I)-A(3))
36 NEXT I\ NEXT J2\ NEXT J3
37 GOSUB 69\ REM   (PRINT TABLE OF VALUES)
38 NEXT Q
39 !"(A)NGLE? (R)EFS? (D)IST? (C)ONT? (P)RINT MATRICES? "
40 INPUT "",A$
41 IF A$<>"A" THEN 42\ GOSUB 102\ GOTO 29
42 IF A$<>"R" THEN 43\ GOSUB 109\ GOTO 29
43 IF A$<>"D" THEN 44\ GOSUB 116\ GOTO 29
44 IF A$<>"C" THEN 45\ GOTO 29
45 IF A$<>"P" THEN 46\ GOTO 76
46 !" ???? "\ !\ GOTO 39
47 REM        *********************************
48 REM        CALCULATIONS FOR LEVEL L SYSTEMS
49 REM        *********************************
50 FOR J=0 TO 2
51 V=0
52 FOR K=0 TO 2
53 V=V+P(L-1,K)*S(L,J,K)
54 NEXT K
55 IF L=1 AND V<0 THEN V=0
56 P(L,J)=V\ V=0
57 FOR K=0 TO 2
58 V=V+O(L+1,K)*M(L,J,K)
59 NEXT K
60 R(L,J)=V\ V=O(L,J)
61 E(L,J)=R(L,J)-P(L,J)
62 V=V+K(L)*(G(L)*E(L,J)-V)
63 IF L=1 AND V<0 THEN O(L,J)=0 ELSE O(L,J)=V
64 NEXT J
65 RETURN
66 REM        ***********************
67 REM        DATA LISTING SUBROUTINE
68 REM        ***********************
69 !\!"ITERATION # ",%2I,Q,"     ----------------------------"
70 FOR J=2 TO 1 STEP -1,
71 !\! "LEVEL ",%2I,J,%#7F2
72 FOR I=0 TO 2\!"       ",R(J,I),"        ",\ NEXT I
73 !\FOR I=0 TO 2\!" ",P(J,I),"   ",O(J,I)," ",\ NEXT I
74 !\ NEXT J
```

*Text continued from page 106:*
level system to iterate twice for every iteration of the higher-level system. This proves to be an exceedingly useful, easy way to stabilize the 2-level system. (I have also tried this with a 3-level system, and it worked just as well.) I have no formal rationale for why this works; informally, it seems to be a good idea to let the lower-level system correct most of its error before the higher-level systems take their own errors seriously.

The inner loop, line 35, simply calculates the values of the input quantities for the level-1 systems, using the angles of the muscles and of the disturbance. This is, in effect, the simulation of the environment (the muscles are in the environment of a neural control system).

At line 37 a routine is called which prints out the signals for all systems: the reference signal on 1 line, the perceptual signal to the lower left of it, and the output signal to the lower right for each system. Line 38 closes the iteration loop; 5 iterations are called for.

Lines 39 through 46 ask what action is to be taken after 5 iterations.

*Calculation subroutine.* Lines 50 to 65 calculate the signals for each system. The V that occurs here and there is simply a way to reduce the number of times a subscript has to be calculated. The perceptual signal is calculated first, then the reference signal, the error signal, and the output signal, for each system of level L. The level is set at lines 31 and 33 by the calling program. Line 62 contains the slowing routine which appeared in part 2. Lines 55 and 63 determine whether or not level 1 is being calculated; if it is, the perceptual and output signals are prevented from going negative.

```
75 !\ RETURN
76 !\!"SENSORY MATRIX"\ !
77 FOR L=1 TO 2
78 !"LEVEL ",%1I,L
79 FOR J=0 TO 2
80 !"     ",
81 FOR K=0 TO 2
82 !%6F2,S(L,J,K),
83 NEXT K
84 NEXT J
85 !
86 NEXT L
87 !\!"MOTOR MATRIX"\!
88 FOR L=1 TO 2
89 !"LEVEL ",%1I,L
90 FOR J=0 TO 2
91 !"     ",
92 FOR K=0 TO 2
93 ! %6F2,M(L,J,K),
94 NEXT K
95 NEXT J
96 !
97 NEXT L
98 !\ GOTO 39
99 REM            **********************
100 REM            SET UP MUSCLE GEOMETRY
101 REM            **********************
102 !\!"MUSCLE ANGLES:"
103 INPUT1 "#1\ ",A(0)\ INPUT1 "  #2\ ",A(1)\ INPUT1 "  #3\ ",A(2)
104 A(0)=A(0)*P\ A(1)=A(1)*P\ A(2)=A(2)*P
105 RETURN
106 REM           ***********************
107 REM           SET UP REFERENCE SIGNALS
108 REM           *************************
109 !\!"REFERENCE SIGNALS:"
110 INPUT1 "X: ",O(3,0)\ INPUT1 "  Y: ",O(3,1)
111 INPUT1 "  TONE: ",O(3,2)
112 RETURN
113 REM           *************************
114 REM           SET UP DISTURBANCE & ANGLE
115 REM           *************************
116 !\! "DISTURBANCE:"
117 INPUT1 "MAGNITUDE: ",D\ INPUT1 "  ANGLE: ",A(3)
118 A(3)=A(3)*P
119 RETURN
READY
```

*Listing 4: A sample session with the simulator in listing 3. When the simulator is initialized, the user is allowed to set up several values: the 3 muscle angles, the reference signals, and the disturbance magnitude and angle. For each iteration the values for level 1 and level 2 are output in the following form. First the reference signal for the particular muscle is printed. The perceptual signal is printed on the next line, just to the left of the reference signal, and the output signal is printed to the right. This is repeated for every muscle.*

```
RUN


MUSCLE ANGLES:
#1\ 30  #2\ 150  #3\ 270
REFERENCE SIGNALS:
X: -30  Y: 40  TONE: 175
DISTURBANCE:
MAGNITUDE: 0  ANGLE: 0

ITERATION #  1     ----------------------------
                              REFERENCE
LEVEL  2             PERCEPTUAL SIGNAL  OUTPUT
      -30.00            SIGNAL  40.00  SIGNAL          175.00
  -18.19      -20.76     38.50       20.55     187.25       80.50

LEVEL  1
       80.29                 121.81                 39.19
   74.52       73.35    109.52      110.46     37.83        36.14

ITERATION #  2     ----------------------------

LEVEL  2
      -30.00                  40.00                 175.00
  -32.12      -19.13     45.65       10.29     163.72       61.33

LEVEL  1
       52.49                  90.75                  31.91
   47.36       47.64     82.67       82.54      27.25       28.61

ITERATION #  3     ----------------------------

LEVEL  2
      -30.00                  40.00                 175.00
  -29.56      -18.68     37.28       12.56     177.48       67.63

LEVEL  1
       61.51                  98.87                  36.40
   55.96       55.93     89.92       89.89      33.67       33.22

ITERATION #  4     ----------------------------

LEVEL  2
      -30.00                  40.00                 175.00
  -29.54      -18.83     40.19       12.57     172.81       65.13

LEVEL  1
       58.87                  96.52                  33.73
   53.51       53.52     87.72       87.74      30.57       30.64


DISTURBANCE:
MAGNITUDE: 40  ANGLE: 135

ITERATION #  1     ----------------------------

LEVEL  2
      -30.00                  40.00                 175.00
  -72.05        2.40     82.15       -8.75     173.67       65.75

LEVEL  1
       59.40                  54.60                  76.90
   52.56       63.30     63.87       16.98      57.11       93.27
```

*Listing 4 continued on page 114*

*Data listing subroutine.* This subroutine is called after every complete iteration of both levels. It prints only the perceptual signal, reference signal, and output signal from the 3 systems at each level.

**Running the Program**

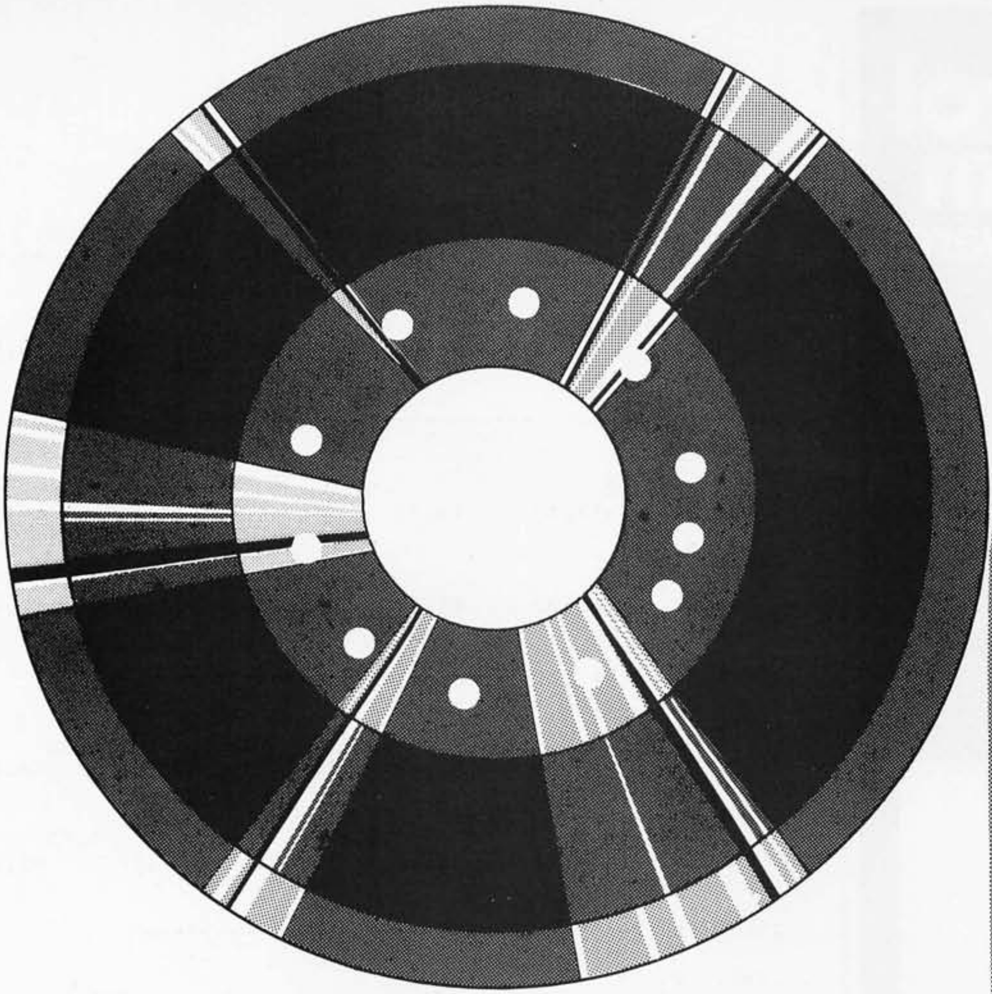
After the RUN command is given, the program asks for all adjustable parameters and then does 5 iterations, printing out the values of all signals each time. It then issues a prompting message, the answer to which determines what happens next. The C command means do 5 more iterations. The P command causes the sensory and motor matrices to be printed out. To get an idea of the time scale on which human level-1 and level-2 systems work, imagine that each iteration takes about 1/20 of a second. (If you are looking for mental exercise, you might adapt the plotter from part 2 to show the variables in this simulation.)

**What the Simulator Shows**

There has always been a problem in conventional models of the brain that have to do with coordinated actions. The standard description is that something high in the brain thinks of a general command like "push!" and sends the equivalent signals downward toward lower systems. Those lower systems receive the general commands, and *elaborate* on them, turning them into more detailed commands at every step. At the lowest level, all of the detailed commands converge into the *final common pathway*, the relatively few channels running from the spinal cord to the muscles. There, at last, the neural signals are turned into tensions that create motions that create behavior.

The problem that nobody has ever been able to figure out is how a simple general command gets turned into specific commands *that will have effects that satisfy the general command*. Unfortunately, neurology is full of sentences that sound like explanations but are really restatements of the effect that is to be explained. When such sentences are uttered, they create the impression that the problem has been solved and needs no further investigation.

The simulator described here shows a different way for commands to get turned into actions. The command that specifies an X force doesn't



# It isn't in the hole it's on the doughnut!

## INTRODUCING — LOUMAR MANAGEMENT SYSTEMS SOFTWARE

**4** programs to give you a comprehensive package for business management

The Loumar General Accounting System is a versatile, fully integrated software package designed for small and medium sized businesses. It is also suitable for CPA's and bookkeeping service films.

The complete software system is composed of four main modules: GENERAL LEDGER, ACCOUNTS RECEIVABLE, ACCOUNTS PAYABLE AND PAYROLL. Each module may be used separately or in combination with any other module. Supplied on disk as run-time modules. Source not available.

All software is written in CBASIC II and utilizes the powerful CP / M operating system.

General system features include:

Automatic posting to general journal • Strict error detection • Report production on demand • Consistent operating procedures • User oriented. No previous computer knowledge required • Designed by accounting professionals • Comprehensive, well presented reports and manuals • Single or multiple client capabilities.

**HARDWARE SPECIFICATIONS**

The end user's microcomputer must satisfy the following requirements:

48k RAM
Dual floppy disk system
Printer with tractor. All printing is done in 80 col. format
CRT with at least a 64 character by 16 line display
CP / M and CBASIC II

**PAYROLL:** Up to 500 employees — **$550.**

**ACCOUNTS RECEIVABLE:** Up to 1000 customers and 1000 monthly transactions — **$550**

**ACCOUNTS PAYABLE:** Up to 1000 vendors and 1300 monthly transactions — **$400**

**GENERAL LEDGER:** Up to 200 accounts with 2000 entries — **$450**

Write for our brochure — Dealerships still available
Contact: Distributor

**MISSION CONTROL • 2008 WILSHIRE BOULEVARD, SANTA MONICA, CA 90403 • (213) 829-5137**

Circle 237 on inquiry card.

*Listing 4 continued from page 112:*

```
ITERATION #  2       ---------------------------

LEVEL  2
       -30.00                40.00               175.00
  -12.87      -16.21     21.01      20.17    180.28      62.52

LEVEL  1
        66.48                98.91                26.14
   59.89       69.94     90.08      54.02     25.92      50.56

ITERATION #  3       ---------------------------

LEVEL  2
       -30.00                40.00               175.00
  -31.36      -17.12     49.55      10.51    167.41      64.63

LEVEL  1
        58.02                92.26                37.01
   52.07       62.22     87.97      48.88     29.19      58.92

ITERATION #  4       ---------------------------

LEVEL  2
       -30.00                40.00               175.00
  -29.97      -16.26     37.42      11.18    175.04      66.18

LEVEL 1
        61.10                93.62                38.75
   54.44       64.92     88.54      49.97     32.97      61.01

ITERATION #  5       ---------------------------

LEVEL  2
       -30.00                40.00               175.00
  -29.55      -18.39     39.87      11.75    173.88      64.95

LEVEL  1
        60.31                93.10                36.81
   53.94       64.25     88.17      49.52     30.93      59.18
```

simply get partitioned among the muscles. It is a request for a *perception*, not a command to *act*. The system receiving this request perceives the X force through a *convergent*, not a *divergent* network. A divergent network cannot be treated as a function; a convergent network can. When the perceived X force matches the reference X force, the cause of the perception must be in one of the states that will, in fact, create that component of force in the X direction. There is an infinity of different muscle tensions that could create the same component of force. If I were not also specifying 2 *other* functions of force, there would be no way to predict the exact muscle tensions that would exist when the X control system experienced zero error.

Since we are specifying 3 functions of 3 variables, and setting reference levels for the value of each function, there is only one state of the muscles that will allow zero error in all 3 systems at once. What we have done, in fact, is set up an analog computer for the simultaneous solution of 3 equations in 3 variables.

This simulator shows that the reference signals for the lower-level systems do not correspond to any one output from a higher-level system. Nevertheless, the perceptual signal sensed by each higher-level system matches the corresponding reference signal. The higher systems each sense a different function of the set of lower-level perceptual signals. Independent control is possible only because the functions represent independent dimensions of variation of the lower-level world.

In the environment of this 2-level system, there is no such thing as X force, Y force, or tone. There are simply 3 tendons in various states of tension. I have *created* the idea of

these 3 forces, by designing input functions that will sense them. I could have made one system that would sense force along a set of curved lines representing direction, and another that would sense force along a different set of curved lines crossing the first set; a coordinate system without any straight lines in it. This would result if the sensors were nonlinear, as we know they are. It would have made no difference, except for the fact that there would not have been a simple label like X force to assign as a meaning for the perceptual signals. It would still be possible to specify 3 reference signals and thus set the 3 perceptual signals to specific values, thereby creating a specific state of tension in all 3 tendons that would automatically resist disturbances. The *way* in which the external situation is represented is almost immaterial, as long as 3 reasonably independent perceptual functions are created. There is no coordinate system in the outside world. The behaving system makes up one of its own.

If there were sensors on each muscle to detect muscle length as well as force, we could add 3 more control systems at level 1, and 3 more independent aspects of the external world to control at level 2. In fact, there *are* muscle-length sensors, and I am working on several models that take them into account.

If you now imagine 500 to 800 muscles involved with at least twice as many level-1 control systems (length and force surely; rate of change highly likely), you will begin to perceive the richness of the world in which level-2 systems exist. Add to this the millions of sensors for heat, cold, vibration, joint angle, light, sound, taste, smell, hunger, pain, illness, angular acceleration, joint compression, and so on, and you might begin to glimpse the complexity of the real system we are modeling. Since perceptions that arise from sources other than direct effects of muscles exist in large numbers, there can clearly be far more level-2 systems than level-1 systems, although the number of level-2 systems that can act independently at the same time is limited by the total number of comparators available at level 1.

Perhaps you can now see why this approach to a model of a human being (rudimentary is it is at this point) has some powerful implications for the building of robots. I suggest a formal distinction between a *robot* (an imitation of a living system) and an *automaton* (a device which automatically produces complex actions). An automaton is designed to create preselected *movements*; a robot is designed to control preselected *perceptions* (its own). In order for an automaton to produce precise and repeatable behavior, it must be built so strongly that normal disturbances cannot alter its movements, or it must be protected from disturbances that might interfere with its movements. In order for a robot to create, for itself, precise and repeatable perceptions (and thus precise and repeatable *consequences* of behavior), it need only perceive precisely, have a sufficiently high error sensitivity, and be capable of producing forces as large as the largest disturbances that might reasonably occur.

There is much more that can be said about the general relationship of one level of control to another, but this installment has raised enough points to ponder. To prepare for part 4, you should run this simulator and observe what happens to all of the variables in it. Try keeping the disturbance constant in magnitude and rotating its angle; try altering the muscle angles; change line 3 to use different error sensitivities (G(x)) and slowing factors (K(x)). Use the C command for longer iterations, and convince yourself that a steady state has really been reached. See what happens if the muscle tone isn't set high enough (there is a very good reason for muscle tone control). Do a series of iterations with slowly changing reference signals, and plot muscle tension against each reference signal. Get the feel of this small extract of the whole human hierarchy because in part 4 we will widen the field of view to include everything, and we will begin to look at some experiments with human subjects. These experiments will be noninvasive, nondestructive — more like video games than science — but far more useful than the games. ■

*Text continued from page 16:*

### Conditional Expressions

Clearly, the meaningful use of predicates and recognizers requires the existence of language constructs to modify the program flow. Such constructs are called *control structures.* One basic control unit in LISP is called the *conditional expression.* In M-LISP it is written:

$$[<p_1> \rightarrow <e_1>; <p_2> \rightarrow <e_2>; \ldots t \rightarrow <e_n>]$$

The meaning of such a conditional expression is as follows:

> Each $<p_i>$ is a predicate; the $<e_i>$s are arbitrary LISP expressions. We evaluate the $<p_i>$s from left to right, finding the first which evaluates to true. The value of the conditional expression is the value of the corresponding $<e_i>$. If none of the $<p_i>$ s are true, then the value of the conditional is $<e_n>$. Notice that this last case is really forced upon us since the last predicate is the constant *t*. It is common to read *t* used in this context as "otherwise."

We extend our M-LISP to S-LISP mapping to include this new construct, mapping it to:

$$\begin{aligned}&(COND\ \ (<\text{predicate}_1>^T\ <\text{expression}_1>^T)\\ &\qquad\quad (<\text{predicate}_2>^T\ <\text{expression}_2>^T)\\ &\qquad\quad \ldots\\ &\qquad\quad (T<\text{expression}_n>^T))\end{aligned}$$

The evaluation of a conditional expression is different from the technique we have used in previous LISP instructions. Previously we have insisted that we evaluate all of the operands in an instruction. In the conditional expression, we evaluate the minimal part of the conditional which gives us a true predicate; then we evaluate the corresponding expression.

For example: *(COND ((ATOM 'A) 'FOO) (T 1))* gives value *FOO,* since *(ATOM 'A)* gives *T. (COND ((ATOM '(A)) 'FOO) (T 1))* gives value 1 since *(ATOM '(A))* gives *NIL.*

We have introduced all the instruments in the LISP orchestra. Now it's time to make some music.

### The Factorial Function

Our first example is the venerable LISP program to compute the factorial function:

$$n! = \begin{cases} 1 \text{ if } n \text{ is } 0 \\ n \times (n-1)! \text{ if } n \neq 0 \end{cases}$$

We want to convert this description into a LISP algorithm. The "if" structure can be converted into a conditional expression, and we can name the new operation *fact.* We assume our LISP machine has such a multiplication operation named *times;* we also assume the existence of a simple subtract-by-one function, *sub1.* Here's the body of a factorial algorithm in M-LISP:

```
[eq[n;0]→1;
 t→times[n;fact[sub1[n]]]]
```

Notice the occurrence of the function name *fact* in the

**The problem will solve itself before we get tired of reducing.**

body; it is the name of the function we are defining, and somehow we must associate that name with the body. We symbolize that association using "<=". For example:

```
fact[n]<=[eq[n;0]→1;
          t→times[n;fact[sub1[n]]]]
```

Here is its pretty-printed translation in S-LISP:

```
(DEF FACT (N) (COND ((EQ N 0) 1)
                    (T (TIMES N (FACT (SUB1 N))))))
```

The new ingredient in these definitions is the use of *recursion*. A typical recursive definition has several characteristics:

- The body of the definition should be a conditional expression. A definition like *foo[x]<= baz[foo[bar[x]]]* will cause nothing but grief. The conditional expression will contain two basic parts: the *termination case* and the *general case(s)*.
- The termination case describes what to do when a primitive data structure is recognized. We consider the integers built from zero, using the successor function, *add1*. Therefore, our termination case in *FACT* involves recognition of *0*, and terminates with value *1*.
- The general cases involve "composite" data structures. We can decompose a positive (composite) integer down to zero by a sequence of subtract-by-one operations. The essential idea is that reducing the complexity of the argument in a recursive call will thereby reduce the complexity of the problem. That's an old trick; what recursion says is that we can solve the original problem by reducing it to a simpler case of the same problem. If we persist, the problem will solve itself before we get tired of reducing; it's like dieting.

Recursive definition is similar to inductive description, like those we gave for defining lists or the M-LISP to S-LISP mapping. The techniques involved in finding the right inductive steps are similar to those involved in finding the right decomposition in a recursive definition. Recursive definition is a powerful descriptive technique; fortunately it can also be implemented as a very efficient computational mechanism.

### Equal

For a further example, assume that we want to test the equality of two lists, where equality means that each element of two lists is identical and the order in which those elements occur is identical. The identity relation also extends to sub-elements of lists. For example:

**equal**
*(A B C) (A B C)*
*(A(B C)D) (A(B C)D)*
*( ) ( )*

**nonequal**
*(A B C) (A B D)*
*(A(B C)D) (A D(B C))*
*(A(B(C)D)) (A B C D)*

Let *EQUAL* be an algorithm to compute this extended equality; it will be recursive. Regardless of the complexity of objects, all we need to do is find the right way to decompose them, and then pounce on the pieces. The decomposition operators we have for lists are *FIRST* and *REST*. We also have to stop the decomposition. In *FACT* we tested for the occurrence of zero; in *EQUAL* we test for the occurrence of an empty list, and since we are assuming that elements of a list may either be sublists or atoms, we need to test for the occurrence of an atom. Let's try the simplest case first, the empty list:

```
(DEF EQUAL (X Y)(COND ((NULL X) ...?)
```

What should we do? If $x$ is empty, then we will only have equality if $y$ is also empty, otherwise we will have an inequality:

```
(DEF EQUAL (X Y)
     (COND ((NULL X)(COND ((NULL Y) T)
                          (T NIL)))
           ...?)
```

Note that we embedded a conditional expression within a conditional expression. Note also that the interior conditional returns either *T* or *NIL*; but that's what we wanted

since *EQUAL* is to encode a **predicate** and *T* and *NIL* are our representations of the truth values *t* and *f*. Note too that we depend on the order dependence of the conditional evaluation; we won't test the *(NULL Y)* expression unless *(NULL X)* is true. We won't get to the *"...?"* condition unless *(NULL X)* is false.

We can still have $x$ non-empty, and $y$ empty; let's take care of that:

```
(DEF EQUAL (X Y)
    (COND ((NULL X)(COND ((NULL Y) T)
                         (T NIL))
          ((NULL Y) NIL)
          ...?)
```

Now the *"...?"* has been reduced to the case that both lists are non-empty, and we can massage the pieces with *FIRST* and *REST*. We look at the *FIRST* pieces; if they're equal, then our decision on the equality of the original lists depends on the equality of the remainders (or *REST*s) of the lists. If the *FIRST*s are not equal, then we can stop immediately with a false indication. This analysis yields two cases: if the first elements are atomic, then use *EQ* to check their equality; otherwise use *EQUAL* itself on the first elements. Here we go:

```
(DEF EQUAL (X Y)
  (COND ((NULL X)(COND ((NULL Y) T)
                       (T NIL))
        ((NULL Y) NIL)
        ((ATOM (FIRST X))
         (COND ((ATOM (FIRST Y))(EQ X Y))
               (T NIL)))
        ((ATOM Y) NIL)
        ((EQUAL (FIRST X)(FIRST Y))
         (EQUAL (REST X)(REST Y)))
        (T NIL))))
```

## Reverse

So far our examples have been either numerical or predicates. Predicates only require traversing existing lists; we will certainly want to write algorithms which build new lists. Consider the problem of writing a LISP algorithm to reverse a list $x$. There is a simple, informal computation: take elements from the front of $x$ and put them onto the front of a new list $y$. Initially, $y$ should be *( )* and the process should terminate when $x$ is empty.

For example, reversal of the list *(A B C)* would produce the sequence:

| $x$ | $y$ |
|---|---|
| (A B C) | ( ) |
| (B C) | (A) |
| (C) | (B A) |
| ( ) | (C B A) |

The *reverse* function will build the new list by concatenating the elements onto the second argument of *rev'*:

```
reverse[x]<=rev'[x;( )]
rev'[x;y]<=[null[x]→y;
            t→rev'[rest[x];
                   concat[first[x];y]]]
```

Since $y$ was initialized to *( )* we are assured that the resulting construct will be a list.

We leave it to the reader to translate this algorithm into S-LISP.

## Summary

Those of you who have already heard about LISP programming know that LISP's two major characteristics are: lots of parentheses, and strange function names like *car*, *cdr*, and *cadadr*. By now you should at least understand *why* the parentheses are used, if not totally understand why the representation is a benefit rather than a curse.

LISP's second characteristic is definitely a blemish. More to the point, it's a commentary on the state of LISP

# NEECO

## PROUDLY ANNOUNCES THE NEWEST PET MICROCOMPUTERS BY COMMODORE!

**PET 2001 — 32K**

**NEW!**

### The PET™ is now a truly sophisticated Business System with the announcement of these Peripherals.

The PET™, incorporated with the Floppy Disk and Printer makes an ideal business system for most professional and specialized fields: medicine, law, dental, research, engineering, toolmaking, printing, education, energy conservation, etc. ... The PET™ Business System as a management tool, delivers information to all levels of Business, previously attainable only with equipment many times more expensive. The PET™ Business System is one of the most cost efficient business tools today. It offers a wide range of applications from logging management strategy in major corporations to organizing accounts and inventory control of small businesses. Here are just a few of the cost-saving uses in the corporation, professional office or small business: stock control, purchasing, forecasting, manufacturing costing, customer records, mailing lists, etc. ... The PET™, Floppy Disk and Printer, a compatible business system at a reasonable price — Take a closer look at these Peripherals.

| PRODUCT | DESCRIPTION | PRICE | AVAILABILITY |
|---|---|---|---|
| PET 2001—4K | 4K RAM | $ 595 | IMMEDIATE |
| PET 2001—8K | 8K RAM | $ 795 | IMMEDIATE |
| PET 2001—16KN (Large Keys) | 16K RAM* | $ 995 | IMMEDIATE |
| PET 2001—32KN (Large Keys) | 32K RAM | $1295 | IMMEDIATE |
| PET 2023 PRINTER | ROLL FEED | $ 850 | IMMEDIATE |
| PET 2022 PRINTER | TRACTOR/ROLL | $ 995 | IMMEDIATE |
| PET 2040A | SINGLE FLOPPY | $ 895 | JUNE/JULY |
| PET 2040 | DUAL FLOPPY* | $1295 | IMMEDIATE |
| PET C2N | 2nd Cassette | $ 100 | IMMEDIATE |

*The 16K/32K (large keyboard) units do not include a cassette drive. Order C2N Cassette. 2040 Floppy Drive requires a 16K or 32K unit. 8K RAM Retrofit available July.

LARGE TYPEWRITER KEYBOARDS NOW AVAILABLE!

ALL UNITS ARE FULLY TESTED BY NEECO BEFORE SHIPMENT. ALL PET'S ARE WARRANTEED (BY NEECO) FOR 1 FULL YEAR! NEECO IS A FULL CUSTOMER-ORIENTED BUSINESS. PLEASE CALL FOR ADDITIONAL INFORMATION. ALL UNITS ARE IN STOCK & READY TO SHIP. FULL SOFTWARE AVAILABLE!

## COMPUTHINK .4 & .8 MEGABYTE DISK DRIVES FOR THE NEW 16/32K PETS!

DISK SYSTEMS INCLUDE DISKMON OPERATING SYSTEM IN ROM AND INTERFACE TO 16/32 PETS!

- Dual Minifloppy Drives with 200K per diskette side for total 400K/800K on line.
- 800K model accesses all 4 diskette sides via dual read and write arm system.
- Dual Density Hardware and DOS loads 20K (with verification) in 4.2 seconds complete.
- DISKMON (DOS) adds 17 commands to BASIC including Random Access and printer support.
- System comes complete with plug in internal board containing 8K RAM, DOS, and Disk Controller Hardware—Board plugs directly onto internal memory expansion pins.
- System does not utilize IEEE or USER Port, system functions directly from memory port.
- All DISKMON DOS commands reside interactively with BASIC—disk directory command and format command do not interfere with program in RAM—DOS command were designed for simplicity of use. System was manufactured for heavy commercial use.
- System installs completely in less than ten minutes—immediately ready for use.
- $1295 and $1595 prices include all hardware, DOS, complete user manual, and demo/utility diskette.
- Available software includes PLM Compiler ($250), Relocatable Assembler ($70), Source-Editor Program ($70), Autolink Linking Loader ($70), and a complete Database system (Pagemate $495).
- Call or write for complete product information and specifications—User manual $10.

FOR 8K PETS (small keyboards) .4 Megabytes of Disk Storage for 8K PETS! (Requires Expandamem)
400K-8S DISK SYSTEM INCLUDES RANDOM ACCESS IN DOS-LOADS 20K IN 4 SECONDS!
24K Expandamem Memory $525
32K Expandamem Memory $615
**$1295**

(PRODUCT AVAILABILITY IS AUG/SEPT—CALL FOR INFO)

ALL 16/32K MODELS INCLUDE AN INTERNAL PLUG-IN INTERFACE BOARD CONTAINING DOS, 8K OF RAM, AND CONTROLLER

400K-16N ....... $1295
400K-32N ....... $1295
800K-16N ....... $1595
800K-32N ....... $1595

**CALL OR WRITE FOR A FREE COPY OF OUR NEW JULY CATALOG!**

## NEW CENTRONICS 730 PRINTER FOR PET!

• NEWEST TECHNOLOGY FROM CENTRONICS • 50 CPS • 80 CHARACTER LINE • 10 CPI • 7x7 DOT MATRIX • HANDLES ROLL FEED, PIN FEED PAPER • UPPER & LOWER CASE—$1099 PRICE INCLUDES INTERFACE TO IEEEPORT. AVAILABLE AUG/SEPT.

**$1099**

TRS-80 USERS!-THE MODEL 730 IS AVAILABLE FOR USE WITH THE TRS-80. PRICE INCLUDES CABLE only **$995**

WE CANNOT LIST ALL OF OUR SOFTWARE AND HARDWARE PRODUCTS
CALL OR WRITE FOR OUR *FREE* SOFTWARE/HARDWARE DIRECTORY

ALL NEECO PETS CARRY A FULL ONE-YEAR NEECO WARRANTEE.

NEECO

NEW ENGLAND ELECTRONICS CO., INC.
679 HIGHLAND AVE., NEEDHAM, MASS. 02194
MON. - FRI. 9:30 - 5:30, EST.

**(617) 449-1760**
MASTERCHARGE OR VISA ACCEPTED
TELEX NUMBER 951021, NEECO

*programming,* rather than the language. When we examine the very low level representation of LISP operations, we see that the primitive selection operations of LISP data structure can be described as selecting either the left or right branch of a binary graph. *Car* and *cdr* are these selection functions, and *cadadr* is an abbreviation for a composition of these operations. Since all LISP data structures (in our simple subset, remember) must ultimately be representable as combinations of atoms and binary graphs, then all algorithms must ultimately be expressible as manipulations of graph structure involving *car, cdr,* and a function to construct new graphs, *cons.*

Most LISP programs are constructed in just such a fashion. The result is unsatisfactory from at least two views. First, the programs become almost totally unreadable. Instead of couching the data structure abstractly in terms of the *concept* ,recognizer: *is_dog[x];* selectors: *left_eye[x], tail[x],...;* and constructor(s): *make_dog[*$x_1$*;...*$x_n$*]—,* the programmer performs the transformation *mentally* and gives us *eq[cadr[x]; DOG],cadaddr[x],* and *cons[x; cons[z;y]...],* which borders on gibberish. Neither the programmer nor a reader has much chance of remembering what is going on.

An equally serious problem is that this style of programming deeply intertwines conception and implementation. Given that a new representation of "dog-ness" is required, the programmer must search out all areas of program which use the arcane encoding and replace them *very carefully.*

Essentially there are two solutions to this problem. One solution is to require the programmer to spell out detailed rules for data structuring *a la* Pascal. Of course there's no reason to suppose that the programmer's ability to remain abstract will survive any better here. Indeed since Pascal really supplies "abstract *storage* structures" rather than "abstract *data* structures," along with the requisite verbiage of a typed language, there are reasons to believe that the programming process will suffer in the long run. The alternative is to supply the programmers with an exceptional programming tool and an understanding of abstraction, modularity and the power of their tool. It may be naive to believe that programmers can be self-disciplined, but the alternatives are not at all attractive.

The other LISP articles in this issue explore detailed examples of LISP applications. Throughout these articles a recurrent theme is the delicate balance between realistic abstraction and overspecification. One of the real wonders of LISP is that it allows you to work with ideas.

Traditionally, all LISP implementation problems have been dealt with in software. An exciting alternative is to build LISP machines in hardware, thereby raising the programming floor to a much more acceptable machine level than previously available. Several very healthy projects exist, from re-microcoded machines, through specially constructed hardware, to experiments with very large scale integration LISP devices. For those readers who are interested in more details, several of these efforts will be documented in an issue of the IEEE Transaction on Computers later in 1979. It is clear to me that LISP is only beginning to have an impact upon the computing community. ■

# MINIMAX. THE COMPUTER TH

No microcomputer in the world can match Minimax incredible feature for incredible feature.

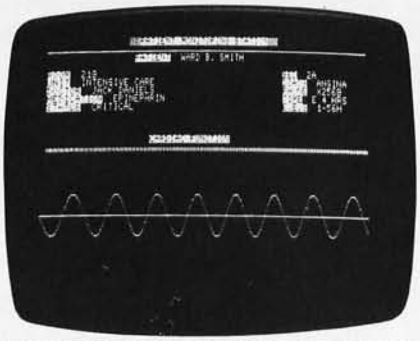
Simply stated, Minimax is the most advanced, least expensive computer ever created.

**NEVER BEFORE HAVE THESE FEATURES BEEN AVAILABLE ON A MICROCOMPUTER AT ANY PRICE – LET ALONE $4495.**

**More Memory.** Minimax comes in two sizes. Both with 108,544 bytes of semiconductor internal memory.

In addition, the Minimax I has an external storage capacity of 800k while the Minimax II features an enormous 2.4 megabytes of on-line disk storage. It's the most memory for the least money.

**More Speed.** Minimax's speedy 15,000 character per second data transfer rate is substantially faster than any other microcomputer.



**Split screen combination of character and graphic information.**

**More Graphic Resolution.** Minimax's screen has the highest resolution graphics of any microcomputer. 122,880 (240x512) individual points, each program addressable.

**More Screen Capabilities.** Minimax offers the most advanced full screen data entry and editing capabilities of any microcomputer. They include word or character insertion or deletion, split screen modes, individual field editing and automatic skip to next field.

**More Programability.** Minimax features exceptionally advanced software development tools. The CPU comes complete with 64 user definable opcodes and ROM resident software. Plus Microsoft BASIC, PLM, DOS operating system, FIFTH language interpreter, a machine language monitor with Tiny Assembler, Disassembler, Dump Facility and Debugging Aid.

**More Applications Software.** Minimax's superior software selection allows you to perform most of the common data manipulation, accounts receivable, accounts payable, general ledger, inventory, payroll and software development functions with-

Circle 71 on inquiry card.

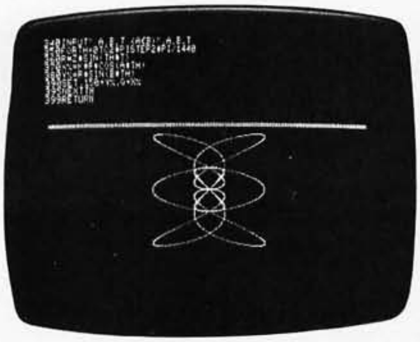
# AT OUTCOMPUTERS THEM ALL.

out expensive or time consuming programming.

### YOU GET 2.4 MEGABYTES WITHOUT TAKING A BIG BITE OUT OF YOUR BUDGET.

At $4,495 for the 800k Minimax I and $5,995 for the 2.4M Minimax II, it is not only incomparable feature for feature, but dollar for dollar as well.

While these prices do not include a printer, a wide range of character, high speed and word processing printers are available.



**High resolution graphics for original design development.**

### THERE'S MORE INSIDE A SINGLE MINIMAX THAN MOST COMPUTERS TWICE ITS SIZE AND FOUR TIMES ITS PRICE.

Minimax's remarkable sophistication also makes it remarkably versatile for business, scientific or engineering applications.

For business, everything from payroll checks to accounting ledgers can be instantly processed, printed and filed.

Engineers, architects and scientists can use the high resolution graphics as a visual research and planning tool.

Because of its comprehensive software capabilities, software engineers can develop finished programs for other microcomputers or the most sophisticated mainframes.

### PEOPLE CALL IT INCREDIBLE. WE CALL IT MINIMAX.

There has never been so much capacity, flexibility and sophistication built into a computer so affordable.

To obtain the complete Minimax story — including ordering information — please send the reader service card or contact us directly.


**Complete business software packages.**

The more you know about Minimax, the more incredible it becomes.

### WHAT MAKES ONE COMPUTER BETTER THAN ANOTHER MAKES MINIMAX THE BEST.

# MINIMAX™

**Developed and Manufactured by COMPU/THINK™ 3260 Alpine Road, Menlo Park, CA 94025 (415) 854-2577**

## Technical Forum

# Permutation Bibliography

Eduardo Kellerman
IBM
Endicott NY 13760

In the article "Solving the Eight Queens Problem" (October 1978 BYTE, page 122) Terry Smith asked readers for information on algorithms for generating permutations. In April 1975, I compiled the following bibliography on the subject (I have not updated it since then). I think some readers may find it useful.

### I Generation of Permutations

Mark B Wells, "Generation of Permutations by Transposition," *Mathematics of Computation*, volume 15, 1961.

Frank Harary, "Permutations with Restricted Position," *Mathematics of Computation*, volume 16, 1962.

J R Howell, "Generation of Permutations by Addition," *Mathematics of Computation*, volume 16, 1962.

Selmer M Johnson, "Generation of Permutations by Adjacent Transposition," *Mathematics of Computation*, volume XVII, number 83, July 1963.

D H Lehmer, "The Machine Tools of Combinatorics" in *Applied Combinatorial Mathematics*, edited by E F Beckenbach, John Wiley and Sons Inc, New York.

G G Langdon Jr, "An Algorithm for Generating Permutations," *Communications of the ACM*, volume 10, number 5, May 1967.

M Renaud and S Regnier, "Programme de Permutations," *Revue Francaise d' Informatique et de Recherche Operationell*, May-June 1967.

D Pager, "A Number System for the Permutations," *Communications of the ACM*, volume 13, number 3, March 1970.

E Kellerman, "Method for Generating Permutations," *IBM Technical Disclosure Bulletin,* volume 13, number 8, January 1971.

Kazuaki Harada, "Generation of Rosary Permutations Expressed in Hamiltonian Circuits," *Communications of the ACM,* volume 14, number 6, June 1971.

B R Heap, "Permutations by Interchanges," source unknown.

D J Lorch, "Permutations of N out of M," personal communication, July 18 1972.

Ronald C Read, "A Note on the Generation of Rosary Permutations," *Communications of the ACM,* volume 15, number 8, August 1972.

C T Fike, "A Permutation Generation Method," *IBM Technical Report TR73.002,* Systems Research Institute, New York.

Mohit Kumar Roy, "Reflection-Free Permutations, Rosary Permutations, and Adjacent Transposition Algorithms," *Communications of the ACM,* volume 16, number 5, May 1973.

R A Davis, "Permutation of Bits in a Bit String," *IBM Technical Disclosure Bulletin,* volume 16, number 5, October 1973.

E Kellerman and D J Lorch, "Generation of Permutations and an APL Implementation," *IBM Technical Disclosure Bulletin,* volume 15, number 5, October 1972.

M M Halpern, "Permutations," Proceedings of the Fifth International APL Users Conference, May 15 thru 18 1973 (Canadian Printco Limited).

B M Zlotnik, "An Algorithm of Permutation Enumeration," *Avtomatika i Vychislitel'naya Teknika,* number 2, 62, 1972.

S Even, *Algorithmic Combinatorics,* Macmillan, New York, 1973, pages 2 thru 11.

C K Wong and D Coppersmith, "The Generation of Permutations in Magnetic Bubble Memories," *IBM Technical Report RC5174,* IBM Research, December 1974, Yorktown Heights, New York.

B M Zlotnik and V S Kogan, "A Method of Transforming Permutation n!-Codes," *Avtomatika i Telemekhanika,* number 1, January 1975, pages 139 thru 142.

E W Stacy, "Exact Evaluation of Determinants Via Permutation Arrays," *IBM Technical Disclosure Bulletin,* volume 18, number 9, February 1976.

F M Ives, "Permutation Enumeration: Four New Permutation Algorithms," *Communications of the ACM,* volume 19, number 2, February 1976.

Nachum Dershowitz, "A Simplified Loop-free Algorithm for Generating Permutations," *BIT 15* 1975, pages 158 thru 164.

*Algorithms in the Communications of the ACM:* 71, 86, 87, 102, 115, 202, 235, 242, 250, 306, 307, 308, 317, 323, 362, 383.

*Algorithms in the Computer Journal:* 6, 27, 28, 30.

*Technical Forum is a feature intended as an interactive dialog on the technology of personal computing. The subject matter is open-ended, and the intent is to foster discussion and communication among readers of BYTE. We ask that all correspondents supply their full names and addresses to be printed with their commentaries. We also ask that correspondents supply their telephone numbers.*

# Technical Forum

## TI Has Faster Solutions

*Marvin A Larson, 345 Birchwood Dr, Moraga CA 94556*

Before reading Mr Arp's article, "The Power of the HP-67 Programmable Calculator, Part 2" (April 1979 BYTE, page 176), I was under the impression that the Hewlett-Packard HP-67 and the Texas Instruments TI59 programmable calculators were about equal in function, utility, and calculating power. Both are "top of the line" although the HP-67 costs about 70% more than the TI59.

The procedures used by Mr Arp in writing his simultaneous equations program can be applied, with minor reprogramming, to the TI59. The resulting program would then be capable of solving 29 simultaneous equations in 29 unknowns, as opposed to 9 equations in 9 unknowns with the HP-67.

The TI59 can use up to 100 data storage registers, compared to 26 registers for the HP-67. It can read/write data from/to magnetic cards in banks of 30 values. Each card can thus contain the 29 coefficients and one constant term for one complete row of the solution array.

The Library Module supplied with the TI59 contains a program for solving simultaneous equations which will solve up to 8 equations with 8 unknowns, as compared to 4 equations with 4 unknowns for the HP-67.

Mr Arp did not tell us how much time is required to solve the set of 9 equations given in his listing 4 (page 186), or the resultant accuracy of the solution. It appears to involve one hundred or more read/write operations from/to magnetic cards, a considerable amount of external manual bookkeeping to keep track of the cards, hand copying of coefficients, and the like. My guess is that solution time is about 90 minutes, provided the wrong card does not slip in. With regards to accuracy, Mr Arp gives his solution results with 6 digit values, but does not state the closure error on back substitution in the original equations.

For comparison, I tried the library program in the TI59. To reduce the problem to eight equations instead of nine, I deleted cell 9 in figure 1 (page 180). This has the effect of deleting the ninth coefficient of the first eight equations and the entire ninth equation of table 1 (page 180).

This was my first experience with using the TI59 to solve simultaneous equations, so I read the instructions carefully. Then I timed the operation. From the beginning at the start of data entry, to the end after all eight unknowns had been copied down, the procedure took just 13 minutes.

All answers came out as 10 digit numbers. On back substitution all equations closed out with a maximum error of 4.6E−9 and a mean absolute error of 2.2E−9. Most of the functions and operations on Mr Arp's "wish list" are already available on the TI59. He would be well advised to check out the TI59.

Incidentally, Texas Instruments software isn't always quite as good as its hardware. The TI59 has sufficient computing capacity to solve 10 simultaneous equations in 10 unknowns with the program entered from magnetic cards, and 11 equations in 11 unknowns with the program resident in a library module. This is with a full set of equations with non-zero values for all coefficients.

## 6809 Commentaries, Continued... Don't Be So Superficial!

*Jim Howell, 5472 Playa Del Rey, San Jose CA 95123*

I would like to correct some statements made by David Kemp concerning the 6809 microprocessor in "Compare New Microprocessors Carefully" (Technical Forum, May 1979 BYTE, page 213).

The 6809 has several more 16 bit instructions than those mentioned by Mr Kemp (ADDD, SUBD, and CMPD). The CMPX, CMPY, CMPS, and CMPU instructions compare the X, Y, S, or U register with (up to) 16 bits of data. The ABX instruction adds B (8 bits, unsigned) to X (16 bits) putting the 16 bit result into X.

The major 16 bit arithmetic instruction of the 6809, however, is the Load Effective Address instruction. This instruction is actually four instructions: LEAX, LEAY, LEAS, and LEAU, depending on which register gets the result of the arithmetic. This instruction computes an address in the same way as the indexed addressing mode, but puts the resulting address into a register (X, Y, S, or U). Load Effective Address adds any one of the registers X, Y, S, U, or PC to any of the following: a signed immediate value (5, 8, or 16 bits), the sign-extended A or B register, or the D register (A and B together as a 16 bit register), and puts the result in any of X, Y, S, or U — not necessarily the same as the source register. The PC (program center) can actually be the destination for such a calculation using the branch instruction with the indexed addressing mode. I think Mr Kemp is exaggerating when he states that the user pays "heavily" for the generality of being able to transfer (or exchange) any register with any (like-sized) register. The designers of the 6809 included instructions to transfer and exchange between any pair of the four 8 bit registers A, B, DP (direct page), and CC (condition code), and between any pair of the six 16 bit registers X, Y, S, U, D, and PC. Excluding transfers or exchanges of a register with itself, this gives 42 different transfers and 21 different exchanges. (TFR A,B and TFR B,A are different but EXG A,B and EXG B,A are the same.) Each of these is a 2 byte instruction, the first byte specifying transfer or exchange, and the second byte specifying those registers which are involved. It would have been possible to provide a (small) subset of these transfers and exchanges as 1 byte opcodes at the expense of making some other instructions longer. Transfers and exchanges not provided for in this scheme would take at least two instructions and two bytes (probably three of each for exchange) and would operate more slowly than the 2 byte transfer or exchange. If some transfers and exchanges are allowed and others are not, the assembly language programmer also has to remember which ones these are. Either scheme of register transfers and exchanges would have been possible, but since these instructions are not that common in programs (falling into the "11.3% other" category), I think the designers of the 6809 made the better choice.

I cannot comment much on the 6516 mentioned by Mr Kemp, since my knowledge of that processor is limited to what he wrote in his letter. (Are you sure that's an 8 bit processor?) The comparison of number of cycles, used in the letter, is valid only if the cycle times of the two processors are the same (or are related in a known ratio). In any event, comparing cycle times of some isolated instructions does not necessarily indicate the relative speeds of the two processors on real programs. The 6516 may have 16 bit AND, OR, and XOR instructions, but how often would these be used? As for Mr Kemp's comment that the 6809 "costs more" (more than other 8 bit processors?) because it uses a larger piece of silicon and has more logic gates than other 8 bit processors, how much will a $20 difference in microprocessor cost make in the final product cost? Besides, doesn't the 6516 "suffer" from this same cost problem?

# 6809 Commentaries, continued

*Richard F Serge, 655 Lewisville-Vienna Road, Lewisville NC 27023*

Never, until now, have I been compelled to respond to any magazine article I have read. I refer to David Kemp's commentary "Compare New Processors Carefully" (May 1979 BYTE, page 213).

As a designer of microprocessor systems I have followed the instructions in the title of Mr Kemp's article with great care. In comparing the 6809 with other processors in its performance range, it may take an hour or so of comparing data sheets to get a feel for the typical hardware required, the addressing modes available, the relative execution times, and the number of bytes required for the more common instructions. To stop at this point and decide which is "best" is the equivalent of flipping a coin. At this point several passes through the programming manuals are required, along with a study of any other literature pertaining to the processors in question.

Only after a designer has a thorough understanding of the processors' instruction set and addressing modes, and how to efficiently utilize these features, can the task of careful comparison begin. Recalling past design projects and how they could have been implemented on the processors being compared is an excellent way to make a fair comparison (not just one or two projects, but several). The real test is laying out $30 K for a couple of development systems and actually doing it, but....

The procedure which I have outlined is more of a study than a comparison. It takes a long time, and a concentrated effort to be fair right up to the end.

Although I disagree with most of Mr Kemp's article, I take special issue with the light regard he appears to have concerning the multitude of various addressing modes offered by the '09. The difference between having and not having just one of these modes can very easily alter the entire design of a software package, making the execution times of even most instructions seem like trivia compared to what can be saved. Being able to write recursive, position independent code with the '09 should also weigh heavily in any comparison being attempted with the '09.

There is another point I would like to clarify. Mr Kemp states that "many 6809 instructions require 4 bytes to specify." Many readers may have gone away thinking "most," rather than "a few," since no further explanation followed. Motorola says that they chose these 4 byte instructions as some of the lesser used op codes, and I find that these 4 byte instructions occur about once per page of assembly listing (typically 50 lines of code). The vast majority are 2 bytes.

I have been designing with the 6809 (a real part) since mid-March 1979. The reason: it is the most powerful 8 bit MOS microprocessor. And I do not work for Motorola. ■

# Get Ready to Make Your Own Magic

ınouncement I. The first eight Personal ograms® from Aladdin Automation are aiting for you now at your neighborhood ımputer retailer or direct from Aladdin.

ɔw you can get your full share of Aladdin agic in every one of these Personal ograms®:

**Math-Ter-Mind®** A delightful, educational learning experience for your pre-school child. Watch e smile on your child's face as a correct ıswer makes the mathematician smile on the reen before you. A nursery song also serves a reward for learning elementary addition ıd subtraction. With Aladdin's Math-Ter-ind® your child's pathway to learning will be n-filled . . . for both of you. Math-Ter-Mind®, ıe first release from the Aladdin Education® eries. (nursery song currently available only ı Apple II® program)

**Lunar Lander** In a controlled descent, you're just seconds away from your first landing on the cold, rbidding surface of the moon. As you ıvigate your delicate spacecraft downward to e safety of Moonbase, you must be ever atchful of the dangers rising to meet you with ıch passing moment: a fuel level fast pproaching zero; deadly meteor showers that ɔme from any direction, at any time; sheer-ced rock cliffs and rough terrain; choosing e correct landing pattern and rate of descent. addin's Lunar Lander. Your chance to reach ıt and touch the stars . . . without leaving the fety and comfort of your own chair. The first lease from the Aladdin Simulation® Series.

**Craps** All eyes in the casino are on you. The dice are in your hands. Lady Luck sits at your shoulder, whispering . . . "Just one more time. Try your luck just one more time." You throw . . . and watch the dice tumbling on the screen. With Aladdin's Craps you play against the computer, so it's awfully tough to win. But when you do, it's an experience you're likely never to forget. Craps. An exciting, heart-pounding Personal Program®. The first release from the Aladdin Las Vegas® Series.

**Mastermind** A challenging game of intrigue, centuries old, that will give you full chance to test your powers of logic, deduction and reason. And test them you will, as you try and solve the computer's puzzle, using clues as they're provided one-by-one. You control the degree of difficulty in this classic Personal Program® that offers one simple, yet all-consuming challenge: beat the Mastermind in a direct, one-on-one battle of wits. Aladdin's Mastermind. The first release from the Aladdin Old Favorites® Series.

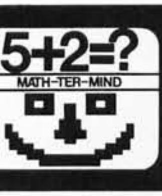
**Tic-Tac-Toe** Five different levels of difficulty allow a person of any age or skill to take part in this relaxing, enjoyable game that can act as a learning tool, as well. Level I, for example, is suitable for children and is excellent also for teaching simple mathematics. The computer plays just about perfectly at Level V. Just about, that is, so go ahead and take your best shot. See if you can beat the computer in this traditional favorite of young and old alike. Tic-Tac-Toe. Another first release from the Aladdin Old Favorites® Series.

**Jungle Island®** Shipwrecked in a raging storm at sea, miraculously you survive only to find yourself stranded on a seemingly deserted jungle island. Without food, water or supplies of any kind, you begin to try and find your way to safety. The computer will be your eyes and ears as you explore your jungle island and all the mysteries and dangers that lie in wait for you. Jungle Island®. A captivating first release from the Aladdin Adventure® Series.

**Stix®** Aladdin's Stix® can be played with 2 to 5 piles of sticks and between 1 and 19 sticks in each pile. The object: to be the one to pick up the last stick. Sounds simple? Yes, but you're playing against the computer. Take heart, though, because you can control the degree of difficulty in this update of the ancient game of Nim. Stix®. Another first release from the Aladdin Old Favorites® Series.

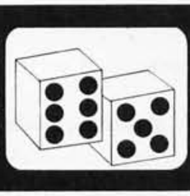
**Super Pro Football®** Here's your chance to be more than just an armchair quarterback. With Aladdin's Super Pro Football® you can replay any Super Bowl game, from the first, between Green Bay and Oakland, to last year's classic victory by Pittsburgh over Dallas. For once you can turn back the clock and go for that one big play that made the difference between victory and defeat in pro football's biggest game of all. Super Pro Football®. The first exciting release from the Aladdin Super Pro® Series.

Visit your neighborhood computer retailer or contact Aladdin direct to get your full share of the magic in Announcement I, the first eight Personal Programs® from Aladdin Automation.

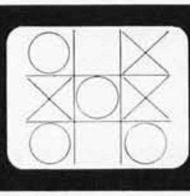


ath-Ter-Mind®

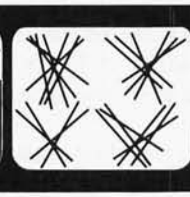

Lunar Lander


Craps


Mastermind


Tic-Tac-Toe


Jungle Island®


Stix®


Super Pro Football®

**/elcome to the All-New World of laddin. And Get Ready to lake Your Own Magic**


ALADDIN AUTOMATION, INC.
A DIVISION OF
ALADDIN COMPUTER CORP.
3420 Kenyon Street, Ste. 131, San Diego, CA 92110




Design and copy by Campbell Marsh Graphic Communications

Circle 3 on inquiry card.

# The Design of an M6800 LISP Interpreter

S Tucker Taft
Harvard University Science Center
1 Oxford St
Cambridge MA 02138

## The primary data structure is the list.

Anyone exposed to small computer systems has used a language interpreter of some sort, and certainly may have thought about implementing their own interpreter. Unhappily, implementing an interpreter for a complete version of most computer languages is a difficult and time-consuming job, unsuitable for a part-time personal computer enthusiast. The language LISP provides a unique opportunity in this respect. The foundation for a very complete interpreter can be programmed by a single person in several months of part-time effort. As a bonus, the resulting interpreter provides the user with a high level language in which to express algorithms.

### The Language

From the user's point of view, the primary data structure in LISP is the *list*. Every element of a list is either an *atom* or another list. An atom is a primitive named object, the name being an arbitrary string of characters:

ABC is an atom.
135 is an atom.
(ABC 135) is a list of two elements, both atoms.
((ABC 135) XYZ) is a list of two elements, the first of which is a list, the second is an atom.
( () () ) is a list of two elements, both being lists of zero elements. A list of zero elements, the *null* list, is identified with the atom NIL.

The feature of the language LISP which makes it at the same time a uniquely interesting language, and relatively easy to implement, is that all program elements are represented using these same kinds of objects: atoms and list. Constants, variables, expressions, conditionals, even function definitions are all represented using only atoms and lists.

A value is associated with each atom, allowing atoms to represent program variables and constants. A symbolic atom, like XYZ, would represent a variable. A numeric atom, like 237, would represent a constant.

Operations on variables and constants, like addition, or a function call, are represented by list expressions:

(ADD 2 5) would represent the expression 2 + 5.
(SIN (MUL 2 Y)) would represent the expression sin(2y).

Conditionals, loops, and function definitions are also represented by list expressions, as illustrated by this recursive function implementing Euclid's greatest common divisor algorithm:

```
(DEF GCD (LAMBDA (X Y)
   (COND
      ((GREATER X Y) (GCD (SUB X Y) Y))
      ((GREATER Y X) (GCD X (SUB Y X)))
      (T X)
   )
))
```

This would be equivalent to the Pascal program:

**function** *gcd(x,y:***integer***)*:**integer**
  **begin**
    **if** $x>y$ **then** $gcd := gcd(x-y, y)$
  **else**
    **if** $y>x$ **then** $gcd := gcd(x, y-x)$
  **else**
    $gcd := x$
  **end.**

An important difference to note in the above comparison is that no explicit assignment to a function return value is made in LISP, whereas in Pascal one must explicitly say *gcd* := ... to specify the return value. In Pascal, and most other *procedural* languages, a distinction is made between program statements and expressions. In such languages some program statement must be

**About the Author**

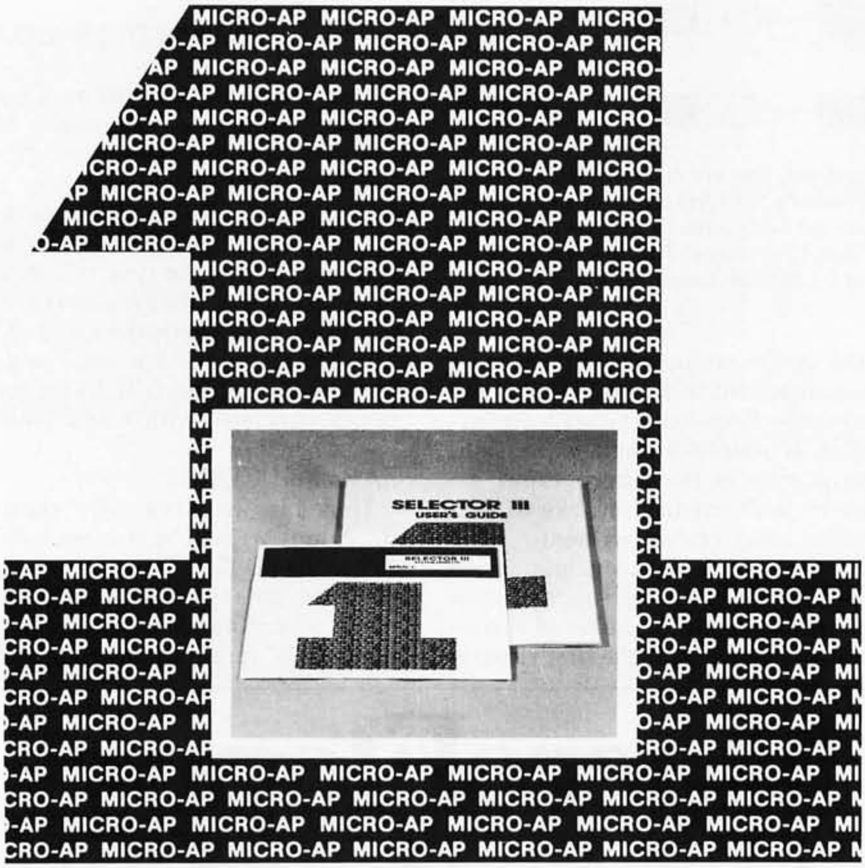
*Tucker Taft first programmed a computer in 9th grade. He spent the following summers at various programming jobs until he graduated fromHarvard in 1975 with a degree in chemistry. Since his graduation, Tucker has spent two years as the full-time systems programmer for Harvard's Student Timesharing System, combined with teaching some introductory computer courses at Harvard.*

*Tucker is now starting a microcomputer software consulting business based on a multilanguage compiler being written in LISP. In what is left of his free time, he is found on a squash or tennis court, in a Cambridge coffee shop, in a bookstore, or in a Chinese restaurant.*

# ONE PACKAGE DOES IT ALL

## Includes these Application Programs . . .

**Sales Activity, Inventory, Payables, Receivables, Check/Expense Register, Library Functions, Mailing Labels, Appointments, Client/Patient Records**




## RANDOM, MULTI-KEY RECORD RETRIEVAL under CP/M, CDOS, IMDOS, ADOS . . .

**SELECTOR III ALLOWS INSTANT RECALL OF ANY RECORD USING ANY INFORMATION ITEM IN THE RECORD.** That statement deserves re-reading, because that ability makes SELECTOR III the most powerful Date Base Management System in microcomputers today!

With **SELECTOR III** you can...

- define a record format, assign retrieval keys, and begin entering data in minutes.
- create sorted pointers to records matching your specif or range of requirements.
- automatically generate reports with control-break summaries and unlimited variety.
- bring an application online in hours instead of months.

**SELECTOR III** comes complete with eight application programs that perform the tasks listed at top of page. And, since it's distributed in source code form, you can easily add subroutines to do specific computations or file updates.

**SELECTOR III** runs under CBASIC Vers. 1 or 2, and is priced at $295. SELECTOR III-C2 is dedicated to Vers. 2 only, runs about twice as fast, and costs $345.

Both systems are available in a variety of CP/M, diskette size and density formats including IBM 8"; North Star; Micropolis; TRS-80; Processor Tech Helios II; Altair; iCOM; Dynabyte; Imsai; and others.

Available from computer stores nationwide:

**LIFEBOAT Associates**
2248 Broadway, Suite 34,
New York, N.Y. 10024 • (212) 580-0082

Or order direct from

**MICRO-AP**
9807 Davona Drive, San Ramon, CA 94583
(415) 828-6697

Circle 213 on inquiry card.

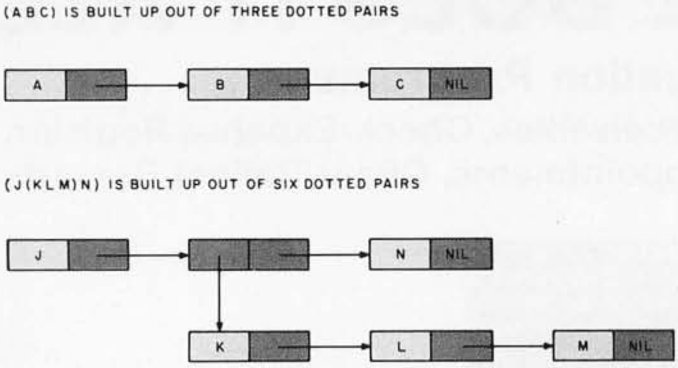



*Figure 1: In most LISP systems, lists are built up out of dotted pairs which are two address cells. The left cell points to the first element of a list, and the right cell points to the rest of the list. The letters in the figure stand for atoms. NIL is a special atom used to signify the end of a chain of dotted pairs.*

executed to specify the return value, usually either a return statement or an assignment to the function name. In LISP, and other *applicative* languages, no such distinction is made. A function is simply a single expression, whose value is the return value of the subprogram.

This is made possible by built-in functions like COND used above. COND takes a list of two element lists as argument. It goes down the list of pairs, evaluating the first element of each pair. If the result is true (the atom T), the result of the entire COND is the value of the second element of the pair. If the value of the first element of the pair is false (the atom NIL), COND proceeds to the next pair. If COND reaches the end of the list, the result of the entire COND is simply NIL. In the above example this would never happen because the first element of the last pair is the atom T (whose value is always guaranteed to be itself, the atom T). This is the normal technique in LISP for using the COND function.

The expression:

```
(DEF GCD (LAMBDA (X Y)...
```

defines the atom GCD to be a function (or *lambda expression*) taking two arguments, to be called X and Y in the body of the definition. Notice that no explicit specification of the type of X or Y is provided. In LISP any arbitrary value, atom, or list may be the value associated with an atom. In this sense LISP is a typeless language. In fact the type of a *value* (ie: whether it is an atom or a list) is always determinable at execution time. Functions must check the types of the values of atoms if only certain types are legal arguments. In the above example the calls on GREATER and SUB would fail if the values associated with X and Y were not numeric atoms.

### CARs and CDRs

Thus far we have only shown how to re-express algorithms written in a more conventional language, in the language LISP. The real power of LISP comes from its ability to directly manipulate lists, a data type not normally accessible in other languages. Three primitives, CAR, CDR (pronounced could-er), and CONS are provided for list manipulation. The function CAR takes a list as argument, and returns the first element of the list, which may either be an atom or another list. The function CDR takes a list as argument, and returns the tail of the list, that is, all but the first element of the orginal list, as a new list. The function CONS takes two arguments, a new first element, and the tail of a list, and reconstructs a list, now one element longer. For example:

Assume the atom X is associated with the value:
  (A B C)
Assume the atom Y is associated with the value:
  (THE CAT IN THE HAT)
(CAR X) would be the atom A.
(CDR Y) would be the list (CAT IN THE HAT).
(CONS (CAR X) (CDR Y)) would be the list:
  (A CAT IN THE HAT)
(CAR (CDR X)) would be the atom B.

In general the CAR of the CDR of a list is its second element, and a function called CADR is frequently defined as a kind of shorthand for CAR of the CDR.

You might wonder what would result if you gave two atoms as arguments to CONS, rather than an atom and a list. In most LISP systems this is in fact legal. The result reveals the underlying representation used for lists in LISP. In virtually all LISP systems, lists are built up out of *dotted pairs,* two-address cells, the left cell pointing to the first element of a list, and the right cell pointing to the rest of the list. This can be diagrammed schematically as in figure 1.

Because dotted pairs are used this way to build up lists, it is natural to call the left cell of a dotted pair the CAR and the right cell the CDR. (In fact the genealogy of the words CAR and CDR runs the other way. Dotted pairs were used in the initial implementation of LISP, and CAR and CDR referred to the address field and the decrement field of a word on the IBM 704.) Now you can perhaps guess that when you pass two atoms as arguments to CONS, you simply get a dotted pair with an atom in both the CAR and CDR. For example:

| | |
|---|---|
| A | B |

would be printed as:

(A . B)

The notation (A . B) is used whenever the CDR of the last dotted pair forming a linked list is a non-NIL atom. In general (D E F . NIL) would be equivalent to (D E F), whereas (D E F . G) could not be expressed without the *dot* notation.

Given the three primitives CAR, CDR, and CONS, and understanding the underlying representation of lists using dotted pairs, it is possible to write powerful list-manipulating programs in LISP. For example, suppose it is desirable to edit a large data structure, and change all occurrences of the symbol APPLE to ORANGE. In LISP we could easily write a routine called REPLACE which, given the data structure (ie: list structure), the original symbol (the atom APPLE), and the replacement symbol (the atom ORANGE), would go through the structure and do the replacement, using itself recursively to do the replacement in all sublists of the list structure:

```
(DEF REPLACE (LAMBDA (STRUC OLD NEW)
   (COND
      ((EQ STRUC OLD) NEW)
      ((ATOM STRUC) STRUC)
      (T (CONS
            (REPLACE (CAR STRUC) OLD NEW)
            (REPLACE (CDR STRUC) OLD NEW)
      ))
   )
))
```

Notice how the first two lines of the COND allow for the possibility that the input data structure is simply an atom (which may or may not be equal to the atom to be replaced). In addition, notice that the entire body of this function definition is a single COND, just as it was in the GCD example given above. This is frequently true in LISP programs. Finally, notice how the function simply *passes the buck* to recursive calls on itself if the STRUC argument is not an atom, CONSing together the results of the two inner calls. The reader is encouraged to go through an example of the execution of this function when the argument OLD is the atom APPLE, the argument NEW is the atom ORANGE, and the argument STRUC is the list structure:

(AN (APPLE A DAY) KEEPS (THE (APPLE MAN) BUSY))

The result should be:

(AN (ORANGE A DAY) KEEPS (THE (ORANGE MAN) BUSY))

If STRUC were:

(PEAR BANANA . APPLE)

the result should be:

(PEAR BANANA . ORANGE)

Other kinds of list-manipulating programs which are relatively easy to write in LISP, but very difficult in more conventional languages, include formula manipulation programs which might take in the list representation for a function (eg: (SIN (MUL 2 X)) ), and return the list representation for its derivative according to the rules of the calculus (eg: (MUL 2 (COS (MUL 2 X))) ).

The author's system is being used for the development of a compiler/interpreter system which generates the list representation for a program written in a programming language, and then either interprets it directly, or generates the list of machine language statements to implement the program on a particular microcomputer. LISP makes such an undertaking quite straightforward (although not trivial, unfortunately!).

### LISP Interpreter

Because programs are data objects (list structures) in LISP, the same routines used to read and print data objects may be used to read and print programs. Furthermore user functions, like a general list editor, can be used also to edit programs. This uniformity vastly simplifies the task of writing an interpreter for LISP. Only three basic modules need be produced: READ, EVAL, and PRINT . READ accepts a LISP list expression from the terminal, in full parenthesized notation, and builds the internal representation of the list, sometimes called a *forum.* EVAL takes a form as its single argument, and evaluates the form according to the LISP convention that the first element of such a list specifies the function, with the rest of the list as arguments.

The result of EVAL is another form. (The term *form* is sometimes reserved for LISP expressions which are legal input to EVAL. The term *S-expression* covers all types of lists, whether or not the first element is a legal function name. Within this paper, form will be used to refer to the internal representation of any type of LISP expression.)

PRINT takes a form as its argument, and types it on the terminal in fully parenthesized form. The top level loop of the LISP interpreter simply prompts the user for input (−> is the LISP prompt), READs in the users input, EVALs the resulting form, and PRINTs the result of EVAL. In a conventional high level language syntax, this would be:

**while** *true* **do begin**
   *patom("−>");*
   *form := read( );*
   *form := eval(form);*
   *print(form)*
end.

or in M6800 assembly language:

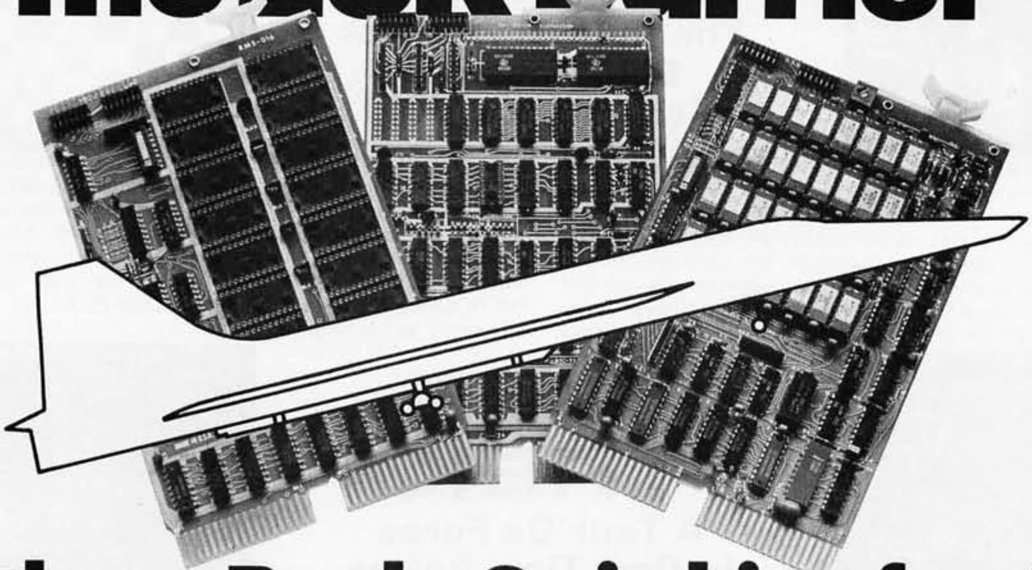
```
BIGLUP LDX   PRMPAT   get prompt atom
       JSR   PATOM    print the atom
       JSR   READ     read the form typed in
*  result now in M6800 x-register
       JSR   EVAL     eval the form
*  result of EVAL back in x-register
       JSR   PRINT    print the form
       BRA   BIGLUP     and loop around
```

PATOM is a subroutine, also called by PRINT, when a form is known to be an atom. In an assembly language implementation, it would be very convenient to pass forms in the M6800 index (X) register. This register is 16 bits long, so it requires that forms be only 16 bits. Some representation must be chosen for all LISP objects so that a single 16 bit number may uniquely specify any arbitrary object. Dotted pairs are used to represent lists. Dotted pairs hold two forms, a CAR and a CDR, so they must be 32 bit objects. A natural choice is to allocate 4 consecutive M6800 memory bytes for dotted pairs, and specify dotted pairs by the address of their first byte. This means that any two different dotted pairs will be easily differentiated by the forms that specify them.

This still leaves the problem of deciding on an internal representation for atoms, including symbolic atoms, numeric atoms, and NIL. In the author's LISP system only two items of information are needed for each symbolic atom, the string of characters which are the print name of the atom, and the value currently associated with the atom (which is an arbitrary form). Again a 4 byte representation is chosen, with the first two bytes used as a memory address pointing to the first character of the print name, and the third and fourth bytes used to hold the value (a form) of the atom. Now the address of

*Text continued on page 140*

# Let your LSI-11* break the 28K barrier



# With our Bank-Switching family

In LOCAL mode our memory is functionally just like DEC memory. But when you run out of memory space you're not lost. Add an inexpensive Bank-Switch Controller (BSC-256) and you can go to **two megabytes.** Add another and go to **four megabytes.**

So don't get boxed in with other brands of LSI-11* memory. Break free. Join the family:

| | | |
|---|---|---|
| **RMA-032** | 32K by 16 bit RAM. On-board refresh | $1200 (Single qty.) |
| **RMS-016** | 16K by 16 bit ROM. (Intel 2716) | $300 (Single qty.) |
| **BSC-256** | The Bank-Switch Controller | $300 (Single qty.) |

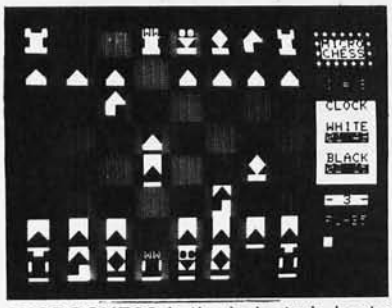
*Substantial quantity discounts are available. For a free copy of our Bank-Switching manual, call or write on your company letterhead.*

*Digital Pathways Inc.*
*4151 Middlefield Road*
*Palo Alto, CA 94306*
*(415) 493-5544*

**Registered trademark of Digital Equipment Corporation*

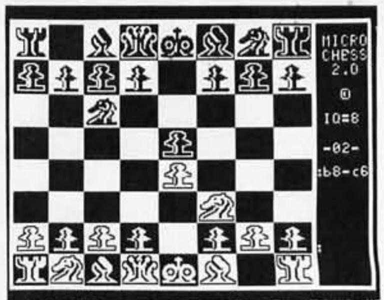
**DIGITAL PATHWAYS**

# PET / TRS-80 / APPLE: *Personal Software brings you the finest!*

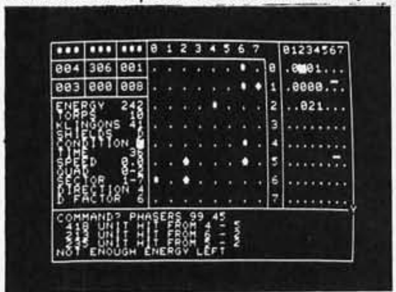



# MICRO CHESS

## The Industry's First Gold Cassette Over 50,000 Sold

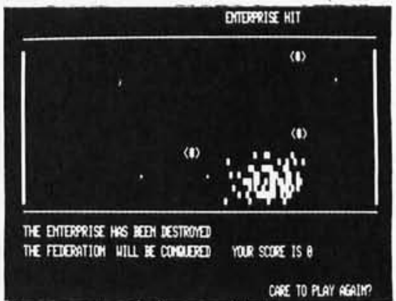



**MICROCHESS** is the industry's best selling computer game. And no wonder—because MICROCHESS gives you more than just a chessplaying program: A convenient, foolproof set of commands and error checks ... complete instructions in a 5½" by 8½" booklet ... a cassette that's guaranteed to load, with disk versions coming soon ... and several levels of difficulty to challenge you not just once, but time after time. It's available through well over three hundred computer stores and many mail order sources ... always originating from Personal Software. What's more, *every* Personal Software product is selected to give you these same benefits of easy availability, reliable cassettes, readable documentation, a carefully thought out user interface ... and most important, continuing challenge and enjoyment, not just once but time after time. If you haven't already, order your own gold cassette: MICROCHESS, by **Peter Jennings**, for 8K **PET**s, 16K **APPLE**s, and 4K Level I and II **TRS-80**s ............................... **$19.95**

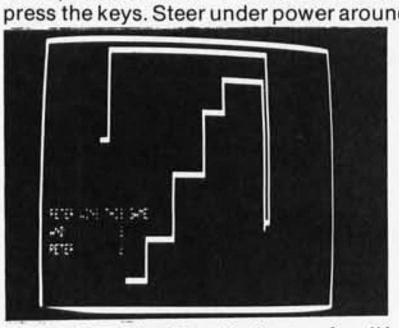


# TIME TREK

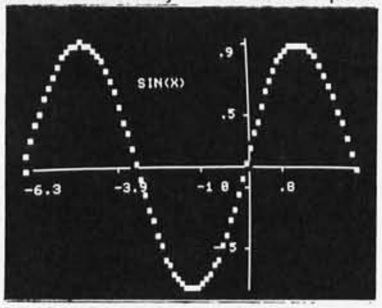
## A Tour De Force In Real Time Action Strategy Games



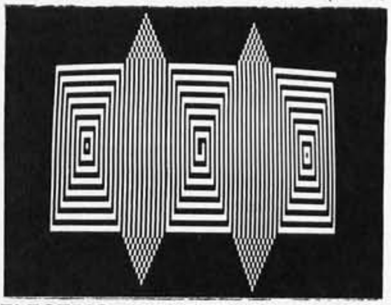

**TIME TREK** by **Brad Templeton** for 8K **PET**s and **Joshua Lavinsky** for 4K Level I and II **TRS-80**s adds a dramatic new dimension to the classic Star Trek type strategy game: REAL TIME ACTION! You'll need fast reflexes as well as sharp wits to win in this constantly changing game. Be prepared—the Klingons will fire at you as you move, and will move themselves at the same time, even from quadrant to quadrant—but with practice you can change course and speed, aim and fire in one smooth motion, as fast as you can press the keys. Steer under power around obstacles—evade enemy shots as they come towards you—lower your shields just long enough to fire your phasers, betting that you can get them back up in time! With nine levels of difficulty, this challenging game is easy to learn, yet takes most users months of play to master. ADD SOUND EFFECTS with a simple two-wire hookup to any audio amplifier; the TRS-80 also produces sound effects directly through the keyboard case, to accompany spectacular graphics explosions! You won't want to miss this memorable version of a favorite computer game ................................ **$14.95**



**BLOCKADE** by **Ken Anderson** for 4K Level I and II **TRS-80**s is a real time action game for two players, with high speed graphics in machine language. Each player uses four keys to control the direction of a moving wall. Try to force your opponent into a collision without running into a wall yourself! A strategy game at lower speeds, BLOCKADE turns into a tense game of reflexes and coordination at faster rates. Play on a flat or spherical course at any of ten different speeds. You can hear SOUND EFFECTS through a nearby AM radio—expect some razzing if you lose! ............ **14.95**




**GRAPHICS PACKAGE** by **Dan Fylstra** for 8K **PET**s includes programs for the most common 'practical' graphics applications: PLOTTER graphs both functions and data to a resolution of 80 by 50 points, with automatic scaling and labeling of the axes; BARPLOT produces horizontal and vertical, segmented and labeled bar graphs; LETTER displays messages in large block letters, using any alphanumeric or special character on the PET keyboard; and DOODLER can be used to create arbitrary screen patterns and save them on cassette or in a BASIC program .................... **$14.95**



**ELECTRIC PAINTBRUSH** by **Ken Anderson** for 4K Level I and II **TRS-80**s: Create dazzling real time graphics displays at speeds far beyond BASIC, by writing 'programs' consisting of simple graphics commands for a machine language interpreter. Commands let you draw lines, turn corners, change white to black, repeat previous steps, or call other programs. The ELECTRIC PAINTBRUSH manual shows you how to create a variety of fascinating artistic patterns including the one pictured. Show your friends some special effects they've never seen on a TV screen! ........ **$14.95**

**WHERE TO GET IT:** *Look for the* **PERSONAL SOFTWARE™** *display rack at your local computer store. If you can't find the product you want, you can order direct with your VISA/Master Charge card by dialing* **1-800-325-6400** *toll free (24 hours, 7 days; in Missouri, dial 1-800-342-6600). If you have questions, please call 408-745-7841. Or you can mail your order to the address below.*

## Personal Software™

592 Weddell Drive
Sunnyvale, Calif. 94086



Circle 302 on inquiry card.

# Look for Personal Software™ products at the dealer nearest you!

**ALABAMA**
BYTE SHOP
Huntsville, AL 35805
COMPUTERLAND
Huntsville, AL 35805
CPU, INC.
Montgomery, AL 36104
THE LOGIC STORE
Opelika, AL 36801
**ALASKA**
ALPHA ELECTRONICS
Anchorage, AK 99503
**ARIZONA**
MILLET'S TV & RADIO
Mesa, AZ 85204
PERSONAL COMPUTER PLACE
Mesa, AZ 85202
COMPUTERLAND OF PHOENIX
Phoenix, AZ 85016
COMPUTER SHOWROOM
Tucson, AZ 85710
**ARKANSAS**
COMPUTERLAND
Little Rock, AR 72212
DATACOPE
Little Rock, AR 72204
**CALIFORNIA**
JAY-KERN ELECTRONICS
Bakersfield, CA 93305
BYTE SHOP
Burbank, CA 91506
SILVER SPUR
Chino, CA 91710
BYTE SHOP OF SACRAMENTO
Citrus Heights, CA 95610
COAST COMPUTER CENTER
Costa Mesa, CA 92627
CAPITOL COMPUTER SYSTEMS
Davis, CA 95616
COMPUTERLAND SAN DIEGO EAST
El Cajon, CA 92020
COMPUTERLAND OF EL CERRITO
El Cerrito, CA 94530
COMPUTERWARE
Encinitas, CA 92024
BUSINESS ENHANCEMENT
COMPUSERVICE
Escondido, CA 92027
CHANNEL DATA SYSTEMS
Goleta, CA 93017
RAINBOW COMPUTING
Granada Hills, CA 91344
JADE COMPUTER PRODUCTS
Hawthorne, CA 90250
BYTE SHOP OF HAYWARD
Hayward, CA 94541
COMPUTERLAND OF HAYWARD
Hayward, CA 94541
COMPUTERLAND OF WEST L.A.
Inglewood, CA 90302
PROFESSIONAL COMPUTER STORE
La Crescenta, CA 91214
COMPUTER COMPONENTS
OF SOUTH BAY
Lawndale, CA 90260
COMPUTERLAND OF SOUTH BAY
Lawndale, CA 90260
A-VIDD ELECTRONICS
Long Beach, CA 90815
COMPUTERLAND
Los Altos, CA 94022
BYTE SHOP
Mountain View, CA 94040
HOBBY WORLD ELECTRONICS
Northridge, CA 91324
COMPUTERS-MADE-EASY
Palmdale, CA 93550
BYTE SHOP OF PLACENTIA
Placentia, CA 92670
COMPUTER CENTER
Riverside, CA 92503
CAPITOL COMPUTER SYSTEMS
Sacramento, CA 95821
COMPUTERLAND
San Bernardino, CA 92404
COMPUTER AGE INC.
San Diego, CA 92111
COMPUTERLAND OF SAN DIEGO
San Diego, CA 92111
COMPUTER MERCHANT
San Diego, CA 92115
COMPUTERLAND OF
SAN FRANCISCO
San Francisco, CA 94105
VIDEO GAMES & COMPUTERS
San Francisco, CA 94118
VILLAGE ELECTRONICS
San Francisco, CA 94121
COMPUTERLAND OF SAN JOSE
San Jose, CA 95129
COMPUTERLAND (Central)
San Leandro, CA 94577
BYTE SHOP
San Luis Obispo, CA 93401
MARIN COMPUTER CENTER
San Rafael, CA 94903
ADVANCED COMPUTER PRODUCTS
Santa Ana, CA 92705
COMPUTER CITY
Santa Ana, CA 92704
BYTE SHOP
Santa Clara, CA 95051
COMPUTER FORUM
Santa Fe Springs, CA 90670
THE COMPUTER STORE
Santa Monica, CA 90401
SANTA ROSA COMPUTER CENTER
Santa Rosa, CA 95404
BYTE SHOP
Suisun, CA 94585
COMPUTERS PLUS
Sunnyvale, CA 94087
BYTE SHOP OF TARZANA
Tarzana, CA 91356
COMPUTERLAND OF
THOUSAND OAKS
Thousand Oaks, CA 91360
SMALL SYSTEM SOFTWARE
Thousand Oaks, CA 91360
COMPUTER COMPONENTS
Van Nuys, CA 91411
COMPUTERLAND
Walnut Creek, CA 94598
BYTE SHOP
Westminster, CA 92683
COMPUTER COMPONENTS OF
ORANGE COUNTY
Westminster, CA 92683
**COLORADO**
BYTE SHOP
Boulder, CO 80301
COMPUTERLAND
Colorado Springs, CO 80917
AMPTEC
Denver, CO 80216
COMPUTERLAND
Denver, CO 80222
BYTE SHOP
Englewood, CO 80110
MICRO WORLD ELECTRONIX
Lakewood, CO 80226
**CONNECTICUT**
COMPUTERLAND OF FAIRFIELD
Fairfield, CT 06430
JRV COMPUTER STORE
Hamden, CT 06518
THE COMPUTER STORE
Hartford, CT 06103
COMPUTER LAB
New London, CT 06320
THE COMPUTER STORE
Windsor Locks, CT 06096
**WASHINGTON D.C.**
COMPUTER CABLEVISION
Washington, D.C. 20007
**FLORIDA**
COMPUTERLAND
Boca Raton, FL 33432
THE COMPUTER STORE
Bradenton, FL 33505
THE COMPUTER STORE
Clearwater, FL 33516
TRANS-DATA CORP
Coral Gables, FL 32134
UCATAN
Destin, FL 32541
BYTE SHOP
Fort Lauderdale, FL 33334
COMPUTERLAND
Fort Lauderdale, FL 33308
COMPUTERS FOR YOU
Fort Lauderdale, FL 33312
DATA MOVERS
Fort Meyers, FL 33901
SOUND IDEAS BYTE SHOPPE
Gainesville, FL 32601
FOCUS SCIENTIFIC ENTERPRISES
Miami, FL 33132
GRICE ELECTRONICS
Pensacola, FL 32589
COMPUTER AGE
Pompano Beach, FL 33062
PAPERBACK BOOKSMITH
Sarasota, FL 33581
AMF ELECTRONICS
Tampa, FL 33612
MICRO COMPUTER SYSTEMS
Tampa, FL 33609
COMPUTER CENTER OF
PALM BEACHES
West Palm Beach, FL 33409
**GEORGIA**
ADVANCE COMPUTER TECHNOLOGIES
Atlanta, GA 30328
COMPUSHOP
Atlanta, GA 30342
DATAMART
Atlanta, GA 30305
THE LOGIC STORE
Columbus, GA 31906
COMPUTERLAND OF ATLANTA
Smyrna, GA 30080
**HAWAII**
COMPUTERLAND
Honolulu, HI 96813
MICROCOMPUTER SYSTEMS
Honolulu, HI 96813
RADIO SHACK (Dealer)
Lihue, HI 96766
**IDAHO**
NORTHWEST COMPUTER CENTER
Boise, ID 83704
**ILLINOIS**
COMPUTERLAND OF
ARLINGTON HEIGHTS
Arlington Heights, IL 03904
FARNSWORTH COMPUTER CENTER
Aurora, IL 60505
KAPPEL'S COMPUTER STORE
Belleville, IL 62220
DOW-COM
Carbondale, IL 62901
BYTE SHOP
Champaign, IL 61820
THE ELEKTRIK KEYBOARD
Chicago, IL 60614
EMMANUEL B. GARCIA JR.
AND ASSOCIATES
Chicago, IL 60613
PERSONAL COMPUTER
Chicago, IL 60611
VIDEO ODYSSEY
Deerfield, IL 60015
COMPUTERLAND
Downers Grove, IL 60515
COMPUTER STATION
Granite City, IL 62040
ORCUTT BUSINESS MACHINES
La Salle, IL 61301
ILLINI MICROCOMPUTERS
Naperville, IL 60540
COMPUTERLAND OF NILES
Niles, IL 60648
COMPUTERLAND
Oak Lawn, IL 60453
BIES SYSTEMS
Oak Park, IL 60302
COMPUTERLAND OF PEORIA
Peoria, IL 61614
WALLACE ELECTRONICS
Peoria, IL 61614
DATA DOMAIN
Schaumburg, IL 60195
**INDIANA**
DATA DOMAIN OF FORT WAYNE
Fort Wayne, IN 46805
HOME COMPUTER CENTER
Indianapolis, IN 46220
PUBLIC COMPUTING
Lafayette, IN 47904
COMPUTER CENTER
So. Bend, IN 46637
**IOWA**
SYNCHRONIZED SYSTEMS
Des Moines, IA 50310
COMPUTER SHOP
Spencer, IA 51301
THE COMPUTER CENTER
Waterloo, IA 50701
**KANSAS**
THE COMPUTER ROOM
Overland Park, KS 66212
PERSONAL COMPUTER CENTER
Overland Park, KS 66206
COMPUTER SYSTEMS DESIGN
Wichita, KS 67214
**KENTUCKY**
BARNEY MILLER'S INC.
Lexington, KY 40507
**LOUISIANA**
COMPUTER SHOPPE
Metaire, LA 70002
**MARYLAND**
COMPUTERLAND
Rockville, MD 20855
COMPUTER WORKSHOP
Rockville, MD 20852
COMPUTERS ETC.
Towson, MD 21204
COMPUTERS UNLIMITED
Towson, MD 21204
**MASSACHUSETTS**
THE COMPUTER STORE
Burlington, MA 01803
THE COMPUTER STORE
Cambridge, MA 02139
CPU SHOP
Charlestown, MA 02129
MAD HATTER SOFTWARE
Dracut, MA 01826
NEW ENGLAND ELECTRONICS
Needham, MA 02194
**MICHIGAN**
NEWMAN COMPUTER EXCHANGE
Ann Arbor, MI 48104
NEW DIMENSIONS IN COMPUTING
East Lansing, MI 48823
COMPUTER HOUSE DIV.
Jackson, MI 49202
COMPUTERLAND OF
GRAND RAPIDS
Kentwood, MI 49508
COMPUTRONIX
Midland, MI 48640
COMPUTER MART
Clawson, MI 48017
TRI CITY COMPUTER MART
Saginaw, MI 48603
COMPUTERLAND
Southfield, MI 48034
LEVEL FOUR PRODUCTIONS
Westland, MI 48185
**MINNESOTA**
COMPUTERLAND
Bloomington, MN 55431
ZIM COMPUTERS INC.
Brooklyn Center, MN 55429
MINN. MICRO SYSTEMS
Minneapolis, MN 5' 54
**MISSISSIPPI**
OXFORD SOFTWAR 0.
Oxford, MS 38655
**MISSOURI**
FORSYTHE COMPUTERS
Clayton, MO 63105
COMPUTER COUNTRY
Florissant, MO 63031
FUTUREWORLD, INC.
St. Louis, MO 63131
GREATEST GRAPHICS
Springfield, MO 65804
**NEBRASKA**
OMAHA COMPUTER STORE
Omaha, NE 68127
**NEVADA**
CENTURY 23
Las Vegas, NV 89102
HOME COMPUTERS
Las Vegas, NV 89109
**NEW HAMPSHIRE**
TRS-80 SOFTWARE EXCHANGE
Milford, NH 03055
COMPUTERLAND OF NASHUA
Nashua, NH 03060
BITS, INC.
Peterborough, NH 03458
**NEW JERSEY**
COMPUTER LAB OF NJ
Budd Lake, NJ 07828
COMPUTER EMPORIUM
Cherry Hill, NJ 08002
COMPUTER MART OF NJ
Iselin, NJ 08830
MSM ELECTRONICS
Medford, NJ 08055
COMPUTERLAND
Morristown, NJ 07960
COMPUTERLAND
Paramus, NJ 07652
COMPUTER NOOK
Pine Brook, NJ 07058
COMPUTER CORNER
Pompton Lakes, NJ 07442
COMPUTER ENCOUNTER
Princeton, NJ 08540
TYPTRONIC COMPUTER STORE
Ramsey, NJ 07446
STONEHENGE COMPUTER CO.
Summit, NJ 07901
**NEW YORK**
COMPUTERLAND
Buffalo, NY 14150
COMPUTERLAND
Carle Place, NY 11514
COMPUTER SHOP OF SYRACUSE
De Witt, NY 13214
THE COMPUTER TREE
Endwell, NY 13760
COMPUTERWORLD INC.
Great Neck, NY 11021
LONG ISLAND COMPUTER
GENERAL STORE
Lynbrook, NY 11563
COMPUTER MICROSYSTEMS
Manhasset, NY 11030
COMPUTER SHOPPE
Middle Island, NY 11953
ARISTO-CRAFT
New York, NY 10001
THE COMPUTER FACTORY
New York, NY 10017
COMPUTER MART OF NEW YORK
New York, NY 10016
DATEL SYSTEMS
New York, NY 10036
AUTOMATIC SYSTEMS
Poughkeepsie, NY 12603
COMPUTER HOUSE
Rochester, NY 14609
THE COMPUTER STORE
Rochester, NY 14618
HOME COMPUTER CENTER
Rochester, NY 14607
THE COMPUTER CORNER
White Plains, NY 10601
READOUT COMPUTER STORE
Williamsville, NY 14221
**NORTH CAROLINA**
BYTE SHOP
Charlotte, NC 28212
COMPUTERLAND
Charlotte, NC 28205
COMPUTER ROOM
Charlotte, NC 28203
FUTUREWORLD
Durham, NC 27707
BYTE SHOP
Greensboro, NC 27401
MICROCOMPUTER SERVICES
Hickory, NC 28601
BYTE SHOP OF RALEIGH
Raleigh, NC 27605
**OHIO**
BASIC COMPUTER SHOP
Akron, OH 44314
INDUCTIVE COMPONENTS
Amelia, OH 45102
CINCINNATI COMPUTER STORE
Cincinnati, OH 45246
21ST CENTURY SHOP
Cincinnati, OH 45202
DIGITAL DESIGN
Cincinnati, OH 45202
CYBER SHOP
Columbus, OH 43227
MICRO MINI COMPUTER WORLD
Columbus, OH 43213
COMPUTER SOLUTIONS
Dayton, OH 45409
DAYTON COMPUTER MART
Dayton, OH 45409
ASTRO VIDEO ELECTRONICS
Lancaster, OH 43130
COMPUTERLAND OF CLEVELAND
Mayfield Heights, OH 44121
RADIO SHACK (Dealer)
St. Clairsville, OH 43950
**OKLAHOMA**
HIGH TECHNOLOGY
Oklahoma City, OK 73106
MICROLITHICS
Oklahoma City, OK 73127
HIGH TECHNOLOGY
Tulsa, OK 74129
**OREGON**
THE COMPUTER STORE
Corvallis, OR 97330
CAMERA AND COMPUTER
EMPORIUM
Portland, OR 97205
COMPUTERLAND OF PORTLAND
Tigart, OR 97223
**PENNSYLVANIA**
BYTE SHOP
Bryn Mawr, PA 19010
PERSONAL COMPUTER CENTER
Frazer, PA 19355
COMPUTER AID
Latrobe, PA 15650
THE COMPUTER WORKSHOP
Murrysville, PA 15668
A B COMPUTERS
Montgomeryville, PA 18936
MICROTRONIX
Philadelphia, PA 19106
COMPUTER HOUSE
Pittsburgh, PA 15220
**SOUTH CAROLINA**
DATA MART
Greenville, SC 29607
**TENNESSEE**
MICROCOMPUTER STORE
Knoxville, TN 37919
COMPUTER LABS OF MEMPHIS
Memphis, TN 38117
DOC'S COMPUTER SHOP
Nashville, TN 37211
**TEXAS**
COMPUTER POST
Arlington, TX 76011
COMPUTERLAND OF AUSTIN
Austin, TX 78757
COMPUTERS 'N THINGS
Austin, TX 78731
MICRO COMPUTER SHOPPE
Corpus Christi, TX 78411
MICROSYSTEMS SERVICES INC.
Corpus Christi, TX 78411
COMPUSHOP
Dallas, TX 75243
COMPUTER IMAGINEERING
Dallas, TX 75234
COMPUTERLAND
Dallas, TX 75231
KA ELECTRONICS SALES
Dallas, TX 75247
COMPUTER TERMINAL
El Paso, TX 79901
RAM MICRO SYSTEMS
Fort Worth, TX 76116
COMPUTERCRAFT
Houston, TX 77063
COMPUTERLAND OF HOUSTON BAY
Houston, TX 77058
INTERACTIVE COMPUTERS
Houston, TX 77036
NEIGHBORHOOD COMPUTER
Lubbock, TX 79401
COMPUTER PATCH OF SANTA FE
Odessa, TX 79762
COMPUSHOP
Richardson, TX 75080
THE COMPUTER SHOP
San Antonio, TX 78216
COMPUTER SOLUTIONS
San Antonio, TX 78229
MICRO MART
San Antonio, TX 78205
WICHITA COMPUTER SYSTEMS
Wichita Falls, TX 76301
**UTAH**
ADP SYSTEMS
Logan, UT 84321
TRI-POWER ELECTRONICS
Murray, UT 84017
COMPUTER SPECIALISTS
Ogden, UT 84402
COMPUTER CONCEPTS GROUP
Salt Lake City, UT 84109
COMPUTERLAND OF SALT LAKE
Salt Lake City, UT 84111
THE HI-FI SHOP
Salt Lake City, UT 84117
**VERMONT**
COMPUTERMART
Essex Junction, VT 05452
**VIRGINIA**
COMPUTER HARDWARE STORE
Alexandria, VA 22314
COMPUTERS PLUS
Alexandria, VA 22304
COW, INC.
Blacksburg, VA 24060
HOME COMPUTER CENTER
Newport News, VA 23606
COMPUTER TECHNIQUES
Richmond, VA 23235
THE COMPUTER PLACE
Roanoke, VA 24015
COMPUTER WORKSHOP
Springfield, VA 22151
COMPUTERLAND
Vienna, VA 22180
HOME COMPUTER CENTER
Virginia Beach, VA 23452
**WASHINGTON**
COMPUTERLAND
Bellevue, WA 98007
OMEGA NORTHWEST
Bellevue, WA 98004
COMPUTERLAND OF SOUTH
KING COUNTY
Federal Way, WA 98003
YE OLDE COMPUTER SHOPPE
Richland, WA 99352
THE COMPUTER SHOPPE
Seattle, WA 98115
EMPIRE ELECTRONICS
Seattle, WA 98166
PERSONAL COMPUTERS
Spokane, WA 99202
COMPUTERLAND
Tacoma, WA 98499
**WISCONSIN**
BYTE SHOP OF MILWAUKEE
Greenfield, WI 53227
COMPUTERLAND
Madison, WI 53711
MADISON COMPUTER STORE
Madison, WI 53711
COMPUTERLAND
Milwaukee, WI 53222
FOX VALLEY
COMPUTER STORE
Neenha, WI 54956
**WYOMING**
COMPUTER CONCEPTS
Cheyenne, WY 82001
**AUSTRALIA**
ELECTRONIC CONCEPTS PTY. LTD.
COMPUTERLAND
Sydney, N.S.W.
**CANADA**
COMPUSHOP
Calgary, Alberta T2N 2A4
THE COMPUTER SHOP
Calgary, Alberta T2T 4T9
ORTHON COMPUTERS
Edmonton, Alberta T5N 3N3
TJB MICROSYSTEMS
Edmonton, Alberta T5M 0H9
CONTI ELECTRONICS
Vancouver, B.C. V5W 2Z4
COMPUTER CITY
Winnepeg, Manitoba R3P 0H8
COMPUTERLAND
Winnepeg, Manitoba R3G 0M8
INTERACTIVE COMPUTER SYSTEMS
Frederickton, New Brunswick
MINICOMP SYSTEMS
Halifax, Nova Scotia B3K 2G1
KOBETEK SYSTEMS
Wolfville, Nova Scotia B0P 1X0
COMPUTERLAND
Burlington, Ontario
LYNTRONICS
Downsview, Ontario M2J 2W6
COMPUTER CIRCUITS
London, Ontario N6A 3H2
COMPUMART
Ottawa, Ontario K2A 1J2
COMPUTER INNOVATIONS
Ottawa, Ontario K1B 4A8
RICHVALE TELECOMMUNICATIONS
Richmond Hill, Ontario
THE COMPUTER CENTRE
Sarnia, Ontario N7T 1B4
COMPUTER MART
Toronto, Ontario M4G 3B5
THE COMPUTER PLACE
Toronto, Ontario M5V 1Z1
COMPUTER SPECIALIST
Toronto, Ontario M3K 1E7
HOME COMPUTER CENTRE
Toronto, Ontario M2M 3W2
HOUSE OF COMPUTERS
Toronto, Ontario
MARKETRON
Toronto, Ontario
MICRO-WARE
Toronto, Ontario M4E 2L2
COMPUCENTRE
Montreal, Quebec H1J 1Z4
FUTUR BYTE
Montreal, Quebec H3B 3C9
CUSTOM COMPUTING SERVICES
Saskatoon, Saskatchewan S7K 2B5
DIGITAL SERVICE
Saskatoon, Saskatchewan S7J 3A9
**PUERTO RICO**
MICROCOMPUTER STORE
Rio Piedras, PR 00921

Circle 302 on inquiry card.

*Text continued from page 136*
this 4 byte object can specify the atom uniquely from all other atoms and from all other dotted pairs.

Unfortunately this does not provide a simple way of distinguishing atoms from dotted pairs, when just given the form. Several solutions to this problem are possible. One is to restrict dotted pairs to a certain part of memory, then the address would determine whether the form specified an atom or a dotted pair. A second method is to add an additional byte to both dotted pairs and atoms which simply contains a type specifier, say 1 for dotted pairs and 2 for atoms. This method makes future expansion of types simple, but is somewhat wasteful in terms of space. The third method, the one chosen for the author's system, is to align all dotted pairs and atoms on 4 byte boundaries, that is, with addresses which are a multiple of four. This means that the low order two bits of the address are expected to always be zero, and hence may be used to encode type information. In the author's system, dotted pairs are specified by forms with both bits zero, and symbolic atoms by *01* in the lower two bits. One of the bits is still unused, but will become very handy when *garbage collection* methods are discussed below.

With numeric atoms, their name determines their value, and hence only their name (or their value) need be specified by a form. On the author's M6800 system only hexadecimal memory addresses 0000 thru 7FFF were accessible for storage of dotted pairs and atoms, meaning that the high order bit of forms specifying either of these was always zero. A representation for numeric atoms was chosen to be a form with the high order bit set, 14 bits of numeric value, and one bit left for *garbage collection*.

A special representation for the NIL atom is used both because the value of NIL is, like numeric atoms, required always to be the atom itself, and because it is used universally to represent the end of a list. The form chosen to specify NIL is simply the value zero. In fact any form with the high order byte zero is treated like NIL to simplify the test for NIL in certain cases. This means that the 256 byte page starting at zero is not usable for storing atoms or dotted pairs, but this restriction causes no problem at all, since both are allocated starting at the highest address available, and the allocator runs into program long before it reaches page zero.

When writing a LISP interpreter, the implementor must decide relatively early on how forms will specify all types of LISP objects. Unfortunately, it may not be until well into the implementation that the implementor discovers that certain choices were inefficient or inconvenient.

One important requirement affecting this decision not yet mentioned is the need to implement the LISP EQ function. This function takes two arbitrary forms, and returns the atom T or the atom NIL depending on whether the forms specify the same dotted pair, or whether the forms specify the same atom. Whenever an atom is input by READ, it must return the form specifying that atom to the caller. Whenever the same symbolic atom name is typed, READ must return the same form, ie: a pointer to the same 4 byte cell. This is accomplished by retaining a linked list of all defined symbolic atoms (called the OBLIST).

Before allocating a new 4 byte cell for an atom, READ scans the OBLIST for an atom of the given print name. If found, READ returns a form specifying that pre-exisiting atom. (Otherwise it must copy the name into some area used for storing names, allocate a 4 byte cell, initialize the left cell to point to the name, and the right cell to NIL, and return a form specifying the new atom.) This method guarantees that two forms specify the same symbolic atom if and only if they have the same address.

In some implementations of numeric atoms, this same rule cannot be guaranteed. In such systems, numeric atoms are simply allocated an appropriately large cell to store their numeric value (and hence allowing numeric atoms greater than 14 bits), a new cell being allocated every time a new number is generated (which happens at every ADD, MUL, etc). In these systems it would be impractical to scan a list like the OBLIST every time any arithmetic calculation is done, and so the LISP function EQ may not rely on the rule that unequal forms indicate unequal atoms. In such systems, EQ must look at the contents of the cell specified by a numeric atom form, and make the comparison that way. In systems like the author's, EQ simply compares the forms themselves, no matter what type of atom the form may specify.

The choices made in representing the various types of LISP objects can be summarized in the high level language (Pascal-like) data structure specification in listing 1.

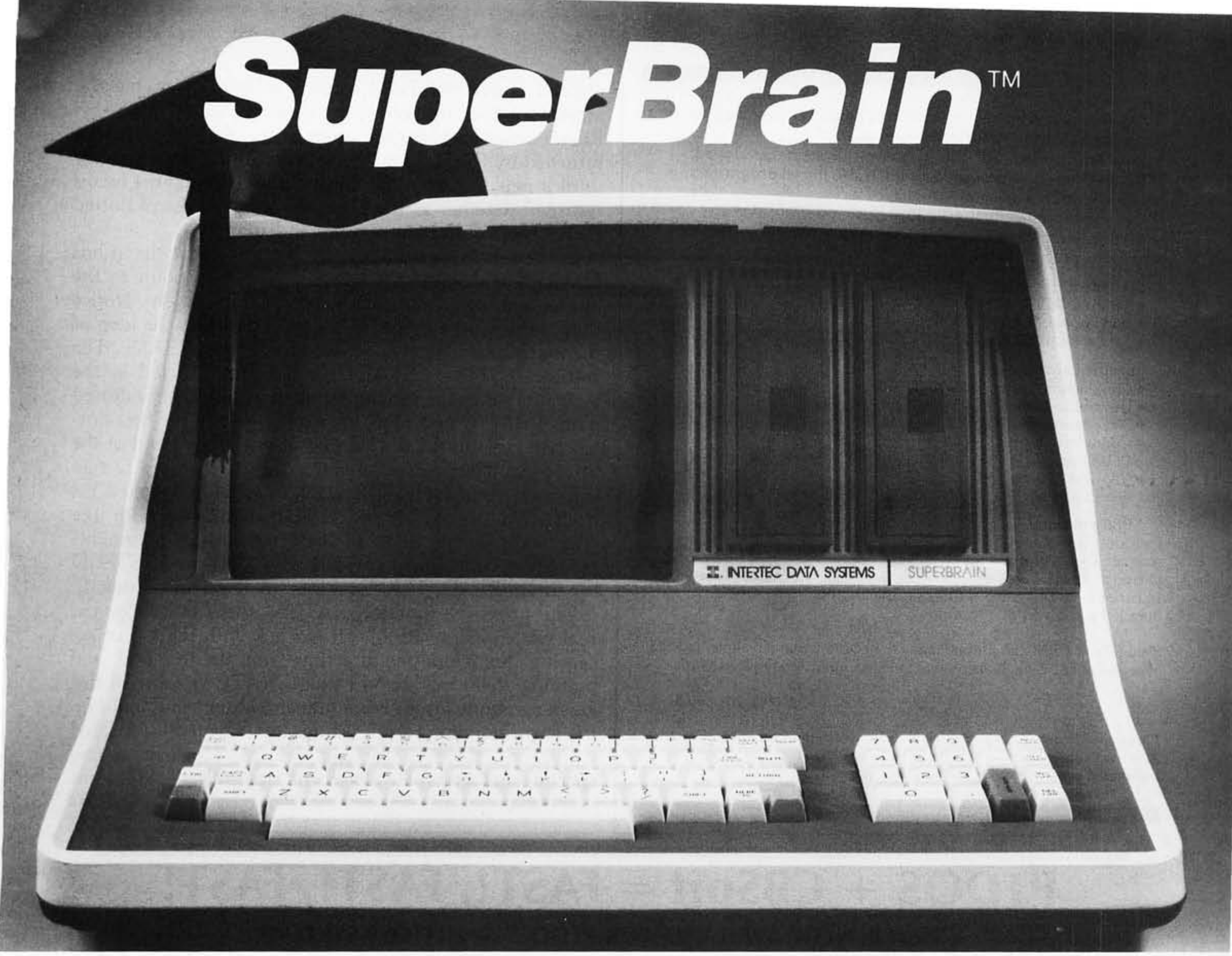
```
type lisptype =
        (dtprtype, symatmtype, numatmtype, nilatmtype);
    dtpr =
        record
            car: form;
            cdr: form
        end;
    symatm =
        record
            name: ↑array [0..n] of char;
            value: form
        end;
    form =
        packed record
            gcbit: boolean;
            case objtype: lisptype of
                dtprtype:      (dtprform:     ↑dtpr);
                symatmtype:    (symatmform:   ↑symatm);
                numatmtype:    (numatmform:   -5000..4999);
                nilatmtype:    ( )
        end.
```

*Listing 1: A Pascal data structure specification that could be used to represent various types of LISP objects.*

## READ Function

READ is the basic input routine for the LISP interpreter. READ accepts a fully parenthesized expression from the terminal, and builds up the internal representation, allocating new dotted pairs and atoms as necessary. If the expression is a list, READ returns a form specifying the first dotted pair of the constructed list. If the expression typed in is simply an atom, READ returns a form specifying the atom.

The logic of the READ routine is straightforward because the syntax of LISP expressions is so simple. READ calls a function RATOM to return the next input atom. RATOM actually does the work of allocating new 4 byte cells for symbolic atoms (when necessary) as ex-




# The Honor Graduate

There's been a lot of talk lately about intelligent terminals with small systems capability. And, it's always the same. The systems which make the grade in performance usually flunk the test in price. At least that was the case until the SuperBrain graduated with the highest PPR (Price/Performance Ratio) in the history of the industry.

For less than $3,000*, SuperBrain users get exceptional performance for just a fraction of what they'd expect to pay. Standard features include: two dual-density mini-floppies with 320K bytes of disk storage, 64K of RAM to handle even the most sophisticated programs, a CP/M Disk Operating System with a high-powered text editor, assembler and debugger. And, with SuperBrain's S-100 bus adapter, you can even add a 10 megabyte disk!

More than an intelligent terminal, the SuperBrain outperforms many other systems costing three to five times as much. Endowed with a hefty amount of available software (BASIC, FORTRAN, COBOL), the SuperBrain is ready to take on your toughest assignment. You name it! General Ledger, Accounts Receivable, Payroll, Inventory or Word Processing . . . the SuperBrain handles all of them with ease.

Your operators will praise the SuperBrain's good looks. A full ASCII keyboard with a numeric keypad and function keys. A non-glare, dynamically focused, twelve inch screen. All in an attractive desktop unit weighing less than a standard office typewriter. Sophisticated users will acclaim SuperBrain's twin Z-80 processors which transfer data to the screen at 38 kilobaud! Interfacing a printer or modem is no problem using SuperBrain's RS-232C communications port. But best of all, you won't need a PhD in computer repair to maintain the SuperBrain. Its single board design makes servicing a snap!

So don't be fooled by all the freshman students in the small systems business. Insist on this year's honor graduate . . . the SuperBrain.

INTERTEC DATA SYSTEMS®
2300 Broad River Road, Columbia, SC 29210
(803) 798-9100 TWX: 810-666-2115

*Quantity one. Dealer inquiries invited.

Circle 180 on inquiry card.

plained above. RATOM returns a form specifying the atom typed. If this atom is anything but the atom "(" READ simply returns the atom as its result. If the atom returned by RATOM is "(", READ calls itself recursively until it gets the atom ")", meanwhile stringing the forms returned together as the CARs on a linked list of dotted pairs. This could be written as in listing 2.

In the LISP functions we are assuming that the atoms LPAREN and RPAREN were initialized to point to the atoms with print names "(" and ")" respectively. Notice that in the LISP version, READ accomplishes the loop of the machine code version with recursion in READL. The routines LSTINI, LSTADD, and LSTEND used in the assembly language version build up a linked list of dotted pairs, using two pointers on a stack, one to the first dotted pair, one to the dotted pair at the current end of the linked list. The pointers are on a stack so that READ may call itself recursively. The stack is actually a linked list itself. The linked-list stack is manipulated with the routines in listing 3. With these routines it is straightforward to implement LSTINI, LSTADD, and LSTEND for use in READ. These routines are shown in listing 4.

The primitive function RATOM turns out to be the real workhorse of READ. It is stuck with the job of accepting characters one at a time from the terminal, and building them up into an atom. RATOM must distinguish symbolic atoms from numeric atoms, and build up the corresponding forms. Atoms are in general separated by spaces, tabs, or carriage returns. However a few special characters always form single-character atoms

*Text continued on page 145*

*Listing 2: LISP (a) and M6800 assembly (b) code for the READ function.*

(a)

```
(DEF READ (LAMBDA ()
    (READR (RATOM))
))

(DEF READR (LAMBDA (A)
    (COND
        ((EQ A LPAREN) (READL (READ)))
        (T A)
    )
))

(DEF READL (LAMBDA (F)
    (COND
        ((EQ F RPAREN) NIL)
        (T (CONS F (READL (READ))))
    )
))
```

(b)

```
READ    JSR RATOM  pick up the next input atom
        CPX LPARAT is it equal to the "(" atom?
        BNE RDRET  no, simply return the atom
        JSR LSTINI yes, initialize a linked list
RDLUP   BSR READ   call READ recursively to pick up next form
        CPX RPARAT is it equal to the ")" atom?
        BEQ RDLDUN yes, finalize the list and return
        JSR LSTADD no, allocate a new dotted pair,
*                    fill in the CAR, and add it to the list,
        BRA RDLUP     and loop around.
RDLDUN  JSR LSTEND finalize the list, get a pointer to
*                    the first dotted pair of the list
RDRET   RTS            and return.
```

```
* set up the stack for LSTADD.
*  first stack a NIL, then a pointer to the NIL.
LSTINI LDX   ZERO    stack a NIL form
       JSR   PUSHX
       LDX   STKPTR stack a pointer to the NIL
       DEX             as though it were the CDR of a cell.
       DEX
       JMP   PUSHX  push it and return.

* add form in X-reg to list pointed to by value on top of stack
*  clobbers X-reg, A-reg, and B-reg.
LSTADD JSR   GETCEL get a dotted pair and fill in the CAR
       STX   CELPTR save address of new cell
       JSR   TOPX   retrieve pointer from top of stack
       LDAA  CELPTR link new cell onto list
       STAA  CDR,X
       LDAB  CELPTR+1
       STAB  CDR+1,X
       LDX   STKPTR update list pointer
       STAA  CAR,X    (which is on top of stack)
       STAB  CAR+1,X
       RTS                and return.

* pop off list end pointer
*  return pointer to first dotted pair of list in X-reg
LSTEND JSR   POPX     pop off and ignore list end pointer
       JMP   POPX     pop off and return list begin pointer.
```

*Listing 4: LSTINI, LSTADD, and LSTEND build up a linked list of dotted pairs using two pointers on a stack, one to the first dotted pair, one to the dotted pair at the current end of the linked list.*

*Listing 3: M6800 implementation of linked-list stack manipulation routines.*

```
*  PUSHX saves the value of the X-reg on a linked-list stack.
PUSHX  BSR   GETCEL allocate a new dotted pair and fill in the CAR
       LDAA  STKPTR fill in the CDR from STKPTR
       STAA  CDR,X    (CAR and CDR are symbols defined as
       LDAA  STKPTR+1  0 and 2 respectively)
       STAA  CDR+1,X
       STX   STKPTR update the STKPTR
       LDX   CAR,X  restore the X-reg
       RTS              and return.

* allocate a new dotted pair, save X-reg in the CAR,
*  and set the CDR to NIL
GETCEL STX   XTMP    save the X-reg temporarily
       LDX   FREPTR pick up a free dotted pair off the FREE list.
       BEQ   NOFREE no more left, so it goes...
       LDAA  CDR,X  update the FREE list pointer
       STAA  FREPTR
       LDAA  CDR+1,X
       STAA  FREPTR+1
       CLR   CDR,X  set the CDR to NIL
       CLR   CDR+1,X
       LDAA  XTMP    fill in the CAR from the X-reg
       STAA  CAR,X
       LDAA  XTMP+1
       STAA  CAR+1,X
       RTS              and return.

* restore the X-reg from the linked-list stack
POPX   LDX   STKPTR point to the top of the stack
       LDAA  CDR,X  update the STKPTR
       STAA  STKPTR
       LDAA  CDR+1,X
       STAA  STKPTR+1
       BRA   FRECEL   and free up the dotted pair used.

* free up a dotted pair, and load X-reg from the CAR
FRECEL LDAA  FREPTR link the cell onto the FREE list
       STAA  CDR,X
       LDAA  FREPTR+1
       STAA  CDR+1,X
       LDX   CAR,X    load X-reg from CAR
       RTS                and return.

* return value at top of stack in X-reg
*  (but leave it on stack)
TOPX   LDX   STKPTR return CAR of cell on top of stack.
       LDX   CAR,X
       RTS
```

*Listing 5: RATOM accepts characters one at a time from the terminal and builds them into atoms.*

```
* return form in X-reg for next atom typed in
RATOM   LDX     SPCPTR  save pointer to first open spot in name area
        STX     NAMPTR
        LDAA    PEEKC   PEEKC always contains the next character typed.
SEPLUP  CMPA    #$20    is PEEKC a separator (space or any control character)?
        BGT     GNAM    no, go collect name of atom
        BSR     READC   yes, get next char
        BRA     SEPLUP    and loop.

* get next char of input, and store in global PEEKC
READC   JSR     GETC    use get char routine in ROM monitor
        STAA    PEEKC   save it in PEEKC
        RTS               and return with it also in A-reg.

* collect name of atom, build up form, etc.
GNAM    BSR     COPYC   save char in name area
        BSR     SPCLCH  is it a special char ("(" or ")")?
        BNE     RCLUP   no, collect multi-character name
        BSR     READC   yes, update PEEKC
        BRA     SCANOL    and go scan OBLIST
RCLUP   BSR     READC   get next character of name
        CMPA    #$20    separator?
        BLE     SCANOL  yes, all done with name
        BSR     SPCLCH  special char?
        BEQ     SCANOL  yes, end atom now
        BSR     COPYC   no, copy this char to name area
        BRA     RCLUP     and loop.

* copy char in A-reg to name area, and advance name space pointer
COPYC   LDX     SPCPTR  point to empty slot in name space
        STAA    0,X     store the char
        INX             increment the pointer
        STX     SPCPTR
        RTS               and return.

* check if char is a special char (i.e. "(" or ")")
SPCLCH  CMPA    #'(     l. paren?
        BEQ     SPCLY
        CMPA    #')     or r. paren?
SPCLY   RTS             return with Z-bit set appropriately.

* null-terminate name, check if numeric, scan OBLIST if not.
SCANOL  CLRA            null-terminate the name
        BSR     COPYC
        JSR     CKNUMB  check if numeric...
        BCC     RETNUM  yep, go return numeric form
        LDX     OBLIST  nope, scan the OBLIST for the symbolic atom
SCANLP  STX     LSTPTR  using LSTPTR to point to the elements of the list
        BEQ     NEWATM  end of list, must be new atom
        LDX     CAR,X   get next atom
        DEX             clear low-order bit of form
        LDX     CAR,X   get pointer to the atom's name
        BSR     CMPNAM  compare with that at NAMPTR
        BEQ     OLDATM  found it! return pre-existing atom
        LDX     LSTPTR  not equal, go on to next atom
        LDX     CDR,X
        BRA     SCANLP

* compare strings pointed to by X-reg and NAMPTR
* Z-bit set if equal
* clobbers X-reg, A-reg, B-reg
CMPNAM  STX     XTMP2   save X-reg temporarily
        LDX     NAMPTR  point at new atom name
CMPLP   LDAA    0,X     get next two chars
        LDAB    1,X
        STX     XTMP    save X-reg temporarily
        LDX     XTMP2   point at other name
        CMPA    0,X     first char equal?
        BNE     CMPRET  nope, return with Z-bit clear
        TSTA            all done now?
        BEQ     CMPRET  yep, return with Z-bit set
        CMPB    1,X     second char equal?
        BNE     CMPRET  nope, return with Z-bit clear
        TSTB            all done now?
        BEQ     CMPRET  yep, return with Z-bit set
        INX             match so far, advance pointers
        INX
        STX     XTMP2
        LDX     XTMP
        INX
        INX
        BRA     CMPLP     and loop around.
CMPRET  RTS             Z-bit now set properly.

* we have an old atom, reset spcptr and return
OLDATM  LDX     LSTPTR  get form specifying pre-existing atom
        LDX     CAR,X
        CPX     NILATM  is this the atom with the print name "NIL"
        BNE     RETNUM
        LDX     ZERO    yep, return a zero form instead
RETNUM  STX     FORM    save form temporarily
        LDX     NAMPTR  reset SPCPTR
        STX     SPCPTR
RATRET  LDX     FORM      and return with FORM in X-reg
        RTS

* we have a new atom, allocate a new cell and return new form
NEWATM  LDX     NAMPTR  set up new atom cell with pointer to name
        JSR     GETCEL    in CAR of dotted pair
        INX             set low order bit of form to indicate atom
        STX     FORM      save in FORM for later
        JSR     GETCEL  link new atom onto OBLIST
        LDAA    OBLIST
        STAA    CDR,X
        LDAA    OBLIST+1
        STAA    CDR+1,X
        BRA     RATRET    and return with FORM in X-reg

* here is a stripped down version of CKNUMB.  It accepts
*   only one-digit numeric atoms.  Otherwise it
*   returns with C-bit set.
CKNUMB  LDX     NAMPTR  look at name
        LDAA    0,X     get first char
        SUBA    #'0     subtract out ASCII code for zero digit
        BLT     BADNUM  not a digit if negative
        CMPA    #9      bigger than 9?
        BGT     BADNUM  not a digit
        TST     1,X     more than one digit?
        BNE     BADNUM  not allowed (for now)
        LDAB    #$80    A-OK, build up numeric form in FORM
        STAB    FORM
        RORA            shift value around so that bit 1 is left open
        ROLB              for garbage collector
        ASLA
        RORB
        ROLA            bits 1,2,3 --> 2,3,4; bit 0 --> bit 0
        STAA    FORM+1  store low order byte of form
        LDX     FORM      and return it in X-reg
        CLC               with C-bit clear.
        RTS
BADNUM  SEC             bad number, return with C-bit set.
        RTS
```

(a)
```
(DEF PRINT (LAMBDA (F)
    (PROGN
        (PRINR F)
        (PATOM NEWLINE)
        F
    )
))

(DEF PRINR (LAMBDA (F)
    (COND
        ((ATOM F) (PATOM F))
        (T (PROGN
            (PATOM LPAREN)
            (PRINR (CAR F))
            (PRINL (CDR F))
            (PATOM RPAREN)
        ))
    )
))

(DEF PRINL (LAMBDA (L)
    (COND
        ((DTPR L) (PROGN
            (PATOM SPACE)
            (PRINR (CAR L))
            (PRINL (CDR L))
        ))
    )
))
```

(b)
```
* type out a form, fully parenthesized, and then go to a new line.
PRINT   JSR   PUSHX    save X-reg on stack
        BSR   PRINR    simply pass the buck to recursive PRINR
        LDX   CRLFAT   type out CR/LF
        BSR   PATOM     using PATOM
        JMP   POPX     restore X-reg and return.

* type out a form, with no CR/LF
*  clobbers X-reg
PRINR   JSR   ISATOM   is the form an atom?
        BCC   PATOM    yes, pass the buck to PATOM
        JSR   PUSHX    nope, stack the X-reg
        LDX   LPARAT   type out a "("
        BSR   PATOM
PRINL   JSR   TOPX     restore the X-reg
        LDX   CAR,X    type out the CAR
        BSR   PRINR     (recursively!)
        JSR   POPX     restore pointer to the dotted pair
        LDX   CDR,X    advance to next dotted pair in linked list
        JSR   ISDTPR    is there a next dotted pair?
        BCS   PRPAR    nope, go type a ")"
        JSR   PUSHX    yep, save the new X-reg again
        LDX   SPACAT   type out a space
        BSR   PATOM
        BRA   PRINL     and loop around.
PRPAR   LDX   RPARAT   type out a ")"
        BRA   PATOM     and return (through PATOM).
```

*Listing 6: LISP and M6800 assembly code of the PRINT routine.*

*Text continued from page 142*
when encountered (eg: "(" and ")") without any separator characters necessary.

In the author's LISP system RATOM is relatively sophisticated, allowing for atoms with spaces in their names if they are quoted ("..."). Also the single quote character ("'") is given special significance, as are "[" and "]". However a simpler RATOM is quite enough for an initial implementation. To make this exposition simpler, only single digit numeric atoms will be allowed. Certainly in an eventual implementation, multidigit numeric atoms, optionally preceded by a minus sign would be accepted.

In this RATOM, the characters are copied into an area set aside to hold the names of atoms as they are input. A null character (ASCII code zero) is used to terminate the name, when a separator or special character is encountered. If the name is entirely numeric, then the atom is a numeric atom, and the form is simply the value of the number, with the high order bit set, and one other bit left zero for use in the garbage collector. Otherwise the atom is a symbolic atom, and a scan is made of the OBLIST for a pre-existing atom with the same name. If one is found, the characters just typed in are thrown away and a form specifying the pre-existing atom is returned. If the atom is a new one, a 4 byte cell is allocated (using GETCEL defined in listing 4) and a pointer to the new atom is added to the OBLIST. A form specifying the new atom is returned. The M6800 assembly language code for this is in listing 5.

## PRINT Function

PRINT is the second major recursive function comprising the LISP interpreter. It takes a single form as argument, and types the value as a fully parenthesized LISP expression. PRINT simply calls the more primitive function PATOM when it is given an atom to type. Otherwise, PRINT types a left parenthesis, calls itself recursively to type out the elements of the list, and then types a right parenthesis. In any case, PRINT always types out a carriage-return/line-feed at the end. This can be coded as in listing 6.

In the LISP routines, the special function PROGN is used. PROGN simply evaluates all of its arguments in sequence, and then returns the value of the last one as the value of the entire PROGN. The two functions ATOM and DTPR are used to test the type of a LISP object. ATOM returns T if the argument evaluates to an atom — symbolic, numeric, or NIL. Otherwise ATOM returns NIL. DTPR is the exact opposite. It returns T if the argument evaluates to a dotted pair, and returns NIL otherwise. Such functions which return either T or NIL are called "predicates" in LISP in analogy with predicates as used in symbolic logic. Such functions in other languages are called *Boolean* functions.

Nowhere in the routines for PRINT, nor for that matter in the routines given earlier for READ, is the allowance made for the input or output of list structures which require the use of "dot" notation. A structure like (A B C . D) could not be input, and the above PRINT routines would type it out as (A B C), simply assuming that the atom which ended the linked list was NIL. It turns out that the changes necessary to implement dot notation are quite straightforward. For example, to add it to the LISP version of PRINT, only the routine PRINL need be rewritten, as follows:

```
(DEF PRINL (LAMBDA (L)
    (COND
        ((DTPR L) (PROGN
            (PATOM SPACE)
            (PRINR (CAR L))
            (PRINL (CDR L))
        ))
        ((EQ L NIL) NIL)
        (T (PROGN
            (PATOM SPACE)
            (PATOM DOT)
            (PATOM SPACE)
            (PATOM L)
        ))
    )
))
```

A corresponding change could be made to the assembly language routines.

As with the primitive function RATOM, the function PATOM turns out to be more difficult to implement than the recursive PRINT. PATOM must distinguish between symbolic atoms, numeric atoms, and NIL, and act accordingly. With symbolic atoms, PATOM simply types the null-terminated name of the atom. With numeric atoms, PATOM must convert back from the internal representation of the numeric value, to the string of ASCII characters which represent the number. With NIL, PATOM simply types 'NIL'. Listing 7 is a simplified version of PATOM with numeric atoms of only a single digit.

### EVAL Function

The EVAL function is the *heart* of the LISP interpreter. EVAL accepts one form as an argument, and evaluates it according to the LISP convention: the value of NIL is NIL, the value of a numeric atom is itself, the value of a symbolic atom is the form associated with the atom, and the value of a list is determined by applying the function specified by the CAR of the list to the list of arguments which make up the CDR of the list.

In most LISP systems at least two distinct kinds of functions exist, SUBRs and LAMBDAs. SUBRs are the built-in functions of the LISP system, written in machine code (like CAR, CDR, PATOM). LAMBDAs are the user-defined functions, defined like (DEF GCD (LAMBDA (X Y) ...)). The effect of such a DEF is simply to define the list (LAMBDA (X Y) ...) as the value associated with the atom GCD.

The type of object used to specify a SUBR function varies among LISP systems. Frequently a new type of object is defined, called CODE, distinct from atoms and dotted pairs. A second alternative is to treat SUBRs like a funny kind of atom. The author's LISP system treats the bytes which make up the machine code of the SUBR like the print name of an atom. The SUBR is then specified by a dotted pair, with the CAR being the atom "SUBR" to identify the type of function, and the CDR being this atom with the funny print name. In fact the print name is prefixed with a special string which is unlikely to occur in a normal atom's print name, and hence PATOM could detect that the print name was not typeable, and simply type, say, "!" instead. In addition EVAL can check for the presence of this special string at the beginning of the print name to avoid treating a normal atom's print name as machine code. This method for specifying SUBRs avoids introducing an additional type, but the added complication in PATOM and EVAL may rule out the method in some implementations.

When EVAL is given a list to evaluate, it first evaluates the CAR of the list (recursively). The evaluation of the CAR should be either a LAMBDA expression, or a SUBR expression. If the evaluation of the CAR is an atom, or a list not headed by LAMBDA or SUBR, then EVAL stops, and indicates an error to the user.

If the CAR of the list gives a LAMBDA expression, the arguments to the function call are evaluated one at a time and saved on a list. The value associated with the "formal" arguments of the LAMBDA expression (eg: X and Y to the GCD routine given earlier) are saved on the stack. These formal arguments are then set one at a time to have the value of the corresponding actual arguments to the function (which were evaluated already). Finally, the "body" of the LAMBDA expression is evaluated, with the formal arguments now holding their new values. The result of evaluating the body is the result of the original function call. As a last step, EVAL restores the original values of the formal arguments.

Following the details of evaluation of such a function call is very difficult at first. The sequence of these steps is critical: evaluate actual arguments, save old values of formal arguments, set new values of formal arguments, evaluate body of LAMBDA, restore old values of formals. With any other sequence there is a chance that changes to the formal arguments of this function might interfere undesirably with the values of atoms in the calling routine's environment. These formal arguments are supposed to be strictly "local," that is, the choice of a name for a formal argument should be a strictly local decision, having no impact on variables with the same name in calling routines. Observing these rules allows LISP functions to be freely recursive. As the above examples of routines demonstrate, this recursion is in fact heavily used in LISP programming.

The steps in applying a SUBR function are simpler, because there are no formal arguments to worry about. EVAL simply evaluates the arguments to the SUBR, and passes them as a list to the machine code subroutine. EVAL expects the result of the SUBR to be left in register X when the subroutine returns.

This much of EVAL can be implemented on the M6800 as in listing 8.

The routines EVLALS, POPFRE, EVLNSV, EVLRSO, and EVLRST have not been included in listing 8 for brevity's sake. They are all relatively straightforward routines, making heavy use of GETCEL, PUSHX, POPX, and FRECEL to build up and then release the lists of saved values.

Two additional types of LISP functions, normally recognized by an EVAL function, are called NLAMBDAs and NSUBRs (or FSUBRs, or FEXPRs if you prefer). These types of functions take their argument lists unEVALed. NSUBRs are simply passed the CDR of the original function call list, instead of a list of evaluated arguments. Similarly, NLAMBDAs are provided with only a single argument, the list of unevaluated arguments. Without NSUBRs it is necessary for EVAL to recognize functions like COND as special cases, so that their argument list is not immediately evaluated. NSUBRs are specified in the same way as SUBRs, with the atom "NSUBR" replacing "SUBR" in the CAR of the dotted pair. PRINT will type out NSUBRs as "NSUBR .!)"

NLAMBDAs are very useful for creating elaborate user-defined functions which take argument lists that are as or more complicated than COND. NLAMBDAs are necessary anytime the number of arguments is variable, or some of the arguments are wanted unevaluated.

To incorporate NLAMBDAs and NSUBRs in the above

*Text continued on page 148*

*Listing 7: A simplified version of PATOM which assumes single digit atoms.*

```
* given atomic form in X-reg, type out name on terminal
*  preserves X-reg
PATOM   STX     FORM      save X-reg for later
        BEQ     PNILAT    go print 'NIL' if form is zero
        BLT     PNUMAT    go print numeric atom (high bit set)
        DEX               symbolic atom, clear low bit of form
        LDX     CAR,X     get pointer to print name of atom
        BRA     PNAME       and go type it out.
*
NILNAM  FCC     'NIL'     string to print for NIL form
        FCB     0          (the null terminator)
*
PNILAT  LDX     NILNAM    point to a null-terminated string "NIL"
PNAME   LDAA    0,X        type out the chars one at a time
        BEQ     PATDUN      until null char,
        JSR     PUTC         using the ROM monitor put char routine.
        INX               advance to next character of name
        BRA     PNAME      and loop around.
*
PATDUN  LDX     FORM      restore X-reg
        RTS                 and return.

* simplified version of numeric atom type-out
*  form already stored in FORM.  Only look at low order byte for now.
PNUMAT  LDAA    FORM+1    get low order byte, and shift it a bit
        LSRA               bits 2,3,4 --> 1,2,3
        ADCA    #0         bit 0 --> bit 0
        ADDA    #'0       add in ASCII code for zero
        CMPA    #'9       more than a single digit?
        BLE     PNM2
        LDAA    #'*       yes, simply type out "*" for now.
PNM2    JSR     PUTC       send digit to monitor put char routine
        BRA     PATDUN      and return from PATOM.
```

*Listing 8: A simplified version of EVAL.*

```
* evaluate form passed in X-reg, return result in X-reg.
EVAL    STX     FORM      save form temporarily
        BLE     NUMNIL    numeric atom or NIL, simply return
        LDAA    FORM+1    is this a symbolic atom?
        RORA                (rotate bit 0 into carry bit)
        BCC     EVLDTP    no (low bit clear), must be dotted pair
        DEX               yes (low bit set), load current value of atom
        LDX     CDR,X
        STX     FORM      store it back in FORM
NUMNIL  RTS                and return with value in X-reg and FORM.

* evaluate a function call (function specified by CAR of dotted pair)
EVLDTP  LDX     ALP       save value of ALP, FLP, NLP on stack
        JSR     PUSHX
        LDX     FLP
        JSR     PUSHX
        LDX     NLP
        JSR     PUSHX      (now ALP, FLP, and NLP are "local" variables)
        LDX     FORM      point back to original form
        STX     ALP       save it in ALP (recursive call on EVAL
*                           will clobber FORM)
        LDX     CAR,X     evaluate CAR to get LAMBDA or SUBR expression
        BSR     EVAL        (recursively!)
        JSR     ISATOM    is result an atom?
        BCC     EVLERR    yes, illegal expression!
* X-reg should now be pointing to either
*  (LAMBDA (...) ...)
* or
*  (SUBR . funny-atom)
        STX     FLP       save pointer to function def
        LDX     CAR,X     now check the CAR for LAMBDA or SUBR
        CPX     SUBRAT    SUBR?
        BEQ     EVLSUB    yes, go evaluate args, etc.
        CPX     LAMBAT    LAMBDA?
        BEQ     EVLLAM    yes, go apply the LAMBDA
* illegal expression to EVAL, go report error to user
EVLERR  LDX     #ERRMSG   error break routine expects error message
        JSR     ERRBRK     pointed to by X-reg.
* normal or error return, restore ALP, FLP, and NLP
*  return with result of evaluation in X-reg and FORM
EVLXIT  STX     FORM      save return value
        JSR     POPX      restore NLP, FLP, ALP
        STX     NLP
        JSR     POPX
        STX     FLP
        JSR     POPX
        STX     ALP
        LDX     FORM        and return with result in X-reg
        RTS                  and FORM.

* evaluate the arg list, call the machine code subroutine.
EVLSUB  JSR     EVLALS    EVAL list of args (pointer to it in ALP)
        LDX     FLP       point at CDR of function def
        LDX     CDR,X
        BLE     EVLERR    error if numeric or NIL atom
        JSR     ISATOM
        BCS     EVLERR    error if not atom at all
        DEX               point to print name of funny atom
        LDX     CAR,X
        LDAA    0,X       check for presence of special string
        CMPA    #$21       (hex $21,$00 used on this system)
        BNE     EVLERR
        LDAA    1,X
        CMPA    #$00
        BNE     EVLERR
        JSR     2,X       SUBR looks OK, call subroutine
        STX     FORM      save returned result
        JSR     POPALS    free up list of EVALed args
        LDX     FORM       restore X-reg
        BRA     EVLXIT      and return from EVAL.

* evaluate call on (LAMBDA (J K) ...)
EVLLAM  JSR     EVLALS    evaluate list of args
        JSR     EVLNSV    save old values of formals on stack
        JSR     EVLRSO    store new values for formals
        LDX     FLP       point to body of LAMBDA
        LDX     CAR,X
        JSR     EVAL      evaluate body (recursively!)
        STX     FLP       save result form temporarily
        JSR     EVLRST    restore old values of formals
        LDX     FLP        restore X-reg
        BRA     EVLXIT      and return from EVAL.
```

EVAL routines, two additional checks must be added immediately prior to EVLERR:

```
        .
        .
        .
      BEQ EVLLAM      ...
      CPX NSUBAT      NSUBR?
      BEQ EVLNSU      yes, go call machine code
                      subroutine
      CPX NLAMAT      NLAMBDA?
      BEQ EVLNLA      yes, pass list of args as single
                      argument
* illegal exp...
EVLERR
        .
        .
        .
```

and the additional routines EVLNSU and EVLNLA must be included. Both of these routines are simpler than the corresponding routines EVLSUB and EVLLAM.

To make EVAL useful, some number of built-in SUBRs and NSUBRs must be written. The number of such built-in primitives can be kept quite small in LISP if they are chosen carefully. Most routines can be implemented as user functions if a few primitives exist. The primitives will certainly include PATOM, RATOM, EVAL, CAR, CDR, CONS, COND, SET, ADD, SUB, EQ, GREATER, ATOM, and NUMBER. All but SET and NUMBER have been used in the LISP function listings. SET is the primitive LISP assignment function. SET takes an atom and a value, and sets the value associated with the given atom to be the given value. NUMBER is a predicate function like ATOM, and simply returns T when its argument is a numeric atom. Listing 9 is an example of one of these primitives, the SUBR EQ.

Notice that the SUBRs and NSUBRs will start with the preface string (hex 21, 00 is used in this system). The argument list is always pointed to by ALP. Also notice that the SUBR may not assume that the proper number of arguments were supplied. The general rule is to treat

```
* two argument SUBR EQ
*  return T if given identical forms, NIL otherwise
EQSBR   FCB     $21      special preface string
        FCB     $00
* ALP points to the list of evaluated arguments
        LDX     ALP      get first arg
        BEQ     TRUE     no args is equivalent to
                         (EQ NIL NIL)
*                         which should return T.
        LDX     CAR,X    save first arg temporarily
        STX     XTMP
        LDX     ALP      pick up second arg
        LDX     CDR,X
        BEQ     EQSNIL   (EQ X) is equivalent to
                         (EQ X NIL)
        LDX     CAR,X
EQSNIL  CPX     XTMP     are the forms identical?
        BEQ     TRUE     yes, return T.
        LDX     ZERO     no, return the NIL form
        RTS
*
TRUE    LDX     TATOM    return T atom
        RTS
```

*Listing 9: EVAL may have built in primitives to expand the language. This is an example of the primitive SUBR EQ.*

unspecified arguments as though they were NIL. In EQ above, this gives some rather strange behavior, where simply (EQ) will always return T. It still remains for the implementor to initialize the atom EQ to point to a dotted pair, (SUBR . funny-atom), with the print name of the funny atom set to point to the code at EQSBR as shown in listing 9. The final section of this article goes over some of the problems involved with this kind of initialization.

## Garbage Collector

A *garbage collector* eventually becomes essential in any LISP system. It is possible to create dotted pairs that are no longer accessible to a LISP program by any path. This happens, for example, if a function like REPLACE is called and then the value returned simply PRINTed but not saved as a LISP atom. This cannot go on for long before all of the free space is used up with dotted pairs. The garbage collector's job is to find all of the dotted pairs.

The various algorithms for locating such jetsom of the LISP function evaluation process are all quite intricate. The basic idea is always to trace systematically down every list structure to its component atoms, marking every dotted pair encountered along the way. If a dotted pair is encountered which is already marked, then that branch of the list structure is assumed to be already fully traced. The garbage collector then makes a sequential scan of all of memory space occupied by dotted pairs, and links together all unmarked dotted pairs onto a special list, the *free* list. During the scan, the marked dotted pairs are simply skipped over, because they are assumed to still be a part of some useful list structure. When a marked dotted pair is skipped over, its mark is also cleared in anticipation of future garbage collections, when it might no longer be so lucky.

The difficulty with this trace and collect algorithm is that each dotted pair points to possibly two more dotted pairs, so during the tracing phase the garbage collector must eventually follow both paths. What this means is that a second indication must be made on each dotted pair, indicating that the garbage collector is now busy tracing the CAR of this dotted pair, and will be returning later to trace the CDR of the dotted pair.

During the tracing phase, the garbage collector might very well be thought of as an ant determined to visit every branch of a tree. It goes out to the tip of each branch, but as it returns it must remember whether it has already traversed the other paths going out from each branching point. Even this analogy underrepresents the difficulty of a garbage collector, because the ant can simply turn around when it reaches the tip of a branch, but the garbage collector would normally have no clue as to how to climb back toward the root of a list structure once it gets out on a distant dotted pair.

The solution to the garbage collector's problem is to either reverse all the pointers in the list structure as it forays out to the terminating atoms and then reset the pointers on the way back in, or to keep a list of all dotted pairs which still require that their CDRs be traced. The first solution is like stringing a spool of thread behind you as you venture into an unexplored cave, following the thread back toward the mouth of the cave when you reach a dead end. Of course the same danger exists; that

The garbage collector may run at any moment.

the delicate thread leading you back to the starting point might get tangled or broken.

The second solution is simpler, but suffers from the grave problem that it requires room to store the list of partially visited dotted pairs, and garbage collectors tend to be called upon at times when there is no more room to spare. In fact, the list of partially visited pairs need get no longer than the maximum "depth" of any list structure in the system, so that by setting aside a small portion of memory reserved for the use of the garbage collector's list, the implementor can get by with coding a much simpler tracing algorithm.

The author's system uses the pointer reversal method, and he will testify to the unlimited number of obscure problems which can appear during the debugging phase of its implementation.

It should be clear now why it was important to leave one bit in each form, and hence two bits per dotted pair, free for the use of the garbage collector. The bit in the CAR form can be used to indicate that the dotted pair has been visited once, and the bit in the CDR can be used to indicate that both paths from the dotted pair have been traced. These bits are only used during garbage collection, but because the garbage collector may be called at any time when GETCEL finds that there are no more 4 byte cells on the free list it may, in fact, run at almost any moment.

Because of this unpredictability, a LISP system with a garbage collector must be coded "defensively," jealously protecting any dotted pair allocated but not yet added to some accessible list structure. The machine code routines given in the listings do not all adhere to this rule. The reason for ignoring the garbage collector in the development thus far was simply to keep the design of the routines simple and relatively intuitive.

If the reader intends to include a garbage collector in an implementation of a LISP interpreter, more care must be taken. For example, two versions of the routine PUSHX would be defined, normal PUSHX and PROPSH (protected push). The PROPSH would be used when the 16 bit value being pushed on the stack pointed to list structure which might not be accessible in any other way, and hence might get collected in the next garbage collection scan. PROPSH avoids this danger by marking the cell used to store the saved value so that the garbage collector will know to trace this form and its descendents.

### Initialization

It is ironic, but somehow appropriate, that the section on initialization comes at the end of this article. Frequently it is in fact one of the last things an implementor thinks about. That is probably because initialization is one of the biggest difficulties facing the implementor of any language: assembler, interpreter, or compiler. By initialization is meant the inevitably awkward methods of getting the symbol tables, or the OBLIST in LISP preloaded with the names which are to be built-in to the system. Most of the routines written to enter symbols into symbol tables, or to add new atoms to the OBLIST, are all oriented toward names entered by the *user* of the language processor. The initialization phase of the system becomes quite complicated because of this orientation. The methods finally chosen are, in general, tedious, requiring a lot of special preparation by the writer of the intialization routine.

The best way to avoid these initialization difficulties is to spend a little extra effort in designing a few nice routines for taking information out of tables which are convenient for the *implementor* to set up and modify, and let these routines do the intricate bit-twiddling work necessary to get the objects in shape for the symbol table, or the OBLIST.

In the author's LISP initialization module are routines to build up dotted pairs in the form required for SUBRs and NSUBRs, and routines to allocate 4 byte cells for built-in atoms. The atom initialization routines are given the address of a contiguous table of null-terminated ASCII names, each followed by the address of a memory cell where the form specifying the new atom should be stored. This is where the symbols like TATOM, SUBRAT, LAMBAT, etc came from. They refer to memory locations in the base page of the M6800 (0 thru 255), where the forms specifying the atom T, SUBR, and LAMBDA, etc, are stored. The table to initialize these atoms was simply:

```
ATMTAB FCC    'T'
       FCB    0
       FDB    TATOM
       FCC    'SUBR'
       FCB    0
       FDB    SUBRAT
       FCC    'LAMBDA'
       FCB    0
       FDB    LAMBAT
       FCC    . . .
        .
        .
        .
       FDB    . . .
       FCB    0      null-name terminates table
```

Although writing the special initialization routines was initially time-consuming, it was more than compensated for by the ease of adding more built-in atoms as the system grew.

## Conclusion

We have traced through the implementation of a LISP interpreter and looked at a specific example for the M6800 processor. For further information on the garbage collecting routines and a complete listing of the interpreter, order BYTE document number 112.■

## Nybbles

*The Nybbles Library is an inexpensive means for BYTE readers to share some interesting but specialized forms of software. These programs are written by readers with small computers and printer facilities, and are therefore designed for particular systems. The algorithms and programming techniques in these programs can be directly used by readers with similar equipment, or can serve as an inspiration for improvisation on computers of different characteristics.*

*Potential authors for such programs should send us a self-addressed stamped envelope, with a request for a copy of our "Guideline for Nybbles Authors." Payment for Nybbles items is based on sales and length of the item. Rates are set when author's proofs are returned.*

*Nybbles Library programs are sent in listing form, printed on 8.5 by 11 inch, three hole punched paper for collection in loose leaf binders.*

*This month "An M6800 LISP Interpreter" has been added to the Nybbles Library. To order your personal copy at $10.00 (US and Canada), $12.00 (foreign airmail) postpaid, fill out the coupon below.*

Please send ________ copies of BYTE Nybble #__________ at $__________ postpaid.

Please remit in US funds only.

__________ *Check Enclosed*

*Bill my VISA #* ______________________ *Exp Date* ______________

*Bill my MC #* ______________________ *Exp Date* ______________

Name ____________________________________________

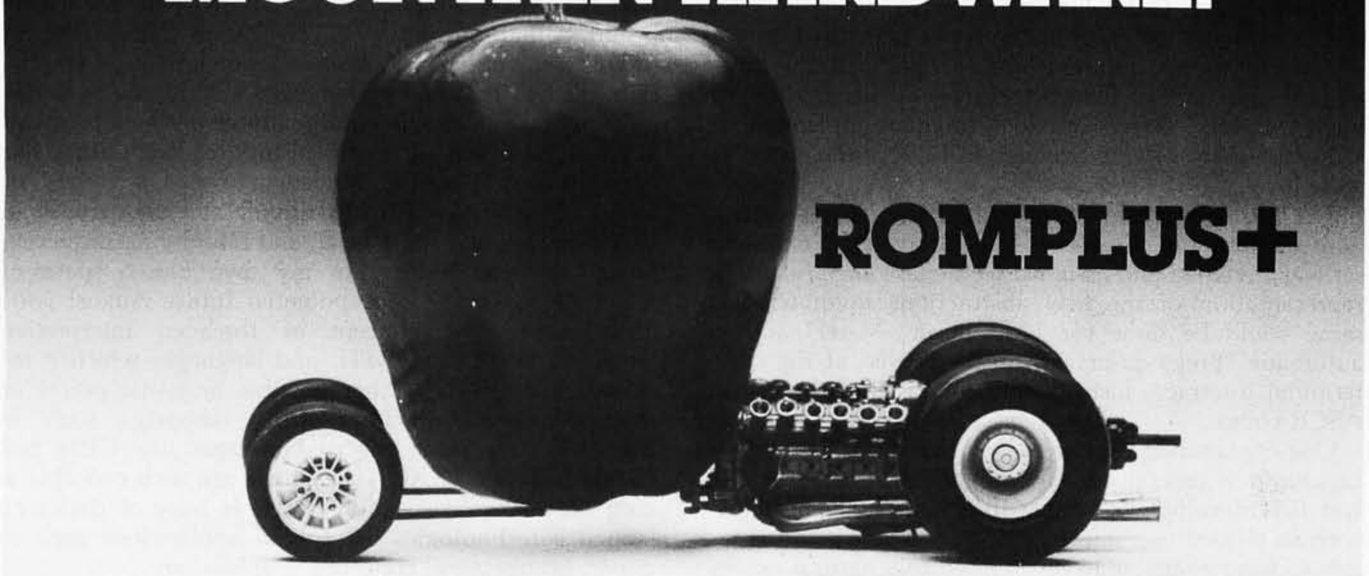
Street ____________________________________________

City ______________________ State ______________ Zip Code ______________

BYTE Nybbles Library, 70 Main St, Peterborough NH 03458

You may photocopy this page if you wish to keep your BYTE intact.

# NEW POWER FOR YOUR APPLE II FROM MOUNTAIN HARDWARE.




**NEW EXPANDABILITY.**
ROMPLUS+ is a peripheral board whose added features can turn the Apple II* into the most powerful personal computer available today.

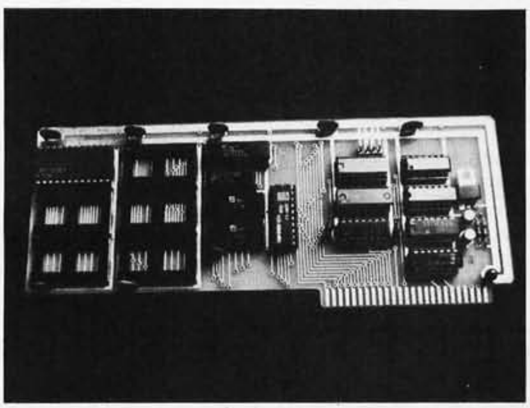
**NEW POWER.**
ROMPLUS+ provides six sockets to accept individually addressable 2K ROM's or EPROM's. Keyboard Filter™, a 2K ROM program, comes installed on the ROMPLUS+ board and adds many useful features to your Apple II, including:

- Upper and lower case letters. The only system that offers keyboard input and standard shift key operation.
- Multiple user-defined character sets.
- Colored or inverse-colored letters.
- Keyboard macros—two key-stroke, automatic typing of multiple, user-defined words or phrases. Including BASIC and DOS commands.
- Mixed text and graphics.
- Improved cursor control.
- STOP LIST and END LIST.
- Works with Integer BASIC, RAM or ROM Applesoft, and DOS.
- And more...quick to learn. Easy to use.
- Software support provided on disk includes demonstration programs and two Editors that allow you to define your own characters or keyboard macros.

**SOPHISTICATED FIRMWARE.**
In addition to the Keyboard Filter ROM, ROMPLUS+ offers five sockets for ROM or EPROM, plus "scratch-pad" RAM. And, sophisticated firmware on ROMPLUS+ allows one, two, or more of its chips to be used simultaneously for programs longer than 2K.

**EXPANDED UTILITY.**
Many software programs really ought to be utilized as firmware. ROMPLUS+ makes that an actuality for the Apple II by providing six additional ROM sockets.

**AT YOUR DEALER. NOW.**
$169. Complete and tested. Including the powerful Keyboard Filter ROM and full documentation. Ask your dealer for a demonstration.



**Mountain Hardware, Inc.**
LEADERSHIP IN COMPUTER PERIPHERALS
300 Harvey West Blvd., Santa Cruz, CA 95060
(408) 429-8600

Sounds great.
☐ Please send me all the details on ROMPLUS+ and Keyboard Filter.

Name ______________________

Address ______________________

City ____________ State ____________ Zip ______

Phone ______________________

*Apple II is a trademark of Apple Computer, Inc.

*Text continued from page 8:*

Gary also wanted readers to understand that the LISP of the "Lots of Infernal Stupid Parentheses" does not represent the essential beauty of this approach. This relatively awkward notation is the assembly language of a LISP machine. It suffers from all of the disadvantages of assembly languages. Relatively simple to program, this "S-expression" form of LISP notation is one that is most often implemented, and it tends to give a distorted view of the language. Gary wanted readers to understand that the alternative "M-expression" form of LISP, with special characters noting relationships, is perhaps the most elegant and natural form of expression for many problems. Rarely, however, does anyone implement an "M-expression" oriented version of LISP at the user software level.

The problem is similar to that of the language APL, with one notable difference. In APL a special character set was invented and assigned to the language for use in representation of the new abstractions involved. The same could be done for LISP if an "M-set" and an automatic "pretty-printer" were employed at the user's terminal interface, instead of a lot of parentheses and ASCII codes.

One explanation for the reason that LISP has not yet caught on as generally as APL might be the fact that APL was first developed on large IBM computer equipment with an elegant user interface. IBM Selectric printing terminals were available to be adapted to a natural expression APL via the "APL-ball," while LISP was seldom used with IBM equipment during its period of development as a tool. With today's technology of personal computer graphics, the same principle can be adapted to the user interfaces of LISP software. The best LISP packages for personal computers should incorporate an appropriate display philosophy which allows the elegance of the language to shine.

[While on this subject of "today's technology," we have also heard some exciting words about a computer system design from the Laboratory for Computer Science at MIT. This is only an advance hint of what may come. The machine is described as an experimental computer with a very high resolution (1024 point) black and white display, 32 bit internal architecture, an advanced LSI processor such as Z-8000 or 68,000, gobs of memory implemented with 65,536 (64 K) bit parts, and an advanced operating system. As a commercial product it may be available in 12 to 24 months in a price range of about $5000. The word I have from its designer, Steve Ward, is that the technology has been transferred by license to a commercial firm which has existing interests in personal computing products. MIT's motivation with respect to having a commercially manufactured version is to be able to buy several hundred of the machines for local use in its technological community. We may have thought that the past two years were exciting, but the field has hardly begun its maturation...]

This series of BYTE August issues on languages emphasizes the fact that no one language will optimally satisfy all uses. Just as people continually create new forms of expression in any art, the history of computing has demonstrated a similar tendency toward a variety of forms of expression for algorithmic and data concepts. Our coverage of APL, Pascal, and LISP by no means exhausts the possibilities. In my own biased space of language concepts, I see potential future August issue attention to the concept of threaded interpretive languages such as FORTH, and languages which it inspired, like URTH. Other possible linguistic points of discussion might include string languages such as SNOBOL, and even macro languages like GPM and Calvin Mooers' TRAC. Then there are such concepts as data base languages, and the whole issue of designing language technologies for special applications such as music, architectural concepts, graphics, etc.

The fundamental point of this essay still remains: no one language will optimally satisfy all the needs of all users. Some people care only about quick implementation and debugging, and do not really care about speed. Some people just like one particular style of expression. Some people think literally in tree forms and have to strain to think in sequential processing forms. To the extent that programming concepts are universal, choice of a language is often a matter of personal aesthetics. And where languages go off in one or more partially or wholly orthogonal conceptual directions, then the choice of language is based upon the underlying uses of the tool. (Fuel for a number of heated arguments is present in the determination of just what *is* an orthogonal conceptual direction.)

While on the subject of different languages and choices of tools, I should mention one of the most exciting items seen at the recent West Coast Computer Faire. This item

is a whole new language for interaction with computers at an intellectual user's level. As a tool for use with computers, this language-like method of structuring an interface is completely orthogonal to any conventional sequential language from BASIC to Pascal, although its conceptual underpinnings are very LISP-like. The product has roots in the artificial intelligence community, and it is a direct result of the programming efforts of two gentlemen with strong ties to the MIT computer science scene, Dan Bricklin and Bob Frankston. It is presently available on the Apple-II computer, and will soon be available on Atari, Pet and TRS-80 computers.

Dan and Bob have formed a firm of their own called Software Arts Inc. Their only customer is Personal Software, a company formed last year by another graduate of the Cambridge computer scene, Dan Fylstra (along with Peter Jennings of Microchess fame). The Personal Software company distributes this new product exclusively, at retail cost and through manufacturers. The product is called "Visi-Calc." The first "public" showing of Visi-Calc occurred last May in the form of a hospitality suite at the Fourth West Coast Computer Faire in the San Francisco Convention Center. The display was oriented to dealers and manufacturers. Advertisements have appeared earlier this year, and we should see more detailed publicity by the time of this issue.

As an interactive screen oriented piece of software, Visi-Calc makes the memory of the computer a logical "blackboard" where data is remembered along with relationships. This last phrase, "along with relationships," is the key element of the concept. When I record some number, eg: 3.1415927, at an intersection of the blackboard's coordinate grid named [B:32], that number is stored in that location on the blackboard.

Since available memory is much larger than the visible screen, I can use cursor controls to make my display window examine any portion of the total blackboard. I might move to location [A:12] and write the value of some angle, perhaps 0.33 radians. I can refer to other locations in defining a relationship for some location instead of raw data.

Suppose, then, that I put the relationship:

"SIN([A:12] * [B:32]) — COS (— [A:12] * [B:32])"

in location [Z:21]. Location [Z:21] now depends upon locations [A:12] and [B:32]. I can then move the cursor back to [A:12] and put in any angle that I like, for example 1.2. On changing any such location, Visi-Calc automatically searches the tree of dependent expressions and evaluates new data for such locations. The dependency can effectively go through many levels of calculation so that we can look at any intermediate stage of a calculation by noting it on the blackboard. When I return to location [Z:21] with the cursor controls, I will find the results of the [Z:21] expression as calculated with the new data. All pointing is done via cursor movements, so for the most part users never even refer to the "[letter:number]" coordinates of places on the blackboard.

The same technique can be applied to many programming tasks of an ad hoc nature; for personal, business, engineering and scientific applications. The software handles strings as well as arithmetic data and includes a full set of engineering and scientific functions such as the transcendentals used in the above example. Visi-Calc has to be one of the neatest software innovations of 1979, if not the most fundamental new concept to date in the personal computing field. It will certainly be used as a practical piece of software by many of our readers with various mass-marketed small computers.

An interesting comment was noted by authors Bricklin and Frankston and relayed in a recent conversation with Dan Fylstra of Personal Software. The comment was that the techniques used in Visi-Calc are possible only when a full processor is totally available to one user as a personal computer. The calculational bandwidth required to support this sort of technique is impossible to find at reasonable cost in a traditional large computer time sharing system. It only works when the concept of "one user, one processor" is employed, ie: when the computer power is "personal." As part of Visi-Calc's authors' experiences at MIT over the past decade, they often had this kind of relationship with traditional main frame computers like PDP-10's and IBM 370's. Such excessively expensive computing power devoted to one user is not possible outside of a research context. With the coming of the current age of microcomputing however, the personal (one user, one processor) approach is possible on a wider and less expensive scale. The products that are now available in this market for under $3000 are getting very close to the level of power which was restricted to research laboratories. Software products like Visi-Calc take advantage of this.


*"...and this one was owned by a little old lady who only used it once a month to balance her checkbook."*



Returning to the LISP theme of our current issue, Visi-Calc is an example of a tree-oriented parallel data structuring problem for which LISP is a most appropriate language of expression. Due to a lack of availability of LISP as a software development tool for personal computing hardware, its authors did not use LISP. They also had to make a number of compromises and tradeoffs as a result of the small size (eg: 16 K to 48 K bytes) of the main memory of personal computers. But they did use many of the tree concepts of artificial intelligence research. This provides us with the ultimate example of the relevance of LISP-like languages and approaches to personal computing: one of the most generally useful new user software tools for small machines, Visi-Calc, tackles just the sort of problem for which LISP is an appropriate tool of expression.

# Notes on the Appearance of BYTE...

*by Carl Helmers*

Readers will notice a number of changes in the appearance of the design layout of BYTE, starting with this issue. These changes are the cumulative result of several trends in our organization.

Perhaps the biggest such trend, from our readers' point of view, is the arrival of a form of computerized typesetting for BYTE magazine. I have often felt during the four years since BYTE started that we have been like the proverbial shoemaker's children who went barefoot. We have been producing a computer magazine without the benefit of any computer technology in the actual operation of our business. My own personal recovery from this situation occurred last fall when I began using a machine capable of running UCSD Pascal for all of my programming and writing. At about the same time, we were able to specify and order a computerized system of typesetting and page layout produced by Compugraphic. With this August 1979 issue, approximately 80% of the copy for the magazine was produced using the Compugraphic system. (Of course this measure is exclusive of advertisements which are generally prepared in final form by advertising agencies.)

The new magazine layout beginning in this issue was designed by Ellen Bingham and Nancy Estle of our production department. One of its major features is the use of symmetrical page layouts employing 2, 3, and 4 column widths on a page, depending upon the demands of subject matter and placement in the magazine. In the old layout, an asymmetrical two and a half column format wasted a lot of blank space. It also greatly complicated the production department's magazine layout design task each month. Since article pages in the old format were committed to either a right or left-hand side of an open magazine, the relative placement of pages became quite involved, sometimes even requiring last minute modification of "final pasted" pages to switch them from left to right-hand asymmetry!

The new format, aside from freeing up placement in the magazine, also allows more information to be placed on each page. It simplifies the problem of embedded equations or examples since the column width is greater in the two or three columns used for articles. When an article includes many long examples and equations, these will often fit on one line in the two column format, making the result easier to read. When an article does not have a large proportion of such embedded illustrations, the three column variant is available for use by our designers.

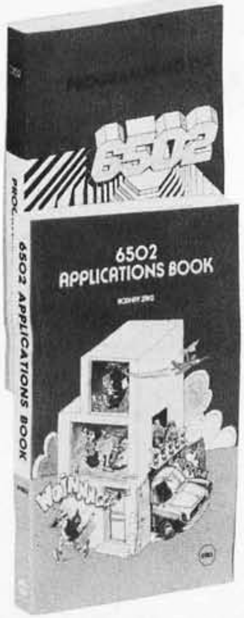
One question that we are frequently asked is related to magazine layout: Why do certain articles get split into sections, with portions of text continued at the back of the magazine? One reason for this is the use of color in the magazine. Approximately half of each issue is printed in color. Color pages are printed in groups of sixteen, called forms. It is sometimes necessary to begin two color articles in the same form, continuing one of the articles in another location in the issue. The relative length of articles also plays a part in how they are laid out in the magazine. We make every effort to keep each article in one contiguous piece whenever possible.

Speaking of computers for magazine production, we hope eventually to be able to accept articles from authors on floppy disks, using either the CP/M or Pascal format on full-size floppy disks. This means 8 inch single or double density, IBM compatible; for nonstandard information formats, documentation sufficient for conversion would have to be included. We will report on this subject as matters progress.

Changing the format of a magazine requires months of preparation and hard work. We want to reassure our readers that we plan to keep the content of BYTE just as it is. The new typeface, new column layouts, and updated feature pages are designed with you in mind. We would appreciate your comments and suggestions.

# Coming Up in BYTE...

With next month's September issue of BYTE, we begin our fifth year of publication. Returning to the genesis of personal computers in the hands of inveterate hackers, the theme of that issue is "homebrewing." In future issues we will see such special interest theme topics as education and computers, "domesticated computers," music, data bases, and a special theme on computer games of the Adventure/Dungeons and Dragons variety. Other topics we are contemplating for the coming year include continued attention to themes of voice input and output, graphics, languages, artificial intelligence and robotics...**CH**




# LEARNING

**6502 APPLICATIONS BOOK**
**by Rodnay Zaks**

☐ 6502 microcomputer board users, here's your book! It is the key to putting your KIM, Sym, or Aim65 to work. The 6500 family I/O chips are covered in depth (6520, 6522, 6532, and 6530) along with the interface techniques needed to use them. Applications examples include displays, computer music, alarms, and A/D conversion. Standard peripheral interfacing is explained. The appendices contain applications programs and a 6502 assembler in BASIC. The only prerequisite is some familiarity with 6502 assembly language programming. This is the definitive 6502 "input/output" book. Get it!

NEW!

284 pp. $12.95

**PROGRAMMING THE 6502**
**by Rodnay Zaks**

☐ Here is the 6502 microprocessor from top to bottom: basic concepts, hardware organization, basic programming techniques, the 6502 instruction set, addressing techniques, input-output techniques, data structures and program development. This book is a systematic course in 6502 assembly language programming, including excercises and application examples. Solid for you KIM and Apple users!

304 pp. $10.95


# LIVING

**FUNDAMENTALS OF RECORDKEEPING AND FINANCE FOR THE SMALL BUSINESS**
**by Robert C. Ragan, CPA, And Jack Zwick, Ph.D.**

☐ Once you have your organization or business up and running, records must be kept. What should I keep, and how do I record them? This book on fundamentals will give you a helpful start. Section one deals with maintaining records, protecting assets, and providing a basis for planning. Section two provides a starting point for owner-managers who want to sharpen their financial management skills.

196 pp. Hardcover $10.00

**UP YOUR OWN ORGANIZATION!**
**by Donald Dible**

☐ A great handbook on how to start and finance a new business or get an existing business out of neutral and into high gear. Written for the person willing to work hard at being successful in business, it is intended to save time, money and possible heartache by eliminating false starts and "blind alley" approches where appropriate. All drawn from the author's extensive experience in entrepreneurial business. This is the most comprehensive reference we've seen on the subject. For the programmer-consultant or the basement homebrewer-turned-entrepreneur, this is your book. It is recommended in the Bank of America Small Business Reporter and Changing Times magazines.

372 pp. Hardcover $14.95

Just Arrived!

**RUNNING WILD**
**THE NEXT INDUSTRIAL REVOLUTION**
**by Adam Osborne**

☐ Many of the uses to which computers are being put today could cause profound problems tomorrow. In industry, government, medicine and commerce, the full effects of computer technology -- the next industrial revolution -- have not yet been realized. That is why **Running Wild** has been written. Dr. Adam Osborne, a well-known writer and consultant in the world of microelectronics, has written this book for the layman. Its purpose is to confront the reader with what is going on today, and what will certainly be happening tomorrow.

**NEW FROM OSBORNE AND ASSOCIATES** $3.95

**WHAT EVERYONE SHOULD KNOW ABOUT PATENTS, TRADEMARKS AND COPYRIGHTS**
**by Donald M. Dible**

NEW!

☐ So you've written a program that's of use to microcomputerists everywhere? Got a circuit the world's waiting for. The next crucial step is to secure rewards for your ideas and work. This book is a distillation of the latest information available covering trademarks, copyrights and patents. Each topic is covered in a simple stand-alone style designed to communicate all of the essential points with a minimum of legalistic jargon. As an entrepreneur you should be familiar with your rights and responsibilities under these laws.

254pp. Hardcover $10.00

**BITS**™ inc

**DIAL YOUR BANK CARD ORDERS TOLL FREE 800-258-5477**

Books to erase the impossible

25 Route 101 West, PO Box 428, Peterborough, NH 03458

Name ______________________

Address ______________________

City__________ State__________ Zip______

☐ Check enclosed ☐ Master Charge ☐ VISA

Card#__________ Expires__________

Signature ______________________

| QTY. | TITLE | PRICE | TOTAL |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| POSTAGE & HANDLING | $ .75 for each book domestic<br>$1.00 for each book foreign<br>(maximum postal charge - $3.00) | P & H | |
| | | Total | |

(in N.H. dial 924-3355) MON-FRI 9-5

1B1089

You may photocopy this page

Dealer inquiries invited

# A Mathematician's View of LISP

Vaughan R Pratt
Assoc Prof of Computer Science
and Engineering
MIT Laboratory for Computer Science
545 Technology Sq
Cambridge MA 02139

All higher order languages offer the programmer mechanisms for simplifying and clarifying programs. Viewed from the distance that mathematicians such as myself prefer, away from the distractions of detail, LISP stands out as the first language to pay serious attention to the following issues:

- Mobility of data.
- Modularity of function.
- Declarative programming.
- Metalinguistics (the ability of a language to talk about language).

Since the development of LISP, two other languages (APL and, to a lesser extent, SNOBOL) have joined LISP in dealing with at least some of these issues. As such, one would assume that they would have improved on LISP. I believe that LISP outclasses these languages despite its having been developed earlier. Other languages, such as FORTRAN, BASIC, ALGOL, PL/I, and Pascal (or FBAPP as Professor Alan Perlis of Yale University refers to them collectively) are, in Perlis' opinion and mine, not in the same class as LISP and APL with respect to the issues discussed here. (I do not know Professor Perlis' opinion of SNOBOL.)

## Mobility of Data

In a computer, data flows between three major classes of sites: storage, functions, and devices. Storage consists of registers and main memory in assembly language, and variables (simple and subscripted) in higher level languages. Functions (or procedures, or subroutines) are quite alike in all languages, though with minor technical distinctions. Typical devices are printers, keyboards, floppy disks, paper tape readers, and the like.

The corresponding mechanisms available to the programmer for expediting this flow of data are fetch and store instructions, parameter passing and value returning constructs, and read and write commands.

A *mobile* datum is one which can be moved from one site to another by the program with a minimum of fuss. Here are two tests for mobility of data:

*Width test.* Must the data be moved piecemeal? For example, on your microprocessor, can you move a 2 byte address around as a unit, or do you have to move each byte separately? In your favorite language, can you read in an array from floppy disk or paper tape using one instruction, or must you write a loop to read the array elements individually?

*Length test.* Are intermediate sites needed to get data from one site to another? For example, to take the logarithm of a number that the user types in from a keyboard, do you have to store the number in a variable first and then take its logarithm, or can you just say (LOG (READ)) as in LISP?

If the data type fails either test it is not fully mobile. Note that if it fails both, the effect can be multiplicative. For instance, moving three bytes with each requiring two steps, requires six steps altogether.

It is often possible to enhance the mobility of data by writing the appropriate subroutines. For example you might write a routine to read an array from a device. This observation shows that mobility is a concept that is relative both to the available programming language constructs and to the available software.

*Promised* mobility is the possibility of writing such subroutines. Promised mobility is not as good as real mobility, as it requires the programmer to do the work of supplying the mobility, which may be more effort than it is worth for the particular application the programmer has in mind.

One basis for classifying programming languages is the mobility of their data types in the absence of additional subroutines such as the above mentioned one for reading

**About the Author:**

*Vaughan Pratt joined the MIT faculty in 1972 in the Department of Electrical Engineering and Computer Science and is associated with the Laboratory for Computer Science and Artificial Intelligence Laboratory. He received his PhD under Donald Knuth at Stanford University (Shell Sort and Sorting Networks). He is currently the head of the Theory of Computation Section at the Laboratory for Computer Science. His work includes natural language, algorithms, program semantics, and verification. His hobbies include collecting, repairing, and playing musical instruments and building robots.*

# RADIO SHACK COMPUTER OWNERS

# TRS-80™ MONTHLY NEWSLETTER

- PRACTICAL APPLICATIONS
- BUSINESS
- GAMBLING • GAMES
- EDUCATION
- PERSONAL FINANCE
- BEGINNER'S CORNER
- NEW PRODUCTS
- SOFTWARE EXCHANGE
- MARKET PLACE
- QUESTIONS AND ANSWERS
- PROGRAM PRINTOUTS

. . . . AND MORE

FREE

**WORD PROCESSING PROGRAM (Cassette or Disk)**

**For Writing Letters, Text, Mailing Lists, Files, Etc.**

**With Each New Subscriptions Or Renewal**

Send for **FREE** Software Catalogue (Including listings of hundreds of TRS-80 programs available on cassette and diskette).

COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™
Box 149 New City, New York 10956

master charge

24 HOUR ORDER LINE
(914) 425-1535

VISA

ONE YEAR SUBSCRIPTION $24 ____________

TWO YEAR SUBSCRIPTION $48 ____________

SAMPLE OF LATEST ISSUE $4 ____________

START MY SUBSCRIPTION WITH ISSUE ____________
(#1 - July 1978 • #7 - January 1979)

NEW SUBSCRIPTION ________ RENEWAL ________

CREDIT CARD NUMBER ____________ EXP. DATE ____________

SIGNATURE ____________

NAME ____________

ADDRESS ____________

____________

in arrays. In the machine language of a microcomputer, only bytes (and sometimes words) are mobile, and even then generally not for I/O (input/output). Only numbers and Booleans, and sometimes strings, are truly mobile in BASIC, FORTRAN, and ALGOL.

The major languages developed in the 1950s and 1960s whose structured data types are mobile are (in order of development) LISP, APL, and SNOBOL, the respective types being lists, arrays, and strings. LISP and APL also have mobile strings. In LISP, atoms serve as strings. In APL, a vector of characters is printed without spaces between its characters and so can play the role of a string. LISP and SNOBOL have arrays that are not nearly as mobile as APL's arrays, though some implementations of LISP come close, namely to within the ability to read and write them from and to devices.

Lists are preferable to arrays as a general-purpose data type since anything that an array can represent can be conveniently represented by a list, whereas the converse is far from true. You can't have arrays of differently shaped arrays in APL, for example: LISP, however, permits any data type to be a list element. In this respect, APL data types are not fully mobile with respect to array elements viewed as data sites (which they are).

From the implementation (and hence the efficiency) viewpoint, arrays offer faster random access. However, the modern APL style of programming makes relatively light use of random access. (This is a potential source of endless and quite technical debate between LISP and APL enthusiasts, and is not by any means an easy issue to dismiss.) Moreover, as compiler optimizers get progressively "smarter," it will become progressively harder to infer properties of the implementation from properties of the language definition.

For example, often the compiler has enough information to infer that a LISP list is being used array-style, and it can then choose to represent the list as an array. Conversely it may spot that an APL array would best be implemented as a LISP-style list (eg: when much concatenation of APL arrays is being performed and no random access is used).

An aspect of APL not shared with LISP is its insistence on homogeneous arrays. In APL you can have arrays of numbers, or arrays of characters, but not arrays of a mixture. An advantage of this is that you don't need to store type information for every array element, leading to efficiency gains. A disadvantage is that it restricts the programmer's options considerably. LISP programmers take full advantage of the ability to mix types in lists.

LISP and APL (and to an extent SNOBOL) have mobile expressions. In LISP you can treat the expression (PLUS X (TIMES Y 5)) as an ordinary datum. It can be bound, that is, assigned to variables, passed as an argument to a function, returned as the value of the function, printed out, and read back a year later, still meaning the same thing. And, of course, it can be evaluated by applying the LISP function EVAL to it.

The mobility of an expression is inherited from that of its representing medium, just as the mobility of an integer in the range —128 to 127 is inherited from that of the 8 bit byte that represents it.

With some restrictions, the same is true of APL. The string (ie: character vector) 'X+Y×5' can be passed around just as freely in APL, and of course it can be executed by applying the APL function Execute to it. One restriction is that Execute cannot handle more than one line at a time, effectively preventing the use of APL's version of Goto in conjunction with Execute. Another restriction is that there is no APL expression whose execution results in an APL function becoming defined; instead one uses a separate function, ⎕FX. LISP observes neither of these restrictions.

LISP goes beyond APL by also having mobile functions. From a programmer's viewpoint the main difference between an expression and a function is that functions are objects that explicitly take arguments, whereas the only way to pass information to an expression is to store it in some variables before evaluating the expression.

LISP implements mobile functions by using lambda expressions, a method of representing functions due to the logician Alonzo Church. For example, the function that computes the length of a two-dimensional vector whose coordinates are X and Y could be represented with the list:

```
(LAMBDA (X Y) (SQRT (PLUS (TIMES X X)
                          (TIMES Y Y))))
```

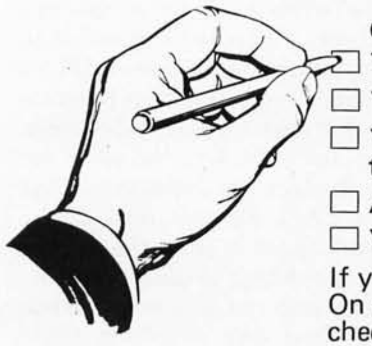
Such an object can be read, printed, assigned to variables, passed as an argument to another function, returned as the result of a function, and of course applied to a pair of arguments. To take an unusual example, run-



# 5 reasons why you should not buy the electric pencil II™

© 1978 Michael Shrayer



Check the appropriate box(es):

- ☐ **You love typing the same copy 20 thousand times a day.**
- ☐ **Your secretary can type 250 words per minute.**
- ☐ **You're dying to spend $15,000 on a word processing system, just for the tax investment credit.**
- ☐ **All your capital assets are tied up in a 10-year supply of correction fluid.**
- ☐ **You never commit a single thought to paper.**

If you have checked one or more boxes, you do not need The Electric Pencil. On the other hand, you may want to join the thousands of people who haven't checked a single box.

The Electric Pencil II is a *Character Oriented Word Processing System.* This means that text is entered as a string of continuous characters and is manipulated as such. This allows the user enormous freedom and ease in the movement and handling of text. Since line endings are never delineated, any number of characters, words, lines or paragraphs may be inserted or deleted anywhere in the text. The entirety of the text shifts and opens up or closes as needed in full view of the user. The typing of carriage returns or word hyphenations is not required since lines of text are formatted automatically.

As text is typed and the end of a line is reached, a partially completed word is shifted to the beginning of the following line. Whenever text is inserted or deleted, existing text is pushed down or pulled up in a wrap around fashion. Everything appears on the video display as it occurs, which eliminates guesswork. Text may be reviewed at will by variable speed scrolling both in the forward and reverse directions. By using the search or search and replace functions, any string of characters may be located and/or replaced with any other string of characters as desired.

Numerous combinations of line length, page length, line spacing and page spacing permit automatic formatting of any form. Character spacing, bold face, multicolumn and bidirectional printing are included in the Diablo versions. Multiple columns with right and left justified margins may be printed in a single pass.

## Wide screen video

Versions are available for Imsai VIO video users with the huge 80x24 character screen. These versions put almost twice as many characters on the screen!!!

## CP/M versions

Digital Research's CP/M, as well as its derivatives, including IMDOS and CDOS, and Helios PTDOS versions are also available. There are several NEC Spinwriter print packages. A utility program that converts The Electric Pencil to CP/M to Pencil files, called **CONVERT**, is only $35.

## Features

- CP/M, IMDOS *and* HELIOS compatible
- Supports four disk drives
- Dynamic print formatting
- DIABLO and NEC printer packages
- Multi-column formatting in one pass
- Print value chaining
- Page-at-a-time scrolling
- Bidirectional multispeed scrolling controls
- Subsystem with print value scoreboard
- Automatic word and record number tally
- Cassette backup for additional storage
- Full margin control
- End-of-page control
- Non-printing text commenting
- Line and paragraph indentation
- Centering
- Underlining
- Bold face

## Upgrading policy

Any version of The Electric Pencil may be upgraded at any time by simply returning the *original* disk or cassette and the price difference between versions, plus $15 to Michael Shrayer Software. Only the originally purchased cassette or diskette will be accepted for upgrading under this policy.

**mss**

**MICHAEL SHRAYER SOFTWARE, INC.**
1253 Vista Superba Drive
Glendale, CA. 91205
(213) 956-1593

## Have we got a version for you?

The Electric Pencil II operates with any 8080/Z80 based microcomputer that supports a CP/M disk system and uses an Imsai VIO, Processor Tech. VDM-1, Polymorphic VTI, Solid State Music VB-1B or Vector Graphic video interface. REX versions also available. Specify when using CP/M that has been modified for Micropolis or North Star disk systems as follows: for North star add suffix A to version number; for Micropolis add suffix B, e.g., SS-IIA, DV-IIB.

| Vers. | Video | Printer | Price |
|---|---|---|---|
| SS-II | SOL | TTY or similar | $225. |
| SP-II | VTI | TTY or similar | 225. |
| SV-II | VDM | TTY or similar | 225. |
| SR-II | REX | TTY or similar | 250. |
| SI-II | VIO | TTY or similar | 250. |
| DS-II | SOL | Diablo 1610/20 | 275. |
| DP-II | VTI | Diablo 1610/20 | 275. |
| DV-II | VDM | Diablo 1610/20 | 275. |
| DR-II | REX | Diablo 1610/20 | 300. |
| DI-II | VIO | Diablo 1610/20 | 300. |
| NS-II | SOL | NEC Spinwriter | 275. |
| NP-II | VTI | NEC Spinwriter | 275. |
| NV-II | VDM | NEC Spinwriter | 275. |
| NR-II | REX | NEC Spinwriter | 300. |
| NI-II | VIO | NEC Spinwriter | 300. |
| SSH | SOL | Helios/TTY | 250. |
| DSH | SOL | Helios/Diablo | 300. |

## Attention: TRS-80 Users!

The Electric Pencil has been designed to work with both Level I (16K system) and Level II models of the TRS-80, and with virtually any printer you choose. Two versions, one for use with cassette, and one for use with disk, are available on cassette. The TRS-80 disk version is easily transferred to disk and is fully interactive with the READ, WRITE, DIR, and KILL routines of TRSDOS 2.1.

| Version | Storage | Price |
|---|---|---|
| TRC | Cassette | $100. |
| TRD | Disk | $150. |

## Demand a demo from your dealer!

Circle 319 on inquiry card.

ning the program (APPLY (READ) (LIST 3 4)) would cause the function typed in response to the Read to be applied to the list of arguments (3 4). If the user typed in the above lambda expression, the result would be 5.

The closest APL can come to this is to have a name of a function, say ZOT, be a datum. To apply the function so named in APL, one would concatenate the name with the argument(s), say 3, then Execute the resulting program "ZOT 3". The catch is that names on their own mean nothing: the technique will not work if the name is not defined, or if somebody changes its definition. Thus if you print the name of an APL function on a device from which you want to read it back in later, the original definition may in the meantime change or disappear from the workspace. This difficulty does not arise with lambda expressions, which contain their own definition. Thus functions have at best limited mobility in APL.

The notion of mobility, perhaps surprisingly, is not a concept that many people are familiar with. In *hindsight* it is clear that mobility was a concern, whether or not a subconscious one, for the designers of LISP, APL and SNOBOL. The late Christopher Strachey, a British computer scientist, made the distinction between "first and second class citizens" when discussing data, the former being what I have called mobile data. The first published reference to the concept appears to have been made in 1968 by another British computer scientist, Robin Popplestone, in a description of the virtues of his language POP-2. Popplestone did not use the word "mobile" either, talking instead in terms of a "charter of rights" for data.

### Modularity of Function

Subroutine libraries have something that programming languages often lack, and that is modularity of function. One does not view a subroutine library as a monolith but rather as a loosely coupled set of subroutines. The term *subset*, often applied in a vague way to programming languages, has an obvious and precise meaning for subroutine libraries.

LISP and APL, in contrast, are each just like a subroutine library, being little more than a set of functions. The user may add to this set by getting more functions from whatever subroutine library is maintained by the local environment. And the user's program itself consists of a set of functions. Any of these functions can be invoked from the user's terminal or from the user's or any other program. All three kinds are invoked with identical syntax (within each language), in LISP:

(Function Arg1 Arg2 ...Argn)

in APL:

op x    for unary functions
x op y   for binary functions, assuming right associativity

The conventions for representing lists, LISP's primary structured data type, are the same for representing programs. Since those conventions are simple, there is little to learn. In this respect LISP differs from APL, which has a convention for representing the structure of its programs (namely the invocation of the right-associativity rule, that x op y op z is read as x op (y op z) that has no analog in the representation of APL data.

I should add that my own preference in programming in LISP is to use an ALGOL-like language, CGOL, which is then automatically translated to LISP. Despite the regular and easily learned syntax of LISP, I do not like having to write $x+y$ as (PLUS X Y). I do too much mathematics to feel comfortable switching representations in order to program. Fortunately it is not necessary to compromise functional modularity in order to use other syntactic conventions. If I were an APL programmer I would want to do the same thing: have a syntactic preprocessor that permitted me to use the syntax I felt most comfortable with.

### Declarative Programming

Here is an innocent looking pair of equations:

$$(a+1) \times b = a \times b + b$$
$$0 \times b = 0$$

What sets these equations apart from the millions of other equations I could have written is that these permit me to convert any method for adding into a method for multiplying nonnegative integers. Suppose, for example, I want to multiply 3 by 7. Since $3 = 2+1$, I can use the equation to express $3\times7$ as $2\times7+7$, reducing the original problem to a smaller one which can be solved by the same method. Eventually I have $(((0\times7+7)+7)+7$, which the second equation turns into $((0+7)+7)+7$. Using the method for adding, three times, I end up with the desired answer.

Turning these equations into a LISP program to give a recursive definition of (TIMES A B) is an essentially mechanical procedure yielding:

```
(COND ((ZEROP A) 0)
      (T (PLUS (TIMES (SUB1 A) B) B)))
```

or in the "syntactically sugared" version of LISP referred to earlier:

```
if a=0 then 0 else (a-1)*b+b
```

The significance of this example lies in two observations: first, the facts were so obvious it was hard to make a mistake; and secondly, the procedure for converting those facts into something we could run as a program was so stereotyped and straightforward (match the problem against the lefthand side of an equation, replace it by the corresponding righthand side) that, again, it was hard to make a mistake.

Programming in LISP comes close enough to this *declarative* style to make programming a remarkably error-free process. To those who can read LISP, a well-written LISP program will look like a collection of facts. The subtlety of the program then amounts to the subtlety of the facts. If the facts are obvious, as with the above, there is little to explain. If the facts are not obvious, then you have a program that needs to be proved correct.

*Table 1: LISP finds applications in many areas dealing with language processing.*

| Area | Language |
|---|---|
| Compiling | Parsed programs |
| Algebraic simplification | Algebraic formulas |
| Natural language | Parsed sentences |
| Automatic theorem proving | Logical formulas |
| Program verification | Parsed programs and logical formulas |
| Automatic programming | Specifications and resulting programs |
| Knowledge'based systems | Facts and rules |

Though the example above dealt with numbers, the mobility of LISP's structured data types makes it possible to apply the same method to writing programs that operate on lists, functions, programs, and so on.

My own research includes developing and testing new algorithms for a variety of problems. For the sake of ease of implementation and short debugging time, my style is, as far as possible, to set down the facts relevant to the computation and express them as LISP functions. Thanks to the quality of the LISP compiler used at MIT, I can produce reasonably efficient programs, in many cases as efficient as if I had adopted a more traditional style of programming with while loops and assignments. (One thing I miss, however, is the ability to just write down the pure equations and have a preprocessor automatically combine them into a single LISP program.)

My prime testing program referred to in Martin Gardner's "Mathematical Games" column in the August 1978 *Scientific American* is written entirely in this style. Some of the facts it uses are obvious ones concerning such topics as exponentiation modulo n. Some of the facts however are considerably deeper and were first proved by the well-known computer scientist Michael Rabin.

Rewriting this particular program in some other programming style would achieve little, if anything, in the way of efficiency. It would, however, make it harder to see the connection between the collection of facts supporting the method and the program itself. Rewriting the program in another programming language while preserving the declarative style would be possible provided recursion was permitted and numbers were mobile. A problem here is that numbers of the size my program works with, up to 1000 decimal digits, are not merely immobile in most languages, they do not even exist. The implementation at MIT is one of the implementations which takes much effort to protect the programmer from frequent painful encounters with boundaries by not limiting the size of integers.

This principle of executing facts as programs has encouraged people to generalize the idea to other facts besides equations, and a series of programming languages have evolved based on this generalization, two of the more prominent ones being Planner and Prolog.

## Metalanguage

*Meta* is Greek for *about*. LISP lists can be used, inter alia, to represent expressions in various languages. Thus LISP makes an ideal metalanguage, a language for talking about language. As such, LISP finds applications in a large variety of areas dealing with the processing in language, as shown in table 1.

In all of these areas, the expressions of the language in question are treated as structures rather than as strings. Structures represent the level of language processing where the real action takes place. Parsing (eg: converting strings to structures) may present more or less of a challenge depending on the area, but the general feeling in most such areas is that it is what takes place after parsing that is more interesting.

What makes LISP particularly well-suited to these applications is that they frequently call for operations on expressions that are best viewed recursively as facts and procedures stated in terms of the immediate constituents of the expressions. This is an instance of the declarative style described earlier, for the case when the data are expressions.

To take an example from algebraic simplification, the derivative of an expression can be defined in terms of the derivatives of its immediate constituents. Thus (DERIV '(PLUS X Y)) would be:

```
(LIST 'PLUS (DERIV X)
            (DERIV Y))
```

where X and Y themselves may be quite complicated algebraic expressions. Similarly (DERIV '(TIMES X Y)) would be:

```
(LIST 'PLUS (LIST 'TIMES (DERIV X) Y)
            (LIST 'TIMES X (DERIV Y)))
```

and so on for other operators. From such facts it is straightforward to construct a recursive LISP program for differentiating algebraic expressions.

A helpful way to think about the principle illustrated by the above is to view the equations from which the LISP programs are derived as dealing with only a small region of an expression at a time. While algebra tends to supply particularly nice examples of this principle, the principle in one form or another pervades essentially all areas where linguistic structures are encountered.

## Conclusion

This discussion of LISP has confined itself to those aspects of LISP directly visible to the user. It has not considered LISP's substantial contributions to language implementation technology, such as garbage collection, the interpreter/compiler dichotomy, and dynamic module linking in place of the usually more static linking loader. It did consider LISP's relation to other languages, finding APL to be as good as LISP in some respects, but lacking in some particularly vital areas.

While it is difficult to consider LISP unique in any single one of its aspects, when looked at as a whole LISP stands out as a quite remarkable and original language that does credit to its inventor, John McCarthy. ■

# CBASIC Business Software for Microcomputers

Osborne & Associates is publishing CBASIC editions of its business software books, **Payroll With Cost Accounting, Accounts Payable and Accounts Receivable** and **General Ledger.** The CBASIC programs in these new editions will run on many floppy disk-based microcomputer systems.

## PAYROLL WITH COST ACCOUNTING

## ACCOUNTS PAYABLE AND ACCOUNTS RECEIVABLE

## GENERAL LEDGER


Each book includes complete source listings with remarks, program and system documentation, and an operator's manual. All this for just $15.00! The softbound Wang BASIC versions of all three books are available now. **CBASIC Payroll With Cost Accounting** is also available now. The remaining books in the CBASIC series will be published in the fall of 1979. The new CBASIC-2 editions are in a convenient, loose-leaf format.

Not a programmer? Don't want to key in thousands of lines of source code from the book? You can buy the programs ready-to-run on floppy disk from one of our 200 dealers. Contact us for the name of your nearest dealer.

CBASIC or Wang BASIC won't work? Here is a partial list of independent consultants who have converted Osborne programs to run on many popular systems using our books as their manuals. Contact us for a complete, up-to-date list of these conversions.

## CONSULTANTS • COMPUTER STORES • SOFTWARE HOUSES

You can be a dealer for the magnetic surface of Osborne's CBASIC-2 software. We will sell the programs on 8" floppy disks to dealers, consultants and computer stores only. The disk for each book sells for $250.00. Once you buy the floppy disk you may copy it as many times as you like, resell it, change the programs, or use it for your own purposes. But the best part is that we have already written your operator's manuals — our books! We grant you the unrestricted right to use, transfer, or otherwise deal in any way with the machine-readable form of our BASIC programs. We copyright only the printed word in our books.

**ALPHA MICRO:**
Peter Burke
THE BASIC BUSINESS SOFTWARE CO., INC.
P.O. Box 2032
Salt Lake City, Utah 74110 (801) 363-1199

**APPLE II:**
Jimmy Walter
A.C.E. COMPUTER SYSTEMS
2449 North Westshore
Tampa, Florida 33607 (813) 872-2798

**COMMODORE PET:**
Chuck Stuart
CMS SOFTWARE
5115 Menefee Drive
Dallas, Texas 75227 (214) 381-0690

**CROMEMCO:**
Richard French
THE NETWORK COMPUTER SYSTEMS
495 Third Avenue #8
San Francisco, California 94118 (415) 668-7777

**DATAPOINT 1500:**
Paul Marchetti
OCCIDENTAL COMPUTER SYSTEMS, INC.
6666 Valjean Avenue
Van Nuys, California 91406 (213) 782-3005

**MICROPOLIS, NORTHSTAR (REQUIRES CP/M):**
James Rountree
R&A COMPUTER SERVICES
7700 Edgewater Drive, Suite 723
Oakland, California 94621 (415) 562-3133

**MICROSOFT disk BASIC:**
Dan Kindred
GNAT COMPUTERS
7895 Convoy Court
San Diego, California 92111 (714) 560-0433

**NORTHSTAR BASIC:**
Isaac Bornstein
COMPUTER SERVICES OF ENCINITAS
341 Willowspring Drive
Encinitas, California 92024 (714) 436-2486

**POLYMORPHIC 8813:**
Bruno Caprez
KEYSTONE SYSTEMS
P.O. Box 767
Spokane, Washington 99210 (509) 747-5623

**TRS-80:**
Don French
APPLIED DATA CORPORATION
P.O. Box 16020
Fort Worth, Texas 76133 (817) 738-0251

Irwin Taranto
TARANTO & ASSOCIATES
P.O. Box 6073
San Rafael, California 94903 (415) 472-1415

**WANG BASIC ON FLOPPY DISK:**
Richard Armour
ATLANTIC COMPUTING & CONSULTING
P.O. Box 7294
Hampton, Virginia 23666 (804) 340-9350

## Order Form

**Name:** ____________________
**Address:** ____________________
**City:** ____________________
**State:** __________ **Zip:** __________
**Phone:** ____________________

**OSBORNE & ASSOCIATES**
630 Bancroft Way, Dept. I23
Berkeley, CA 94710 USA
(415) 548-2805 TWX 910-366-7277

| Title | Price | Quantity | Amount |
|---|---|---|---|
| 22-5 Payroll With Cost Accounting - CBASIC | $15.00 | | |
| 23-3 Accounts Payable/Receivable - CBASIC | 15.00 | | |
| 24-1 General Ledger - CBASIC | 15.00 | | |
| 09-8 Payroll With Cost Accounting - Wang BASIC | 15.00 | | |
| 13-6 Accounts Payable/Receivable - Wang BASIC | 15.00 | | |
| 20-9 General Ledger - Wang BASIC | 15.00 | | |
| | California residents tax | | |
| | Shipping | | |
| | Total Amount Enclosed | | |

**Shipping** (Shipping for large orders to be arranged)
□ All foreign orders $4.00 per book for airmail
□ $0.45 per book 4th class in the U.S.(allow 3-4 weeks)
□ $0.75 per book UPS in the U.S. (allow 10 days)
□ $1.50 per book special rush shipment by air in the U.S.

Calif. residents add 6% sales tax.
S.F. BART residents add 6-1/2 % sales tax.

□ I am an end user interested in running CBASIC programs on disk. Please notify me of the closest dealer for:

□ I am a dealer interested in purchasing CBASIC programs on disk for resale. Please send description and order information for:

□ CBASIC Payroll With Cost Accounting
□ CBASIC Accounts Payable/Receivable
□ CBASIC General Ledger

Please send the following information:
□ Becoming an Osborne dealer
□ School discounts
□ List of foreign distributors
□ Other Osborne publications
□ Please send me the current conversion list

Payment in advance must be enclosed for purchases of up to $70.00. Invoicing U.S. purchases over $70.00 available upon approval of your account. All foreign orders must be prepaid in U.S. dollars drawn on a U. S. bank.

S1034

Circle 292 on inquiry card.

# A Preview of the Motorola 68000

A I Halsema
4921 Patrae St
Los Angeles CA 90066

It is difficult to classify the new Motorola 68000 processor. It seems incongruous to call a machine with 32 bit wide data paths a microcomputer. The 68000 should be available in late 1979. As this is being written, the architecture of the machine has been frozen, and the microcode is nearing completion. A user programmable, on chip, control memory for dynamically changing the machine's instruction set is not planned, but you may be able to specify your own microcode, which is burned into an on chip read only control memory at the factory.

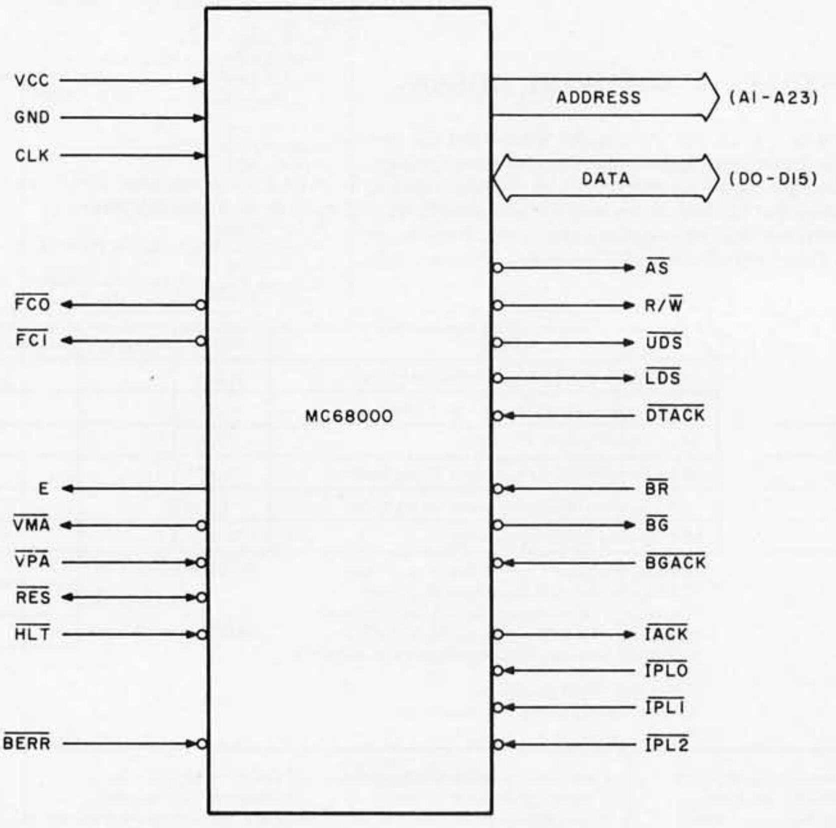



*Figure 1: Pin assignments for the Motorola 68000 package. A description of the functions of the different pins is found in table 1. Figure reproduced courtesy of Motorola Semiconductor Products Inc.*

Using HMOS (high density metal oxide silicon), the 68000 will come in a 64 pin package (see figure 1). Capable of directly addressing up to 16 M bytes (actually $2^{24}$ = 16,777,216 bytes) of memory, the 68000 is about 15 times more complex than the 6800 (coincidentally it has about 68,000 transistors on the chip, and has about 10 times greater throughput). External data paths are 16 bits wide and access memory that is organized as bytes. Through the use of a signal called VPA (valid peripheral address), the 68000 will be able to use the slower 6800 peripheral devices.

Internally the 68000 is an orthogonal and consistent machine, with 16 identical 32 bit accumulators, 61 basic mnemonics (shown in table 2), which can be used with any of the 14 addressing modes and any of the six data types. See figure 2 for an illustration of the programming model. The five basic addressing modes are *register direct, register indirect, absolute, immediate,* and *program counter relative.* The ability to do postincrementing, predecrementing, offsetting, and indexing is included. Data types recognized by the machine are bits, bytes, BCD (binary coded decimal) digits, ASCII characters, 16

**About the Author**

*Aillil Ian Halsema has worked as a programmer since 1971. He is now a senior member of the programming staff at Xerox Corp. His personal computer system includes a Southwest Technical Products Co 6800 and Okidata CP-110 printer.*

bit words, and 32 bit long words. By combining the instructions, data types, and addressing modes, more than 1000 instructions are available.

Some of the more interesting instructions are PACK (pack ASCII to BCD digit form), UNPK (unpack from BCD digits to ASCII), CHK (check register against bounds), TRAP (provides access to 16 software trap vectors), LINK, and UNLK (linked list operations). With eight levels of priority interrupts, this machine can access 256 interrupt vectors. Hardware traps to catch software errors include word access with odd address, illegal instruction, unimplemented instruction, illegal addressing mode, illegal memory access, overflow on divide, and overflow condition code. Through the use of the unimplemented instruction trap, the user can implement his own operation codes (in a fashion similar to SVC on the IBM 360/370 systems).

Designed with timesharing in mind, the 68000 has supervisory and user states, with the ability to run eight tasks in the user state simultaneously. Supervisory state makes certain instructions legal for operating a separate memory management controller. This controller will provide dynamic management of memory segments that contain read only data, read/write data, program code, or protected data or code. As an aid in debugging, the machine includes a bit in the status register that, when set, puts the machine into single step operation.

The 68000 instruction set was designed by programmers for programmers, and is designed for ease of use in compiler generation and timesharing system implementation. The orthogonality referred to above reduces the number of details the programmer must keep in mind when programming — a register is a register like any other on the machine, with no special conditions restricting register use.

Pin Identification and Definitions

| | | |
|---|---|---|
| A1-A23 | Address Leads | 23 bit address bus; capable of addressing 16,777,216 bytes in conjunction with UDS and LDS. |
| D0-D15 | Data Leads | 16 bit data bus; transfers 8 or 16 bits of information. |
| AS | Address Strobe | Indicates valid address and provides a bus lock for indivisible operations. |
| R/W | Read/Write | Defines bus operation as Read or Write and controls external bus buffers. |
| UDS, LDS | Data Strobes | Identifies the byte(s) to be operated on according to R/W and AS. |
| DTACK | Data Transfer Acknowledge | Allows the bus cycle to synchronize with slow devices or memories. |
| BR | Bus Request | Input to the processor from a device requesting the bus. |
| BG | Bus Grant | Output from the processor granting bus arbitration. |
| BGACK | Bus Grant Acknowledge | Confirmation signal from BG indicating a valid selection from the arbitration process. |
| IACK | Interrupt Acknowledge | Identifies that the bus is performing an interrupt service cycle. |
| IPL0, IPL1, IPL2 | Interrupt Priority Level | Provides the priority level of the interrupting function to the processor. |
| FC0, FC1 | Function Code | Provides external devices with information about the current bus cycle. |
| CLK | Clock | Master TTL (transistor-transistor logic) input clock to the processor. |
| RES | Reset | Provides reset (initialization) signal to the processor and peripheral devices. |
| HLT | Halt | Stops the processor and allows single stepping. |
| BERR | Bus Error | Provides termination of a bus cycle if no response or an invalid response is received. |
| E | Enable | Enable clock for M6800 systems. Identifies addressed area as a 6800 compatible area. |
| VPA | Valid Peripheral Address | |
| VMA | Valid Memory Address | Indicates to 6800 family devices that a valid address is on the bus. |
| $V_{cc}$ | +5 V | — |
| GND | Ground (two pins) | — |

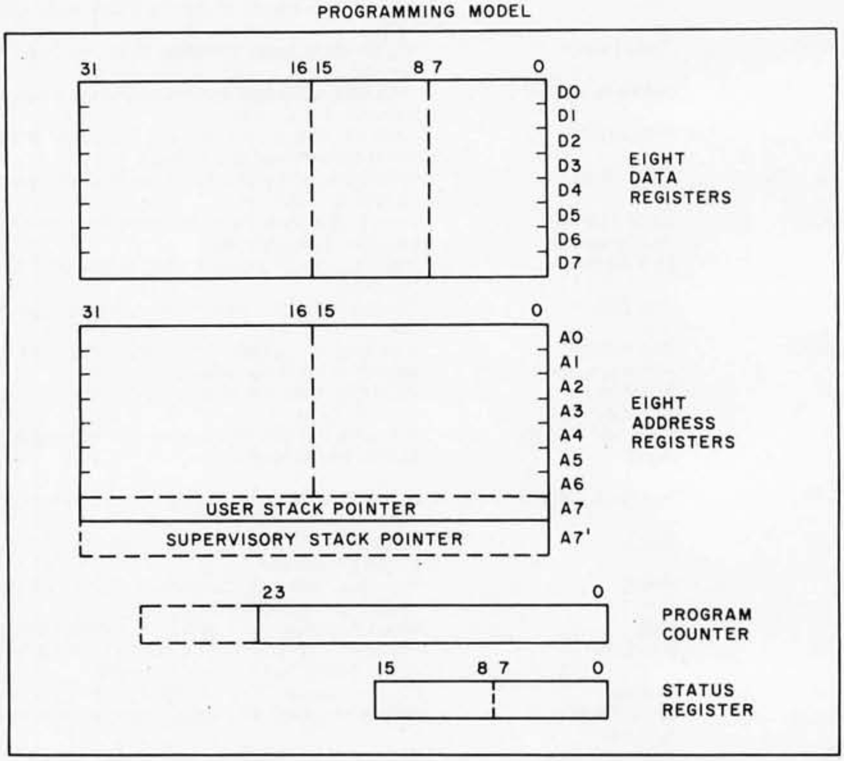
*Table 1: Description of pin functions on 68000 processor.*

## Applications

Computers are useful for processing vast amounts of data, and for performing long repetitive sequences of operations. Since the personal computer enthusiast has neither the facilities nor the time to collect large amounts of data for processing, the computer is more likely to be used in the second mode (number crunching). Today's microprocessors fail miserably as number crunchers due to low speeds and limited amounts of memory space. The 68000 will correct these deficiencies. (Coupled with the new low cost, high density memory devices with 64 K bit capacity and with even greater density coming, the personal computer will attain or exceed the power of an IBM 360 Model 30 within the next decade.) Number crunching applications requiring little external storage (ie: disk or drum) include artificial intelligence, encryption/decryption, simulation, games, and Dynabook type applications. [See the article by Alan Kay on page 230 of the September 1977 *Scientific American* for a general description of small talk, a software system intended for small portable Dynabook computers....**CH**]

Artificial intelligence attempts to provide the computer with the ability to learn from past experience (ie: heuristic procedures), and to simulate operations of the human brain in recognizing patterns. Brain simulations are generally performed using arrays in memory as brain cells, with software logic taking the part of the complex interconnections between cells. Array arithmetic requires a fair amount of processing power. Such power is not available on 8 bit machines.

A common array operation in artificial intelligence is finding the inner or dot product of two arrays. If array X represents a set of cell states, and array D represents data upon which the "brain" is to work, then the inner product of the two arrays is represented by: $z = X_1D_1 + X_2D_2 + \ldots + X_nD_n$, where z might be the result of a vote taken by n cells of the "brain" in a committee network. This calculation can be very slow on an 8 bit machine without hardware multiply, and exceedingly slow if the arrays are large or each element is several bytes long. Multidimensional arrays take up large amounts of



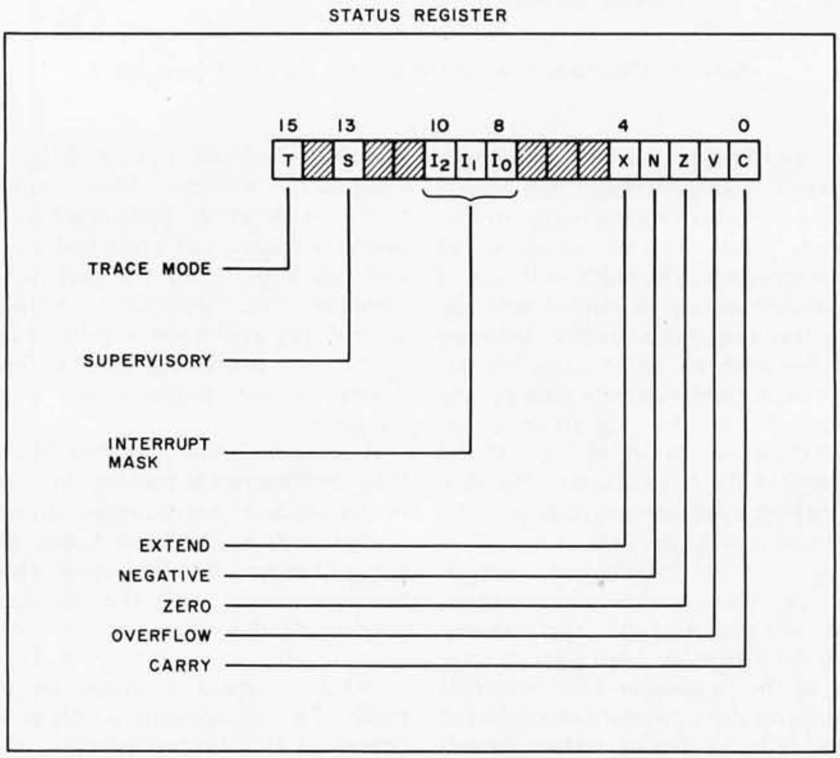




*Figure 2: Programming model and register organization for the Motorola 68000 processor. Note that the data registers and address registers are functionally identical except for register A7. A7', the supervisory mode stack pointer, is not available to the programmer. Figure reproduced courtesy of Motorola Semiconductor Products Inc.*

memory space which today's microprocessors cannot support.

Modern methods of encrypting and decrypting messages can require large amounts of processing power. As an example, the method for obtaining digital signatures and public key cryptosystems developed by R L Rivest (a "trapdoor" system) requires raising the message text to a power and dividing by two large secret prime numbers. Large means anything from 50 to 500 digits, with the larger numbers giving greater security. The mathematical operations of raising to a power, performing lengthy divisions, and finding the large prime numbers (which need be done only once) cannot feasibly be performed on an 8-bit machine, but come within the realm of the possible when using the 68000.

A simple example that the reader can program involves finding the Godel number (named after the mathematician who discovered them) which encrypts a word or message. Each character in the message is represented by the natural order of primes (2, 3, 5, 7, 11, 13, ...etc). The identity of the letter occupying a position in the message is given by an exponent: the exponent 1 meaning that the letter is an A, 2 meaning a B, etc. The message as a whole is then rendered as the product of all the bases and exponents. For example, the word "CAB" can be represented as $2^3 \times 3^1 \times 5^2$ or 600 ($8 \times 3 \times 25 = 600$). Decode the message by dividing the product by each prime number until a remainder appears. The number of divisions is the exponent representing a particular character. Regardless of how the problem is ordered, much computation is required to find the prime numbers, exponentiate, and multiply. This gives you an idea of the sort of processing power required for a full public key cryptosystem.

Games and simulations can become more complex. A space war game was programmed nearly a decade ago at Massachusetts Institute of Technology that included realistic simulations of orbital mechanics in the vicinity of a planet or star. A space war game with simulations of relativistic effects at near light speeds could be challenging both for the

player and the programmer. Simulations of nonlinear and dynamic processes require the large amounts of computing power made available by the 68000. High resolution graphics require the large address space provided by the 68000, and with sufficient processing speed, true real time animation can be created.

Dynabook is a project headed by Alan Kay at Xerox Corporation's Palo Alto Research Center. One of the objects is to provide the power of a medium size computer in a package the size of one encyclopedia volume. The 68000 computer, bubble memories, and low cost semiconductor memories bring this target within reach. With 8 simultaneous tasks, the owner of such a system could use one task as a clock, one for a calculator, one for personal data base processing, another for memos, reminders, and schedules, and yet another for text processing, and still have 3 other tasks available for long-term number crunching, games, or whatever the imagination can visualize. With as much as 16 M bytes of memory, each task could be allotted 2 M bytes. This amount of storage is difficult to comprehend, but for comparison, the text of this article requires about 10,000 bytes of storage. This Dynabook system would be battery powered and portable, with a solid-state display and thin, typewriter keyboard.

There can be no doubt that the inexpensive super computer is coming. IBM estimates that an entire central processing unit with 1 M bytes of memory will fit in a cube 1 inch (2.54 cm) on a side by the end of the 1980s. [This particular device will require cooling to superconductor temperatures.] An example of what is possible with today's technology can be seen in Texas Instruments' "Speak and Spell" toy, which for under $50 provides a keyboard, alphanumeric display, and microprocessor controlled speech feedback with a vocabulary of about 250 words and numerous messages and phrases. The functions that can be performed by the Motorola 68000 and the new generation of microprocessors it represents are limited only by the imagination. ■

| Mnemonic | Description |
|---|---|
| ABCD | Add Decimal with Extend |
| ADD | Add |
| ADDX | Add with Extend |
| AND | Logical And |
| ASL | Arithmetic Shift Left |
| ASR | Arithmetic Shift Right |
| BCC | Branch Conditionally |
| BCHG | Bit Test and Change |
| BCLR | Bit Test and Clear |
| BRA | Branch Always |
| BSET | Bit Test and Set |
| BSR | Branch to Subroutine |
| BTST | Bit Test |
| CHK | Check Register Against Bounds |
| CLR | Clear Operand |
| CMP | Arithmetic Compare |
| DCNT | Decrement and Branch Nonzero |
| DIVS | Signed Divide |
| DIVU | Unsigned Divide |
| EOR | Exclusive Or |
| EXG | Exchange Registers |
| EXT | Sign Extend |
| JMP | Jump |
| JSR | Jump to Subroutine |
| LDM | Load Multiple Registers |
| LDQ | Load Register Quick |
| LEA | Load Effective Address |
| LINK | Link Stack |
| LSL | Logical Shift Left |
| LSR | Logical Shift Right |
| MOVE | Move |
| MULS | Signed Multiply |
| MULU | Unsigned Multiply |
| NBCD | Negate Decimal with Extend |
| NEG | Two's Complement |
| NEGX | Two's Complement with Extend |
| NOP | No Operation |
| NOT | One's Complement |
| OR | Logical Or |
| PACK | Pack ASCII to BCD (binary coded decimal) |
| PEA | Push Effective Address |
| RESET | Reset External Devices |
| ROTL | Rotate Left without Extend |
| ROTR | Rotate Right without Extend |
| ROTXL | Rotate Left with Extend |
| ROTXR | Rotate Right with Extend |
| RTR | Return and Restore |
| RTS | Return from Subroutine |
| SBCD | Subtract Decimal with Extend |
| SCC | Set Conditional |
| STM | Store Multiple Registers |
| STOP | Stop |
| SUB | Subtract |
| SUBX | Subtract with Extend |
| SWAP | Swap Data Register Halves |
| TAS | Test and Set Operand |
| TRAP | Trap |
| TRAPV | Trap on Overflow |
| TST | Test |
| UNLK | Unlink Stack |
| UNPK | Unpack BCD to ASCII |

*Table 2: Instruction set of 68000 processor. Operation of instructions is as consistent as possible.*

## BIBLIOGRAPHY


1. Feucht, Dennis, "Pattern Recognition: Basic Concepts and Implementation," *Computer Design*, December 1977, pages 57 thru 68.

2. Kay, Alan C, "Microelectronics and the Personal Computer," *Scientific American*, volume 237, number 3, September 1977, page 230.

3. Rivest, R L; Shamir, A; Adleman, L, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, volume 21, number 2, February 1978, pages 120 thru 126.

# Here's the LATEST news...

We're expanding again! **CompuKit** is now **CompuPro** to reflect the fact that more and more of our production is being devoted to non-kit products . . . not only is there increased demand for our standard **assembled & tested** models, but the high-reliability **Certified System Component** boards (with 200 hour burn-in and immediate replacement in event of failure within 1 year of invoice date) are really taking off. Kit builders need not fear, however; when economy is paramount, most **CompuPro** products are also available in **"unkit"** form (sockets, bypass caps pre-soldered in place). All of this is part of our continuing commitment to provide the best possible products at the lowest possible prices . . . the same commitment responsible for making us an industry leader in the first place.

# THE ECONORAM* MEMORY LINE:
## LOW POWER AND LOW PRICE MEET HIGH TECHNOLOGY.

| Name | Storage | Buss | Speed | Technology | Configuration | Unkit | Assm | CSC |
|---|---|---|---|---|---|---|---|---|
| **Econoram IIA** | 8K X 8 | S-100 | 4 MHz | static | 2-4K blocks | **$149** | **$179** | **$239** |
| **Econoram IV** | 16K X 8 | S-100 | 4 MHz | static | 1-16K | **$295** | **$329** | **$429** |
| **Econoram VI** | 12K X 8 | H8 | 2 MHz | static | 1-8K, 1-4K | **$200** | **$270** | N/A |
| **Econoram VII** | 24K X 8 | S-100 | 4 MHz | static | 2-4K, 2-8K | **$445** | **$485** | **$605** |
| **Econoram IX** | 32K X 8 | Dig Grp | 4 MHz | static | 2-4K, 1-8K, 1-16K | **$559** | **$639** | N/A |
| **Econoram X** | 32K X 8 | S-100 | 4 MHz | static | 2-8K, 1-16K | **$599** | **$649** | **$789** |
| **Econoram XI** | 32K X 8 | SBC | 4 MHz | static | 2-8K, 1-16K | N/A | N/A | **$1050** |

## BANK SELECT MEMORIES (for Alpha Micro Systems, Marinchip, etc.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Econoram XII-16** | 16K X 8 | S-100 | 4 MHz | static | 2 indep. banks** | **$369** | **$419** | **$ 519** |
| **Econoram XII-24** | 24K X 8 | S-100 | 4 MHz | static | 2 indep. banks** | **$479** | **$539** | **$ 649** |
| **Econoram XIII** | 32K X 8 | S-100 | 4 MHz | static | 2 indep. banks** | **$629** | **$699** | **$ 849** |

*__Econoram__ is a trademark of __Bill Godbout Electronics__

**Econoram XII-16 and -24 have 2 independent banks addressable on 8K boundaries; Econoram XIII has 2 independent banks addressable on 16K boundaries.

## SPECIAL SUMMER SALE: 16K MEMORY EXPANSION CHIP SET ~~$109~~ $87.20

**20% off** one of our all-time best sellers from July 15 to August 31 **only.** For **Radio Shack-80, Exidy Sorcerer, Apple** computers. 250 ns access time, **low** power parts, DIP shunts included, 1 year limited warranty, and easy-to-follow instructions that make memory expansion a snap.

## S-100 MOTHERBOARD

**18 slot unkit: $124.** Each motherboard includes all edge connectors wave-soldered in place for easy assembly, integral active termination circuitry, extra wide power and ground traces, and much more.

## ACTIVE TERMINATOR KIT $34.50

As written up by Craig Anderton in the April '79 issue of **Kilobaud Microcomputing.** Our much imitated design plugs into any S-100 motherboard to reduce ringing, crosstalk, noise, and other buss-related problems.

## H8 EXTENDER BOARD KIT $39

New from **Mullen Computer Products.** Really takes the hassle out of troubleshooting or testing the popular Heath H8 microcomputer; includes jumper links in the power supply lines for insertion of fuses, Ammeters, current limiters, and the like.

## S-100 EXTENDER BOARD KIT $39

From **Mullen Computer Products.** Includes jumper links in supply lines, on board logic probe, and general purpose "kluge board" area for installing custom testing jigs or other circuits.

## DUAL CHANNEL/DUAL FUNCTION S-100 I/O BOARD $189 unkit, $249 assm.

This board does things the others only dream about. Features **two** independently addressable serial ports with full RS232C, current loop (20 mA) and TTL signals on both ports. Includes on board xtal timebase and Baud rate generator for Baud rates up to 19.2 KBaud, EIA 232C receivers and drivers (1488, 1489), hardware LSI UARTs that don't tie up the computer's CPU, operation with 2 or 4 MHz systems, software programmable UART parameters/interupt enables/handshaking lines (handshaking lines are full RS232 — not just a 3 wire system), optically isolated current loop, provision for custom frequency compensation on both receive and transmit sides to accommodate varying speed/noise situations or unusual cable lengths . . . and even all this isn't the full story of what this board can do for you. See it in person at a computer store near you, or order direct from us.

## 2708 EROM BOARD "UNKIT" $85

4 independently addressable 4K blocks, with selective disable for each block. Built to **CompuPro/Econoram** standards (dipswitch addressing, top quality board, sockets wave-soldered in place), and includes dipswitch selectable jump start built right into the board. Includes all support chips and manual, but does **not** include EROMs. **Special: if you order before August 1st, our introductory price of $69.95 is still in effect.** Orders postmarked August 1st or later are no longer eligible for the introductory price . . . no exceptions.

## POPULAR COMPUTER ICs

Low power **21L02** static 1K RAMs on special: **10/$9.90** (under 1¢ per **byte!**) **1791 MOS LSI** dual density disc controller from Western Digital: **$59** with pinout and data. **1771** single density controller: **$22.50.** All parts are offered on a while-they-last basis.

We realize that it is becoming less and less common for manufacturers to offer their products in kit form, so we'd like to emphasize that we are **not** planning to phase out our kits. The "hackers" who derive enjoyment from building and testing equipment themselves were our very first customers, and we recognize that these experimentally-minded individuals have been responsible for many significant developments in the microcomputer field. As long as people want to put things together with their very own hands, we'll continue to offer kits . . . and continue to repair them promptly under the terms of our 1 year limited warranty (almost always without charge), in the rare event that a problem occurs.

**TERMS:** Cal. res. add tax. Allow 5% shipping, excess refunded. Orders under $15 add $1 handling. VISA® /Mastercharge® call our 24 hour order desk at **(415) 562-0636.** COD OK with street address for UPS. Prices good through cover month of magazine except as noted.

**CompuPro**™ from **GODBOUT** ELECTRONICS

Box 2355, Oakland Airport, CA 94614

**FREE FLYER: We'll be glad to tell you more than the space of this ad permits. Just send your name and address, we'll take care of the rest. If you're in a hurry, enclose 41¢ in stamps for 1st class delivery.**

Circle 150 on inquiry card.

# LISP Based Symbolic Math Systems

David R Stoutemyer
The Soft Warehouse
POB 11174
Honolulu HI 96828

On an interactive terminal, a user begins by typing the assignment:

Q ← 6*X↑3/(9*X)

where ← denotes assignment, * denotes multiplication, and ↑ denotes raising to a power. Such a command would be erroneous in most languages because the variable X has not previously received a value. However, symbolic math systems accept and even simplify expressions containing such *unbound variables*. Thus, the response of such a system to the above command is the automatic output:

2*X↑2/3

which is also saved as the *value* of Q. Some of the systems have more elaborate output routines which would display the above output in a two-dimensional format such as the following:

$$\frac{2X^2}{3}$$

It is the ability to accept and transform input-data consisting of expressions which contain unbound variables that most characterizes computer symbolic math. As is also illustrated by this example, virtually all such systems are capable of exact rational arithmetic. In fact, the rational arithmetic is usually *indefinite precision*, wherein each number occupies as much memory as is necessary for exact representation up to some very large maximum, imposed perhaps only by the total amount of remaining space allocated for numbers. Even the small 8080 based muMATH-79 system can compute $99^{99}$ exactly, in less than three seconds, and the SCRATCHPAD system was once involved in a proof that the incredibly large number $2^{19,937}-1$ is prime.

Virtually all symbolic math systems also support symbolic differentiation. For example, if the user enters an expression after the above assignment to Q such as the trigonometic example:

DIF(A*SIN(Q),X),

the automatic interactive response is:

4*A*X*COS(2*X↑2/3)/3

Later sections will discuss even more sophisticated built-in mathematical capabilities of these systems.

Symbolic math systems are often called *computer-algebra systems* despite their ability to do trigonometric simplification, calculus, and other operations aside from algebra.

Most general-purpose computer-algebra systems are implemented in LISP or in a disguised variant thereof, because LISP is especially suitable for the purpose. This is not to say that the user of a LISP based system must know LISP or use a LISP like syntax for his expressions. Because the syntax of traditional applied math is so different from that of LISP, each of these systems provides a *parser* which translates the traditional external representations of input expressions into corresponding internal representations which are more suitable for performing the various mathematical transformations. Similarly, each of these systems provides an output *deparser* which

**About the Author**

*David R Stoutemyer is a Professor of Electrical Engineering at the University of Hawaii. He has received his doctorate in Computer Science from Stanford University, with specialization in numerical analysis. His current research interests include both numerical and nonnumerical scientific computation. Current educational interests include innovative computer aided math education at the elementary through college level.*

translates the internal representation into a traditional mathematical representation for display.

In addition to using the built-in math facilities in the *symbolic calculator* fashion illustrated above, many users want to eventually extend the built-in capabilities by means such as entering appropriate function definitions. Since most users of these systems are accustomed to a traditional Von Neumann style of programming language, rather than LISP, the parser is also generally used to provide users with a surface programming language which resembles ALGOL or another widely acceptable syntax. In fact, many of these systems provide an extendable parser-deparser, so that the user can introduce mathematical operators and programming syntax to suit personal needs and tastes. Such functional or parser extensions can be freely intermixed with calculations utilizing built-in facilities and previous extensions so that the interaction is of the full incremental variety, *a la* LISP and APL, rather than a semi-interactive style, *a la* BASIC.

I have spent many fascinating hours using the four most actively supported and publicized LISP based systems and it seems likely that increasing numbers of students, scientists, engineers, and mathematicians will want an opportunity to try some of these systems. Consequently, the following four sections briefly describe some of their capabilities and their availability, in order of increasing size. In the interest of brevity, each section emphasizes features not described in previous sections.

As with many other LISP programs, these computer algebra systems seem almost magical when first encountered. Thus, it is especially satisfying and educational to learn how they work. Accordingly, these four sections also briefly indicate some of the underlying techniques, together with the issues that they address.

Interest in computer algebra is growing rapidly, and the final section discusses the impact that this powerful tool can have on education, recreation, and research.

## muMath-79

muMATH-79 is a small computer-algebra system implemented by Albert Rich and the author for Intel 8080 based microcomputers using the popular Digital Research CP/M operating system. The system will also run on the upward-compatible Intel 8085 and Zilog Z-80 processors, and upward-compatible operating systems such as the Cromemco CDOS or IMSAI-IMDOS systems. In its entirety, including an allowance of 5.7 K bytes for a resident operating system, the system occupies 28 K bytes, for which an additional minimum of 16 K bytes is recommended to store the control stack, the symbol table, character strings, numbers, expressions, and user-defined functions. The system is modular so that users can save space by omitting unneeded packages. For example, the symbolic integration, differentiation, logarithmic, trigonometric, and inverse trigonometric packages can be omitted when one is interested only in algebra. Similarly, the algebra and rational arithmetic packages can also be omitted when one is interested only in exact integer arithmetic. Here is a brief summary of the built-in facilities:

- The system provides indefinite precision rational arithmetic, including integer factorization and simplification of fractional powers. For example, the system can perform the simplification:

$$\frac{\sqrt{18}-\sqrt{8}}{\sqrt{6}} \rightarrow \frac{1}{\sqrt{3}}$$

where → denotes *is transformed to*.

- Unavoidable automatic algebraic simplifications include collection of similar terms, collection of similar factors, reduction of integer powers of the imaginary number i, and exploitation of the identity properties of 0 and 1, such as:

$$1*u \rightarrow u$$

for any expression u.

- Optional, more drastic automatic algebraic transformations include expansion of integer powers of sums, expansion of products of sums, factoring common factors from all the terms of a sum, placing expressions over a common denominator, and distribution of denominators over the terms of corresponding numerators. Optional transformations are controlled by the values of a few option variables so that users can employ or suppress these more drastic transformations to suit their needs and tastes for each specific problem. Unavoidable and optional automatic logarithmic transformations include:

$$e^{ln(u)} \rightarrow u,$$
$$ln(e^u) \rightarrow u,$$
$$ln(u*v) \rightleftarrows ln(u) + ln(v),$$
$$ln(u \uparrow v) \rightleftarrows v*ln(u),$$

for all u and v.

- Unavoidable and optional automatic trigonometric transformations include exploitation of symmetry to remove minus signs from trigonometric arguments, exact computation for angles which are integer multiples of $\pi/12$, multiple angle expansion, angle-sum expansion, conversion of trigonometric powers to multiple angles, and conversion of trigonometric products to angle sums.
- Symbolic differentiation and integration rules are built-in for all of the built-in mathematical operators and functions. Also, there is a mechanism for introducing differentiation and integration rules for other operators and functions defined by the user.

As an example of the speed of muMATH, on an 8080 running at 2 MHz with 48 K bytes the system can expand 298!, $(1+x)^{20}$, $\sin(17x)$, $(x_1+x_2+\ldots+x_{13})^2$, or $\sin(x_1+x_2+\ldots+x_5)$ in one minute. Try doing these by hand!

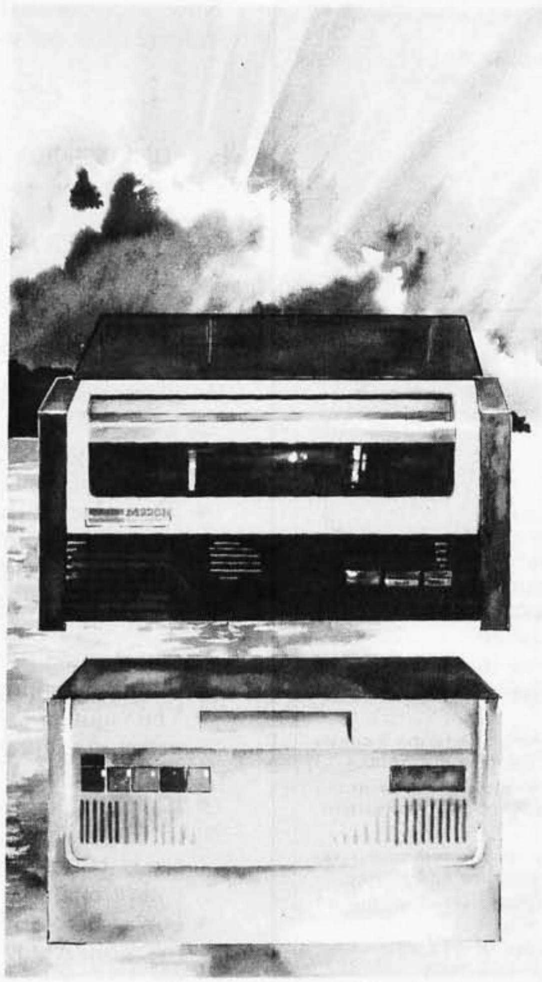
Because of the incremental expression-oriented style, a knowledge of computer programming is unnecessary for using the built-in capabilities of muMATH in the symbolic-calculator fashion. When a user's needs are not met by the built-in facilities, they can be modified or extended by entering appropriate function definitions, simplification rules, or operator parse rules. The built-in

# Expansive but not expensive

## MicroAge introduces for **Horizon** and **Alpha Micro** Systems, the most powerful Hard Disk Units You've Ever Seen.

Welcome to "Hard" Times: a new age of hard disk cartridge drives for Horizon and Alpha Micro . . . so powerful, so reliable, so **economical**, you've never seen anything like it. Fast, easy and versatile: the revolutionary new **Fujitsu M2201** with 40 (formatted) megabytes of storage . . . and the dynamic CDC **Phoenix** with 27 (formatted) megabytes. Both are operated with North Star and Alpha Micro commands. Added capacity means the ability to fully utilize the complete range of software and capabilities of North Star Horizon and Alpha Micro mainframes.

But their family interface compatibility is not all that's expansive. Speed and economy are impressive, too. Super-fast access time means no waiting for command execution . . . easy handling of **large** files. And the price? Unbelievably low for this much flexibility, power, speed and capacity. And the low price includes the hard disk drive, S-100 controller, software interface, cords and disk pack.



**Exclusively from MicroAge**

**Fujitsu M2201 Cartridge Module Drive**

50 megabyte storage
(40 formatted)

**CDC Phoenix Cartridge Module Drive**

32 megabyte storage
(27 formatted)

Either.

# $9995.00

Includes drive, S-100 controller, software interface and disk pack.

Ask your dealer about Hard Times — the Fujitsu M2201 and CDC Phoenix drives from MicroAge. Available to qualified dealers and OEM's. If a dealer is not available in your area, call 800-528-1415.

Circle 212 on inquiry card.

mathematical algorithms are written in the same general environment and high-level syntax provided to the user. Consequently, the user does not need to master a second underlying environment and syntax, such as LISP, in order to understand the system and fully integrate his extensions into the system.

As an example of a functional extension, suppose that we wish to introduce the trigonometric cosecant function named CSC, together with the automatic transformation:

$$CSC(u) \rightarrow 1/SIN(u)$$

for any expression u. To accomplish this, we merely enter the definition:

```
FUNCTION CSC(U),
   1/SIN(U),
   ENDFUN;
```

Thereafter, until the function is redefined, the above transformation will automatically occur for the CSC of any expression.

Now, suppose that as the sole exception to the above transformation, we wish to introduce the transformation:

$$CSC(0) \rightarrow UNDEFINED$$

where UNDEFINED is a variable. To accomplish this, we merely enter the new definition:

```
FUNCTION CSC(U),
   WHEN U=0, UNDEFINED EXIT,
   1/SIN(V)
   ENDFUN;
```

As illustrated by these two examples:

- The body of a function definition consists of a sequence of expressions separated by commas.
- The value returned when a function definition is applied to its arguments is the value of the last expression evaluated therein.
- A *conditional exit* expression consists of the *matchfix* operator named WHEN, followed by one or more expressions separated by commas, followed by the matching delimiter named EXIT.
- The value of a conditional exit is that of the last expression evaluated therein when the conditional exit is evaluated.
- If the first expression in a conditional exit evaluates to FALSE, then the exit fails and evaluation proceeds to any successive expression following the conditional exit.
- For a successful exit, proceeding sequentially from the nonFALSE expression, when evaluation first reaches an EXIT delimiter it proceeds to the point following the next ENDFUN, ENDLOOP, or ENDBLOCK delimiter.

To illustrate the LOOP construct, suppose that we wish to define a function which uses repeated first derivatives to compute the Nth partial derivative of an expression EXPN with respect to a variable VAR, for any specific integer N>0. We could do so as follows:

```
FUNCTION DIFN(EXPN, VAR, N),
   LOOP
      EXPN←DIF(EXPN, VAR),
      WHEN N=1, EXPN EXIT,
      N←N−1
      ENDLOOP
   ENDFUN;
```

As illustrated by this example:

- A loop-expression consists of the matchfix operator named LOOP, followed by zero or more expressions separated by commas, followed by the matching delimiter named ENDLOOP.
- Even an assignment is an expression, having as its value the value assigned.
- A loop can contain any number of conditional exits anywhere in the loop, thus providing a single structured generalization of the REPEAT, WHILE, and halfloop constructs of some languages.

Moreover, when a function definition is applied to fewer arguments than there are parameters, the extra parameters are initialized to FALSE and they are available for use as local variables within the definition.

An alternative *recursive* definition of DIFN is:

```
FUNCTION DIFN(EXPN, VAR, N),
   WHEN N=0, EXPN EXIT,
   DIFN(DIF(EXPN, VAR), VAR, N−1)
   ENDFUN;
```

As is frequently the case, the recursive version is more compact, and compactness is important on small computers.

The *block* control-construct consists of the matchfix operator named BLOCK, followed by a conditional exit, then zero or more arbitrary expressions, then the matching delimiter named ENDBLOCK. The value of a block is the value of the last expression evaluated therein. A block can contain any number of conditional exits interspersed among other expressions, thus providing a structured generalization of the case-statement of some other languages, including the IF-THEN-ELSE construct as a special instance.

Some users may want to extend the syntax by introducing additional mathematical operators or additional programming control-constructs. The incrementally-extendable Pratt parser makes it easy to introduce such extensions as they are needed.

Every operator can have a left and a right binding power. For example, the left and right binding powers of / are 120, whereas ↑ has a left binding power of 140 and a right binding power of 139. When two operators are competing for an operand between them, the operator with higher binding power toward the operand wins the operand (eg: the expression X/Y↑2 is parsed the same as X/(Y↑2) rather than (X/Y)↑2). When there is a tie, the operator on the left wins the operand (eg: X/Y/2 is parsed the same as (X/Y)/2 rather than X/(Y/2 ).

*Prefix* operators precede their operands. For example, to establish COS as a prefix operator so that we can omit parentheses from around suitable arguments of COS, we can enter the command:

PROPERTY COS PREFIX 170.

Then, COS X/Y parses the same as COS(X)/Y, because 170 exceeds 120. Alternatively, we could enter the command:

PROPERTY COS PREFIX 119

if we wished COS X/Y to parse the same as COS (X/Y).

*Postfix* operators follow their operand, *infix* operators lie between their operands, and *matchfix* operators (such as LOOP) precede an arbitrary number of operands separated by commas and delimited by a matching delimiter (such as ENDLOOP). Numbers and variable names parse as themselves. A functional expression parses into a list containing the function name followed by the parsed representations of its arguments. An operational expression parses into a list containing the name of the operator followed by the parsed representations of its operand. As an example, COS(2/N!) parses into the nested list (COS,(/,2,(!,N))).

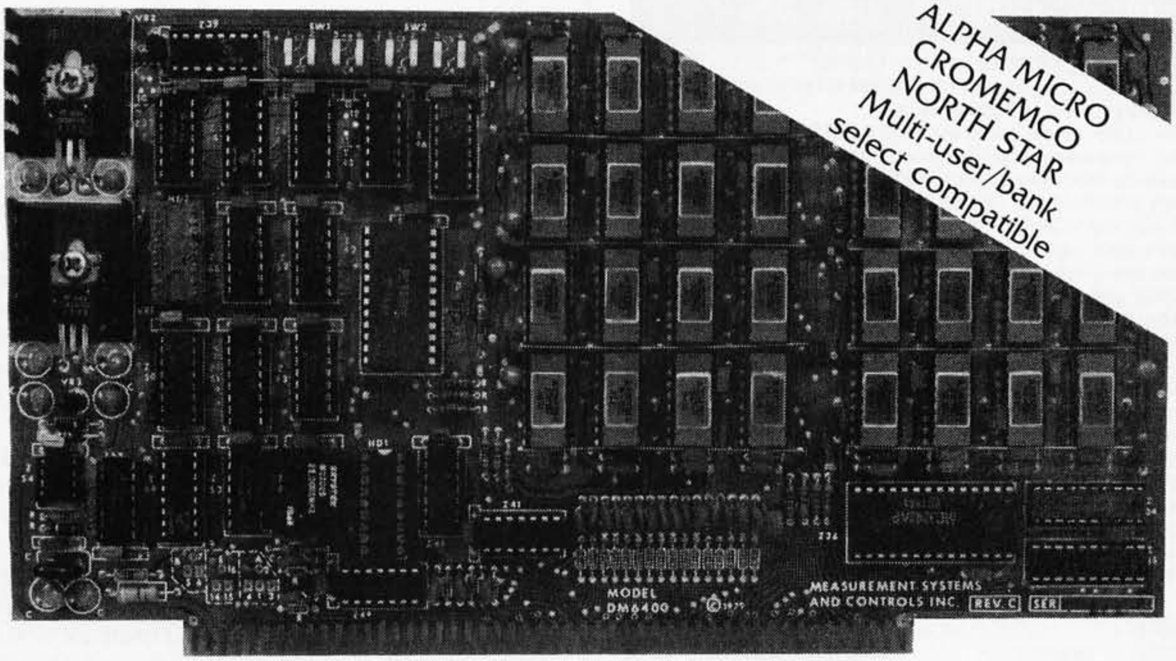
In general, this representation is called *Cambridge prefix* (as opposed to Polish prefix or ordinary functional prefix). We are all so accustomed to infix notation that most people find mathematical Cambridge prefix tiresome to read, and many people also find it tiresome to write. However, the parser prevents us from having to write Cambridge prefix, and the deparser prevents us from having to read it, in order to enjoy its great advantages as an internal representation. These advantages are many.

In order for our programs to determine simply and quickly which transformations to apply to expressions, the programs must be able to easily determine whether the expressions are numbers, variables, or more general. If the latter, the program must be able to easily determine the outermost operator or function name, and easily access the individual associated operands or arguments. To keep the transformation programs fast and compact, the syntactic rules governing the internal representation should be few and simple. Moreover, it is sometimes convenient to regard expressions as *data* in order to *apply* transformations to them. At other times it is convenient to regard expressions as *programs* in order to *execute* them. Cambridge prefix offers all of these advantages.

For each cycle of interaction, after parsing the input expression, muMATH merely applies the built-in LISP like EVAL function, then deparses the result for output. For computer-algebra it is appropriate for such an EVAL function to at least do the following:

- Evaluate numbers and unbound variables as themselves.
- Evaluate bound variables as the values to which they are bound.
- Evaluate a list for which the first element is the name of a function definition as the value obtained by applying the function definition to the values of the other elements in the list.
- Otherwise, the value of a list is the list of its values.

Unfortunately, most LISP EVAL functions implement only a subset of these rules, leaving undefined the result




**Model DMB-6400 Series dynamic 64k byte RAMS incorporate the features which are standard in the DM-6400 Series and adds the following capabilities:**

- ALPHA MICRO, CROMEMCO, and NORTH STAR output port bank select compatible.
- Four (4) 16k byte, functionally independent memory banks.
- Memory bank size can be incremented to 64k bytes in 16k increments.
- Eight (8) 64k byte banks of memory per output port.

**Model DM-6400 Series dynamic 64k memory boards feature IEEE S-100 compatible timing and on board transparent refresh.**

- Memory selectable and deselectable in 4k byte increments.
- 25 MHz on board crystal oscillator for independent timing.

**DMB-6400 and DM-6400 Common Features:**

- 4 MHz Z80 operation with no wait states.
- Tested and burned-in.
- Low power- 8 watts maximum.
- Reliable, expandable memories.

**ONE YEAR GUARANTEE**

THE FOLLOWING PRODUCTS ARE AVAILABLE FROM YOUR LOCAL DEALER.

☐ DMB-6400/64K RAM ☐ DMB-3200/32K RAM ☐ DM-6400/64K RAM ☐ DM-3200/32K RAM
☐ DMB-4800/48K RAM ☐ DMB-1600/16K RAM ☐ DM-4800/48K RAM ☐ DM-1600/16K RAM

U.K. & EUROPEAN REPRESENTATIVE:
ABACUS COMPUTERS LTD.
62, NEW CAVENDISH STREET
LONDON, W1M 7LD U.K.
TEL: 01-580/8841 TELEX: 881-3085

AUSTRALIAN REPRESENTATIVE:
COMPUTERLAND OF MELBOURNE
555 COLLINS STREET
MELBOURNE, VIC3000
TEL: 625581

MEASUREMENT
systems & controls
incorporated

867 North Main Street • Orange, CA 92668
Telephone: 714/633-4460

of applying EVAL to an unbound variable or a list whose first element is not the name of a function definition. In computer algebra, no one would want to quote every instance of every unbound variable. It is often desirable to write subexpressions such as $f(x)$, even though there is no corresponding function definition named $f$. Accordingly, most LISP based algebra systems begin by defining an algebraic EVAL function in terms of the built-in LISP EVAL function.

Since muMATH is intended for microcomputers, we did not want to waste precious space on two near-duplicate EVAL functions so we included the above upward-compatible generalizations of the usual LISP EVAL in one EVAL. These generalizations are convenient in other LISP applications, so we would like to see LISP evolve in this direction.

The lexical and syntactic rules appropriate for input and output of LISP and computer-algebra expressions also differ. Many LISP implementations do not directly accept special characters such as + as valid names, and LISP scanners do not distinguish between blanks and commas. Again, we did not want to waste precious space on two sets of I/O (input/output) routines, one of which would never be employed by users of the computer-algebra system. Accordingly, using assembly language, two semantically similar but lexically and syntactically different general-purpose list-processing systems were implemented: muLISP-77 which implements the traditional LISP lexical and syntatic rules, and muSIMP-77 which employs the lexical rules and high-level syntax illustrated in the preceding examples. We used muSIMP-77 to implement muMATH-79, but muSIMP-77, being a disguised version of LISP, is applicable wherever LISP is applicable. We think that beginners are more comfortable with muSIMP than with LISP, hence they are more willing to learn the lovely semantics of LISP, and to ultimately appreciate the Spartan syntactic simplicity of LISP, together with its consistency between program and data.

To illustrate the convenience of Cambridge prefix as an internal representation, here is an example of how differentiation could have been implemented in muMATH:

```
FUNCTION DIF(EXPN, VAR),
   WHEN EXPN=VAR, 1 EXIT,
   WHEN ATOM(VAR), 0 EXIT,
   WHEN FIRST(EXPN =’+,
           DIF(SECOND(EXPN),VAR)
           +
           DIF(THIRD(EXPN),VAR)
           EXIT,
   WHEN FIRST(EXPN)= ’* ... EXIT,

   . . .

   WHEN FIRST(EXPN)=LN,
           DIF(SECOND(EXPN),VAR)
           /SECOND(EXPN)
           EXIT,
   LIST(DIF,EXPN,VAR)
   ENDFUN;
```

The built-in function named ATOM returns TRUE if its argument is a number or a name. The built-in functions named FIRST, SECOND, and THIRD, respectively, return the indicated elements of the list which is their argument. The function named LIST takes any number of arguments returning a list of their values. As indicated, a single quote is used in contexts where one wishes to prevent the parser from seeking operands for a name which happens to be an operator.

In simplified results the operators + and * have two or more operands which have been sorted into a lexical order to facilitate collection of similar terms and factors. Consequently the above example would have to use a loop or recursion to march down the list of operands of +.

For modularity and other reasons, differentiation and most other mathematical transformations are implemented with the aid of a sort of *pattern matcher*. The following sections illustrate pattern-matching techniques.

## Reduce

REDUCE is a LISP based computer-algebra system implemented by Anthony Hearn and his colleagues for a variety of large computers. Currently there are supported implementations for the PDP-10, PDP-20, IBM360, IBM370, Univac 1108, CDC Cyber, and Cray-1 machines, running under various popular operating systems. In its entirety, the system occupies about 400 K bytes on an IBM370, for which an additional minimum of at least 50 K bytes is recommended as workspace. The system is modular so that users can save space by omitting unneeded packages (eg: 100 K bytes can be saved by omitting the integrator). For those who have access to the ARPA computer network, REDUCE is available at several sites, including USC-ECL and SU-AI, where accounts may be obtainable. REDUCE is also directly available on magnetic tape from Professor Hearn at the University of Utah Computer Science Department in Salt Lake City for $100. It has been distributed to over 500 sites worldwide. Here is a brief summary of the built-in facilities:

- The system provides single-precision floating-point arithmetic as well as indefinite-precision rational arithmetic.
- Unavoidable algebraic transformations and optional ones controlled by flags are approximately similar to those of muMATH, except that REDUCE provides an important additional optional transformation: cancellation of polynomial greatest divisors from the numerators and denominators of rational expressions. REDUCE can perform such simplifications as the following:

$$\frac{2a^2x^2-a^2bx-a^2b^2-ax^3+axb^2-x^4+bx^3}{a^2x^2-a^2b-ax^2-2axb-ab^2-bx^2+b^2x} \to \frac{2ax+ab+x^2}{a+b},$$

which might be overlooked by most people.
- There are some built-in exponential, logarithmic and trigonometric simplifications.
- Matrices having symbolic expressions as elements can be added, subtracted, multiplied, divided and raised to integer powers, including inversion
- There are special facilities for solving the quantumelectrodynamics problems of the high-energy physics.

- There is a high-level surface programming language, which is essentially ALGOL, sweetened by modern control constructs such as a WHILE loop, REPEAT loop, and CASE statement.
- Symbolic differentiation and integration are built-in, and the latter is significantly more powerful than the muMATH integrator, which merely uses a few elementary rules such as:

$$\int(u+v)dx \rightarrow \int udx + \int vdx,$$

$$\int c\ u\ dx \rightarrow c\int u\ dx \text{ if } c=\text{constant},$$

$$\int v\ f(u)dx \rightarrow \frac{v}{du/dx}\int f(u)du \text{ if } \frac{v}{du/dx} = \text{constant},$$

$$\int x^{-1} \rightarrow \ln x,$$

$$\int x^{\alpha} \rightarrow \frac{x^{\alpha+1}}{\alpha} \text{ if } \alpha=\text{const and} \neq -1,$$

$$\cdots$$

$$\int \sin(x)dx \rightarrow \cos(x).$$

In contrast, extensive greatest-common-divisor, factorization, and linear-equation-solving support routines permit REDUCE to use the powerful new Risch-Norman integration algorithm. For a large class of integrands and solution basis functions, this algorithm is guaranteed to determine a closed-form solution if one exists, otherwise terminating with a guarantee that one does not exist.

- REDUCE provides a convenient pattern matcher, which provides a natural means for users to implement many extensions. To have the system automatically replace every subsequent instance of $mc^2$ by E, we can merely enter the rule:

```
LET M*C**2 = E;
```

Thereafter, an expression such as 5*M*C**3+8 would be replaced automatically by 5*E*C+8. There is also a mechanism for letting pattern variables represent arbitrary subexpressions. To make logarithms of all powers, products and quotients can be expanded automatically, we can enter the rules:

```
FOR ALL X, Y LET
   LOG (X**Y) = Y*LOG(X),
   LOG(X*Y) = LOG(X) + LOG(Y),
   LOG(X/Y) = LOG(X) - LOG(Y);
```

Thereafter an expression such as A+2* LOG(B) − LOG(E**A*B**2*C) would simplify to −LOG(C). Finally, there is a mechanism for imposing extra prerequisites to replacements. To make the above LOG rules dependent upon the value of an option variable, we could change the first line to:

```
FOR ALL X,Y SUCH THAT LOGEXPAND > 0
```

Most of REDUCE is written in a modular subset of itself called RLISP. In turn, RLISP is bootstrapped from standard LISP, which is a subset of many LISP implementations. RLISP has the semantics of LISP clothed in the syntax of sweetened ALGOL. RLISP is applicable not only to computer algebra, but also wherever LISP is applicable, and I have found students far more receptive to LISP if they are introduced to it via a surface language such as RLISP.

REDUCE was originally inspired by a desire to perform symbolic high-energy-physics computations which are far too arduous to do manually. Consequently, the internal representations of expressions reflect a major concern with speed and storage efficiency for large expressions:

- In applied math, the most numerous operations in very large expressions are usually addition, subtraction, multiplication, and exponentiation with positive integer exponents. There is frequently, at most, one division operation present, because expressions are often put over a common denominator. If fractional powers, exponentials, logarithms, trigonometric functions or other irrational operations occur, they may usually be reduced to numerous repetitions of a few unnested distinct irrational functions having trivial arguments such as x, x+y or $2\pi x$. Thus, polynomial operations account for most of the time and space. This suggests using a data structure oriented toward polynomials, thereby saving space and time by making the operators $+$, $\times$, and $\uparrow$ implicit. This usual nature of large expressions also suggests storing irrational subexpressions uniquely, and treating them as additional *variables* with respect to any polynomial operations involving them.
- As the number of variables and their maximum degrees increase, a multivariate polynomial must have zero as a sharply increasing portion of its possible terms, in order to fit the polynomial into the computer memory. Moreover, the fit is possible only if the internal representation takes advantage of this *sparsity*. In general, we can avoid wasting space on intermediate-degree terms which are zero only if we explicitly store the exponents of the nonzero terms.
- Many multivariate polynomial algorithms are most concisely stated as univariate algorithms, recursively involving coefficients which are polynomials in at least one less variable.
- Classic multivariate polynomial division requires that one variable be distinguished as the leading variable and that the terms be accessible in decreasing order of degree.

REDUCE uses Cambridge prefix for some purposes, but REDUCE internally represents polynomials in a *standard form*. A standard form is defined as an element from the underlying coefficient domain or as a *leading term* dotted with a *reductum*, where the latter is recursively defined as a standard form of lower degree in the main variable of the leading term. The underlying coefficient domain can be indefinite-precision integers, indefinite-precision rational numbers, integers modulo some modulus, or single-precision floating-point numbers. A leading term is defined as a *leading power* dotted with a *leading coefficient*, where the latter is recursively defined as a standard form not containing the main variable of the leading power. A leading power is defined as the *main variable* dotted with the *leading degree*, where the latter is a positive integer. In Backus-Naur form, we can summarize this definition as follows:

```
standard form ::= domain element
              ::= LT standard form. + RED standard form
RED standard form :: = standard form
LT standard form
     :: = LPOW standard form .* LC standard form
LC standard form :: = standard form
LPOW standard form
  :: = MVAR standard form.** LDEG standard form
```

I have also taken the opportunity to introduce the REDUCE infix *constructor macros* named .+, .*, .**, which clearly indicate the implied operator, but are all defined as merely the LISP CONS operation. Similarly, I have introduced the mnemonic prefix REDUCE prefix *selector* macros named LT, RED, LPOW, LC, MVAR, and LDEG, which are respectively defined as the LISP functions CAR, CDR, CAAR, CDAR, CAAR, and CDAAR.

With this representation and macros, the REDUCE multivariate polynomial addition function definition is extraordinarily compact and elegant — an ideal intermediate level example of *reductum recurso*. Listing 1 below shows this reduce function, expressed in RLISP.

```
SYMBOLIC PROCEDURE ADDF(U,V);
   IF ADDITIVEIDENTITY U THEN V
   ELSE IF ADDITIVEIDENTITY V THEN U
   ELSE IF DOMAINP U THEN ADDD(U,V)
   ELSE IF DOMAINP V THEN ADDD(V,U)
   ELSE IF LPOW U = LPOW V THEN
      ADDFF(ADDF(LC U, LC V),
            ADDF(RED U, RED V ))
   ELSE IF ORDPP(LPOW U, LPOW V) THEN
      LT U .+ ADDF(RED U, V)
   ELSE LT V .+ ADDF(U, RED V);

SYMBOLIC PROCEDURE ADDD(D,V)
   IF ADDITIVEIDENTITY V THEN D
   ELSE IF DOMAINP V THEN ADDDM(D,V)
   ELSE LT V .+ ADDD(D, RED V);

SYMBOLIC PROCEDURE ADDFF(F1, F2);
   IF ADDITIVEIDENTITY F1 THEN F2
   ELSE IF ADDITIVEIDENTITY F2 THEN F1
   ELSE LPOW U .* F1 .+ F2;
```

In listing 1, use has been made of the ADDITIVE-IDENTITY prefix recognizer macro which tests for a zero, the DOMAINP prefix recognizer macro which tests for the underlying coefficient domain, the ORDPP predicate which tests the relative order of two leading powers, and the ADDDM function which adds domain elements. Since the syntax is essentially ALGOL, for which descriptions are widely available, we leave the serious reader to ponder this example, moving on now to another computer algebra system.

## MACSYMA

MACSYMA is a very large computer-algebra system implemented by the Mathlab group at the MIT Laboratory for Computer Science in Cambridge MA. The system will probably be made available for DEC PDP-10 computers in a year or two.

In its entirety, excluding the library of user-submitted routines, MACSYMA occupies 400,000 36 bit words on the PDP-10. The system is modular, starting with a nucleus of 100,000 words. As is perhaps implied by its name, MACSYMA provides more built-in math operations than any other computer-algebra system. Here are some highlights:

- The system provides arbitrary-precision floating-point as well as indefinite-precision arithmetic.
- Besides the usual unavoidable algebraic transformations, there are numerous optional automatic ones controlled by flags or which are employed by applying specific functions to expressions. The most sophisticated of these transformations include cancellation of polynomial greatest common divisors, partial-fraction decomposition, nested polynomial decomposition such as completion of powers, and factorization. For example, MACSYMA can perform the factorization:

$$3w^2z^6 + 2w^3z^4 + 114xy^2z^3 - 10w^2y^2z^3 + 45w^2x^3z^3 - 3w^2z^3 + 76wxy^2z - 2w^3z - 380xy^4 + 1710x^4y^2 + 10w^2y^2 - 45w^2x^3 \rightarrow (3z^2 + 2wz - 10y^2 + 45x^3)(w^2y^3 + 38xy^2 - w^2).$$

- There are numerous built-in transformations for fractional powers, exponentials, logarithms, trigonometric functions, inverse trigonometric functions, hyperbolic functions, and inverse hyperbolic functions. There are also transformations for some higher transcendental functions such as the error, gamma, beta, zeta, and psi functions.
- There is built-in matrix algebra on matrices having unspecified elements and unspecified size.
- There are special facilities for series analysis of periodic phenomena such as orbits.
- There is a high-level surface programming language which resembles ALGOL, with evidence of meta-LISP influence.
- There is a powerful pattern-matching facility and an extendable Pratt parser.
- Symbolic differentiation and integration are built-in. The latter employs a powerful Risch algorithm, among other techniques. There is also a distinct program for definite integrals, which employs contour integration and other techniques besides indefinite integration.

- There is a powerful function which employs L'Hospital's rule and other techniques to computer limits.
- There are powerful functions for determining infinite and truncated generalized power-series expansions of expressions.
- Laplace transforms and their inverses are built-in.
- There is a function which uses a variety of techniques to seek closed-form solutions to first-order and second-order ordinary differential equations.
- There is a built-in function which uses the powerful new Gosper algorithm to find closed forms for sums with indefinite or infinite summations limits. For example, the function is able to make the transformation:

$$\sum_{j=0}^{n} \frac{j^4 4^j}{\binom{2j}{j}} \to \frac{2(n+1)(63n^4+112n^3+18n^2-22n+3)4^n}{693\binom{2n}{n}} - \frac{2}{231}$$

- Equations are legitimate expressions. Two equations or an equation and a nonequation can be added, multiplied, etc, and there is a powerful function named SOLVE which uses a variety of techniques to seek solutions to one or more simultaneous linear or nonlinear equations. SOLVE is able to determine, as exact symbolic expressions involving c, the four values of x which satisfy the quartic equation:

$$x^4 = cx + 1.$$

  As another example, SOLVE is able to determine that the exact solutions for the two simultaneous nonlinear equations:

$$z^4 + x^2z^2 + xz^2 + y^2 + x^3 = 2yz^2 + x^2y + xy,$$
$$yz^2 + 2xyz + xy = 2xz^3 + 2x^2z + y^2,$$

  are the curve ($x=r$, $y=s^2$, $z=r$) together with the surface ($x=r$, $y=s^2+r$, $z=s$), where r and s are arbitrary parameters.
- There is an extensive user-contributed program library which includes packages for vector and tensor analyses, ordinary and variational optimization, solution of integral equations, higher transcendental functions, and dimensional analysis.

Most of MACSYMA is written in MACLISP, which is a particularly elaborate version of LISP also developed at MIT. MACSYMA uses several internal representations, including Cambridge prefix and a recursive polynomial representation somewhat like that of REDUCE. The major difference from the REDUCE polynomial representation is that in MACSYMA the variables are also implicit and stored separately, only once per complete polynomial. This usually saves additional space in the expressions. Although the resulting algorithms are somewhat faster when combining polynomials having the same variables, there is some awkwardness or overhead involved in a preliminary padding phase when combining polynomials that do not have identical variables.

## SCRATCHPAD

SCRATCHPAD is a very large computer-algebra system implemented at the IBM Thomas J Watson Research Center. It is available there on an IBM 370, and it is available from other IBM corporate sites via telephone. Regrettably, this fine system has not yet been released to the public, but it is discussed here because of its novel features.

In its entirety, the system occupies about 1600 K bytes on an IBM 370 with virtual storage, for which an additional minimum of 100 K bytes is recommended for workspace. The variety of built-in transformations currently lies between that of REDUCE and MACSYMA. However, each of the three systems has features that none of the others possess, and one of these features may be a decisive advantage for a particular application. Here are some highlights of the SCRATCHPAD system:

- The system provides single-precision floating-point arithmetic as well as indefinite-precision rational arithmetic.
- The built-in unavoidable and optional algebraic transformations are approximately similar to those of MACSYMA.
- The built-in exponential, logarithmic, and trigonometric transformations are approximately similar to those of REDUCE.
- Besides built-in symbolic matrix algebra, APL like array operations are included, and they are even further generalized to permit symbolic operations of nonhomogeneous arrays and on arrays of indefinite or infinite size.
- Symbolic differentiation and integration are built-in, with the latter employing the powerful Risch-Norman algorithm.
- There is a particularly elegant built-in facility for determining Taylor series expansions.
- There is a built-in SOLVE function capable of determining the exact solution to a system of linear equations.
- There is a powerful pattern-matching facility which serves as the primary mechanism for user level extensions. The associated syntax is at a very high level, being the closest of all computer-algebra systems to the declarative, nonprocedural notation of mathematics. To implement the trigonometric multiple-angle expansions, we can merely enter the rewrite rules:

```
cos(n*x) = 2*cos(x)*cos((n-1)*x)-
              cos((n-2)*x), n in (2,3,...), x arb
sin(n*x) = 2*cos(x)*sin((n-1)*x)-
              sin((n-2)*x), n in (2,3,...), x arb
```

Then, whenever we subsequently enter an expression such as cos(4*b), the response will be a corresponding expanded expression such as:

$$8 \cos^4(B) - 8 \cos^2(B) + 1$$

Thus, programs resemble a collection of math formulae much as they would appear in a book or article.

- SCRATCHPAD has a particularly powerful yet easily used mechanism for controlling the output format of expressions. For example, the user can specify that an expression be displayed as a power series in x, with coefficients which are factored rational functions in b and c, etc. For large expressions, such fine control over the output may mean the difference between an important new discovery and an incomprehensible mess.

This generalized recursive format idea is so natural and effective that SCRATCHPAD is now absorbing the idea into the internal representation. A study of the polynomial additional algorithm in the previous section reveals that it is written to be applicable to any coefficient domain which has the algebraic properties of a *ring*. The coefficients could be matrices, power-series, etc. That coefficient domain could in turn have yet another coefficient domain, and so on. With a careful modular design, packages to treat each of these domains can be dynamically linked together so that code can be shared and combined in new ways without extensive rewriting and duplication. Then not only the output, but also the internal computations can be selected most suitably for a particular application.

For further information about SCRATCHPAD, contact Richard Jenks at the IBM Thomas J Watson Research Center, Yorktown Heights NY 10598.

**The Future**

If the preceding sections have whet your appetite for more information about computer algebra, try some of the survey articles, collections of articles, and relevant books listed in the bibliography. Also, annual membership in the ACM Special Interest Group on Symbolic and Algebraic Manipulation costs a mere $2.50 for students, $5 for other ACM members, or $8 otherwise. Membership includes a subscription to the *SIGSAM Bulletin*, which contains the latest information about relevant meetings, reports, and developments.

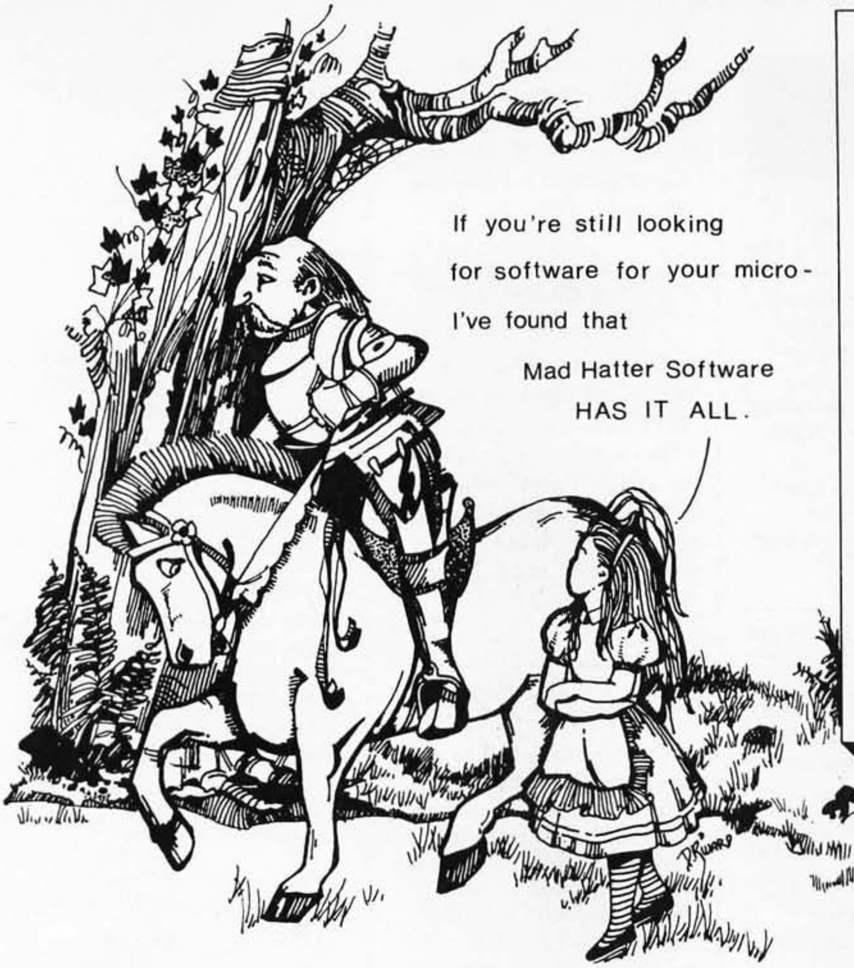
Computer algebra is increasingly available on a wide variety of processors ranging in size from the Intel 8080 microprocessor to the Cray 1 supercomputer. Within a short while computer algebra should be economically and conveniently accessible to most engineers, scientists, mathematicians, students, and hobbyists. This widespread availablity will have a profound effect on research utilizing applied math, math education, computer education, and recreational math. Consider the following:

- How frequently approximate numerical computations are employed without first checking to see if a more informative analytical solution is obtainable with the help of computer algebra.
- How many mistakes in manual analytical analyses could be caught by checking the derivations with computer algebra.
- How little of elementary-school through university math education is concerned with floating-point arithmetic.
- How much of this education is concerned with the kind of arithmetic and symbolic transformations provided by computer algebra, or concerned with theorem proving, which is especially well supported by other LISP programs.
- How dramatically computer algebra demonstrates the utility of LISP like languages, providing numerous well-motivated examples for teaching such languages.
- How much more students and enthusiasts are intrigued by artificial intelligence and game playing application of computers than by accounting and floating-point scientific applications.

The conclusion is inescapable: computer algebra and LISP like languages provide an ideal first exposure to computer programming, and are an invaluable component of scientific programming skills. ■

Bibliography


1. Aho, A V, Hopcroft, J E, and Ullman, J D, *The Design and Analysis of Computer Algorithms*, Addison Wesley Publishing Co, 1975.
2. Borodin, A, and Munroe, I, *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, 1975.
3. Brown, W S, and Hearn, A C, "Applications of Symbolic Algebraic Computation," Comp Phys Comm, (forthcoming).
4. *Communications of the ACM*, August 1966.
5. *Communications of the ACM*, August 1971.
6. Jenks, R, (editor), *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation*, ACM Inc, 1976.
7. *Journal of the ACM*, October 1971.
8. Knuth, D E, *The Art of Computer Programming, Volume 1, Basic Algorithms*, Addison Wesley Publishing Co, 1967.
9. Knuth, D E, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*, Addison Wesley Publishing Co, 1968.
10. Petrick, S R (editor), *Proceedings of the Second Symposium on Symbolic Manipulation*, ACM Inc, 1971.
11. *Proceedings of 1974 Eurosam Conference, ACM SIGSAM Bulletin 8*, August 1974.
12. *Proceedings of the 1977 MACSYMA Users Conference*, NASA CP2012, June 1977.
13. *Proceedings of the Second MACSYMA Users Conference*, MIT Laboratory for Computer Science, June 1979.
14. *Proceedings of the 1979 Eurosam Conference*, Springer-Verlag, (forthcoming).
15. *SIAM Journal on Computing*, June 1979.
16. The Soft Warehouse, POB 11174, Honolulu HA 96828, distributes the author's muMATH-79 source code free to those who obtain muSIMP-77. Object listings of the latter or of muLISP-77 cost $85, and a machine readable version costs an additional $95. Primers and reference manuals are available separately for each of these systems at a cost of approximately $0.10 per page.




# Business & Application Software Available

- Client Billing
- A/R • A/P
- Gen. Ledger
- Inventory
- Payroll
- Mailing List
- File Handling

for

TRS-80 APPLE
&
OTHERS

Call or Write for Details

# MAD HATTER SOFTWARE

| PROGRAM | DESCRIPTION | TRS-80 L1/2 | TRS-80 L2 | PET | APPLE | |
|---|---|---|---|---|---|---|
| GALACTIC BLOCKADE RUNNER | • AN EXCITING SPACE WAR GAME WITH GRAPHICS | • | | • | • | $ 9.95 |
| SCI-FI GAME SAMPLER | • 3 GAMES—LUNAR LANDER—STAR MONSTER—SPACE BATTLE | • | | • | • | $ 7.95 |
| R/T LUNAR LANDER | • A REAL TIME LUNAR LANDER WITH GRAPHICS | | • | | | $ 7.95 |
| MICRO-TEXT EDITOR | • FORMAT TEXT—SAVE & LOAD TO TAPE—OUTPUT TO PRINTER | | • | | | $ 9.95 |
| OTHELLO III | • A STRATEGY BOARD GAME—PLAY AGAINST COMPUTER OR OTHERS | • | | • | • | $ 7.95 |
| AIR RAID | • A REAL TIME, ARCADE TYPE SHOOTING GAME IN MACH. LANG. | • | | | | $ 14.95 |
| MICRO-CHESS | • PLAY CHESS WITH YOUR COMPUTER—VARIOUS LEVELS OF DIFF. | • | | • | • | $ 19.95 |
| BRIDGE CHALLENGER | • DON'T WAIT FOR OTHERS TO PLAY—YOUR COMPUTER'S READY | • | | • | • | $ 14.95 |
| APPLE 21 | • BLACKJACK WITH HIRES GRAPHICS | | | | • | $ 9.95 |
| STAR WARS/SPACE MAZE | • SCI-FI GAMES FOR THE APPLE | | | | • | $ 12.95 |
| RENUMBER | • RENUMBER YOUR BASIC PROGRAMS—RENUMBERS EVERYTHING | • | | | • | $ 14.95 |
| DISK RENUMBER | • SAME AS ABOVE, BUT ON DISK | • | | | | $ 19.95 |
| PILOT 2.0 | • THE EDUCATIONAL LANGUAGE, IN MACH. LANG.—INC. EDITOR | | • | | | $ 14.95 |
| PILOT 3.0 | • THE DISK VERSION OF THE ABOVE | | • | | | $ 24.95 |
| APPLE TALKER | • YOUR APPLE SPEAKS! NO NEW HARDWARE REQUIRED | | | | • | $ 15.95 |
| APPLE LIS'NER | • SPEECH RECOGNITION THE EASY WAY—GREAT WITH THE TALKER | | | | • | $ 19.95 |
| TIC-TAC-TALKER | • TIC-TAC-TOE USING SPEECH SYNTHESIS AND RECOGNITION | | | | • | $ 19.95 |
| SYSCOP | • MAKE BACKUP TRS-80 SYSTEM TAPES THE EASY WAY | | • | | | $ 9.95 |
| ANDROID NIM-2 | • GAME OF NIM WITH ANIMATED ROBOTS AND SOUND | | • | | | $ 14.95 |
| SNAKE EGG | • A BETTING GAME WITH ANIMATED SNAKES AND SOUND | | • | | | $ 14.95 |
| LIFE 2 | • 100 GEN. PER MIN. LIFE & BATTLE OF LIFE W/ANIMATION & SOUND | | • | | | $ 14.95 |
| DCV-1 | • PUT SYSTEM TAPES ON DISK EVEN IF IN SAME MEM AS DOS | | • | | | $ 9.95 |
| MUSIC MASTER | • ENTER SHEET MUSIC—THE TRS-80 THEN COMPILES & PLAYS IT | | • | | | $ 14.95 |
| DISK MUSIC MASTER | • SAME AS ABOVE BUT ON DISK W/MANY SELECTIONS | | • | | | $ 24.95 |
| TRS-80 CP/M | • OPENS UP THE WHOLE WORLD OF CP/M SOFTWARE TO THE TRS-80 | | • | | | $150.00 |

**10% OFF IF YOU ORDER 3 SOFTWARE PACKAGES OR MORE SEND FOR FREE CATALOG—GIVE TYPE OF COMPUTER**

TO ORDER BY PHONE OR FOR DEALER INFO—CALL—(617) 682-8131
ADD 75c SHIPPING & HANDLING • MASS. RESIDENTS ADD 5% SALES TAX
**MAD HATTER SOFTWARE** • 900b SALEM RD • DRACUT, MA 01826

master charge VISA

**AVAILABLE FROM THESE FINE MICRO COMPUTER DEALERS**

CAPITOL COMPUTER SYSTEMS
3396 EL CAMINO AVE.
SACRAMENTO CA 95821

TRS-80 SOFTWARE EXCHANGE
17 BRIARCLIFF DR.
MILFORD NH 03055

OP AMP TECH BOOKS
1033 N. SYCAMORE AVE.
LOS ANGLES CA 90038

COMPUTER CABLEVISION, INC.
2617 42ND ST. NW #2
WASHINGTON DC 20007

KENNEDY SYSTEMS
74 BROAD ST.
LYNDONVILLE VT 05851

ADVANCED COMPUTER PRODUCTS
1310B E. EDINGER
SANTA ANA CA 92705

HOBBY WORLD ELECTRONICS
19355 BUSINESS CENTER DR. #6
NORTHRIDGE CA 91324

AUGUST AUTOMATION
28 MILK ST.
WESTBORO MA 01581

THE CPU SHOP
39 PLEASANT ST.
CHARLESTOWN MA 02129

J&J ELECTRONICS LTD
28 COLLINGTON AVE.
BEXHILL-ON-SEA, E. SUSSEX, ENG.

COMPUTER VILLAGE
931 SW 87TH AVE.
MIAMI FL 33174

L. C. SALES
100 HINCHEY AVE. #705
OTTAWA, ONT., CAN. K1Y4L9

Circle 202 on inquiry card.

# BYTE's Bugs

## Marsport, Here I Come

Delmer Hinrichs has found several corrections which should be made to "Marsport, Here I Come" (April 1979 BYTE, page 84):

- page 90, step 4 should be "$x \leq y$?" Since there is no "$x < y$?" step available on the HP67/97, users could probably correct this.
- page 90, step 25 should be "ST I." Since there is no "ST 1" (only "STO 1"), this is probably correctable by users.
- page 90, steps 119 and 120 must be reversed. Users might be able to figure this out by noting other similar conversions.
- page 92, step 182 should be "$-x-$" (print/pause), not "X" (multiply). This could probably be figured out from the program operating instructions and flow diagram. In any case, if you get here, you're going to crash.
- page 92, step 204 should be "GSB C," not "GSB c". This error is disastrous, as it causes the spaceship to materialize at the center of Mars.

## Don't Share Your Soap

An acronym was wrongly interpreted in "History of Computers: The IBM 650" by Keith S Reid-Green (March 1979 BYTE, page 238.) The name of the SOAP assembler program is properly derived from the phrase "symbolic optimal assembly program," not "SHARE optimum assembly program," as was stated. Thanks to Leo Walder of Greenbelt MD for pointing this out.

## A Bug on the Beam

There was a bug in the labeling of figure 10 on page 49 of Steve Ciarcia's Circuit Cellar article "Communicate on a Light Beam" (May 1979 BYTE). The center tapped transformer should have been labelled as 24 V instead of 18 V.

## Tic Tac Bug

Delmer Hinrichs has discovered a small bug in the program for "Tic-Tac-Toe: A Programming Exercise" (May 1979 BYTE, page 196). Line number 340 should end with 3,2,5,7,9 rather than 2,3,5,8,9.

In addition, BASICs other than TDL 8 K might have to write:

```
230 RANDOM
```

instead of:

```
230 F=RND(-1)
```

to initialize the random number generator. Lines 465 and 570 might have RND(1) replaced with RND(0) to give a random number between 0 and 1.

## A Bug in the Field

John P Costas has informed us that several errors crept into listing 1 of "Cryptography in the Field" (April 1979 BYTE, page 145). The locations and the correct code are given below.

| Location | Code |
|---|---|
| 70 | STO-5 |
| 90 | STO-8 |
| 111 | STO-9 |
| 122 | — |
| 178 | STO-3 ■ |

# Event Queue

## AUGUST 1979

*August 1-3*

**Microcomputer Applications**, Southern Technical Institute, Marietta GA. The emphasis of this seminar will be on the applications of microcomputers in industry. Software, hardware and interfacing techniques will be discussed. Contact Dr Richard L Castellucis, Southern Technical Institute, Electrical Engineering Technology Dept, 534 Clay St, Marietta GA 30060.

*August 6-8*

**Pattern Recognition and Image Processing,** Hyatt Regency Chicago O'Hare, Chicago IL. This conference is sponsored by the Machine Intelligence and Pattern Analysis Committee of the IEEE Computer Society. The program will consist of submitted and invited papers, and a large trade show of graphics and image processing equipment. Contact PRIP 79, POB 639, Silver Spring MD 20901.

*August 6-10*

**SIGGRAPH '79**, Chicago IL. This sixth annual conference on computer graphics will feature tutorials, technical sessions and an exposition of state-of-the-art computer graphics and image processing equipment. Contact Maxine D Brown, SIGGRAPH '79 Exposition, Hewlett-Packard, 19400 Homestead Rd, Cupertino CA 95014.

*August 6-10*

**Modern Communication Systems: Analysis and Design**, University of Southern California, Los Angeles CA. This course is devoted to the analysis and design of modern communication systems, with emphasis on the derivation of practical design equations useful for trade-off studies and overall synthesis. Contact University of Southern California, Continuing Engineering Education, Los Angeles CA 90007.

*August 6-10*

**Advanced Microcomputer System Development: High Level Languages, Technology Trends, and Hands-On Experience**, University of Southern California, Los Angeles CA. This course is intended to present the participants with a clear picture of the microcomputer revolution, provide hands-on programming experience using extended BASIC and FORTRAN, analyze technology trends in the microcomputer field, and assess the impact of VHSI/VLSI. Contact University of Southern California, Continuing Engineering Education, Los Angeles CA 90007.

*August 8-10*

**SIGPLAN Symposium on Compiler Construction**, Boulder CO. This symposium will consider methods of, and experience with, constructing compilers. The emphasis will be less on theoretical methods and more on techniques applied to real compilers. Contact Professor Leon Osterweil, Dept of Computer Science, University of Colorado, Boulder CO 80309.

*August 8-10*

**First Annual Conference on Research and Development in Personal Computing**, Hyatt Regency Chicago O'Hare, Chicago IL. This conference is sponsored by the Association for Computing Machinery (ACM) Special Interest Group on Personal Computing (SIGPC). A large trade show of personal computer and graphics equipment is planned to accompany an assortment of papers, panels, user group meetings, workshops, and person to person poster booths. Contact Bob Gammill, Computer Science Division, Dept of Mathematical Sciences, 300 Minard Hall, North Dakota State University, Fargo ND 58102.

*August 13-15*

**Minicomputers and Distributed Processing**, Atlanta GA. This three day seminar will examine the uses, economics, programming, and implementation of minicomputers. Contact the University of Chicago, Center for Continuing Education, 1307 60th St, Chicago IL 60637.

*August 13-15*

**Conference on Simulation, Measurement and Modeling of Computer Systems** Boulder CO. This conference will feature performance prediction techniques employed during the design, procurement and maintenance of computer systems. It will provide a forum for both applied and theoretical work in the disciplines of performance monitoring, modeling, and simulation of computer systems. Contact Gary Nutt, Xerox PARC, 3333 Coyote Hill Rd, Palo Alto CA 94304.

*August 13-16*

**Q-GERT Network Modeling and Analysis**, Ramada Inn, Lafayette IN 47905. This course will provide the attendee with the information necessary to model complex systems using Q-GERT. Emphasis will be on the procedures for modeling and analysis. Contact Pritsker and Associates Inc, POB 2413, W Lafayette IN 47906.

*August 13-17*

**High Speed Computation: Vector Processing**, The University of Michigan, Ann Arbor MI. In this course, the architectural, software, and algorithmic issues of vector architecture are coordinated by discussion of concepts in computer architecture and detailed study of current vector processors and their use. Contact Engineering Summer Conferences, 400 Chrysler Center, North Campus, The University of Michigan, Ann Arbor MI 48109.

*August 19-22*

**International Conference on Computing in the Humanities**, Dartmouth College, Hanover NH. This conference is intended to foster computer research and technique in all areas of humanistic study; to promote international cooperation in the development of programs, data banks, and equipment; and to make the results of research available. The program will include a plenary session each evening and shorter sessions during the day. Contact Stephen V F Waite, Kiewit Computation Center, Dartmouth College, Hanover NH 03755.

*August 19-24*

**1979 Symposium for Innovation in Measurement Science**, Hobart and William Smith Colleges, Geneva NY. Sponsored by the Scientific Instrumentation and Research Division of the Instrument Society of America, scheduled sessions at this symposium include innovation in computers and electronics, mass flow measurement, chemical analysis, applied analysis in instrument control, physical analysis, medical instrumentation, and advances in industrial measurement. Contact Instrument Society of America, 400 Stanwix St, Pittsburg PA 15222.

*August 22-24*

**Understanding and Using Computer Graphics**, San Francisco CA. This course is for people who are using, or are contemplating using computer graphics and would like to understand its role in their organization. It will describe computer graphics, explain the available hardware and software systems, and give cost and performance comparisons. Contact Frost and Sullivan, 106 Fulton St, New York NY 10038.

*August 23-26*
**National Small Computer Show**, New York Coliseum, New York NY. Exhibitors will include major manufacturers, distributors, and publications in the small computer field. A lecture series will include topics of interest to business and professional people, hobbyists, and the general public. Contact National Small Computer Show, 74 E 56th St, New York NY 10022.

## SEPTEMBER 1979

*September 4-6*
**International Conference and Exhibition on Engineering Software**, University of Southampton, England. The aim of this conference is to provide a forum for the presentation and discussion of recent advances in engineering software and to present a state-of-the-art in this field. An exhibition, held in conjunction with the conference, will cover all software products, services, and equipment related to engineering software. Contact Dr R Adey, Engsoft, 6 Cranbury Place, Southampton SO2 OLG, ENGLAND.

*September 4-7*
**Compcon Fall'79**, Capital Hilton Hotel, Washington DC. This eighteenth IEEE Computer Society International conference will present the latest developments in microprocessor architecture, support software, operating systems, and peripheral devices. Contact IEEE Computer Society, POB 639, Silver Spring MD 20901.

*September 5-8*
**Info/Asia**, Ryutsu Center, Tokyo. This exposition will be devoted to information management, computers, word processing, and advanced business equipment. The exposition will be accompanied by a four day conference. Contact Clapp and Poliak Inc, 245 Park Ave, New York NY 10017.

*September 18-20*
**Wescon/79**, St Francis Hotel, San Francisco CA. Contact Electronic Conventions Inc, 999 N Sepulveda Blvd, El Segundo CA 90245.

*September 24-26*
**Minicomputers and Distributed Processing**, New York NY. See August 13-15 for details.

*September 25-27*
**WPOE '79**, San Jose Convention Center, San Jose CA. This show will be dedicated to word processing and office/business equipment, services and materials. Complementing the exhibit will be a three day executive conference program that focuses on emerging technologies and their applications in the office. Contact Cartlidge and Associates Inc, 491 Macara Ave, Suite 1014, Sunnyvale CA 94086.

*September 26-29*
**MIMI '79**, Queen Elizabeth Hotel, Montreal, Canada. This symposium is intended as a forum for the presentation and discussion of recent advances in mini and microcomputers and their applications. Special emphasis will be given to the theme of the conference: "The Evolving Role of Minis and Micros Within Distributed Processing." Contact The Secretary, MIMI '79 Montreal, POB 2481, Anaheim CA 92804.

*September 28-30*
**Northeast Personal and Business Computer Show**, Hynes Auditorium, Boston MA. Displays and exhibits will showcase microcomputers and small computer systems of interest to businesspeople, hobbyists, professionals, etc. Lectures and seminars will be presented for all categories and levels of enthusiasts, including introductory classes for novices. Contact Northeast Exposition, POB 678, Brookline MA 02197.

*Text continued on page 200*

**SIMULATIONS**

**3-D Tic Tac Toe** by Scott Adams. Three skill levels -- author warns you to practice before tackling computer's third skill level. I or II, 16K $7.95

**Star Trek III.3** by Lance Micklus. One of the most advanced Star Trek games ever written. Level II, 16K $14.95.

**End Zone** by Roger W. Robitaille, Sr. Authentic football simulation, right down to the 2-minute warning. Level I or II, 16K $7.95

**Cribbage** by Roger W. Robitaille, Sr. You versus the computer cribbage played by standard rules. Level I or II, 16K, $7.95.

**Bridge Challenger** by George Duisman. You and the dummy play 4-person contact bridge against the computer. Level II, 16K $14.95

**'Round the Horn** by Rev. George Blank. You're the captain of a clipper ship racing from New York to San Francisco. Level II, 16K $9.95

**Concentration** by Lance Micklus. One of the most popular television games. Level I or II, 16K $7.95

**Safari** by David Bohlke. You're in the running for a film contract at a major Hollywood studio. To qualify, you must photograph the most wild animals in their natural habitat. Level II, 16K $7.95.

**Pork Barrel** by Rev. George Blank. Places you in the shoes of an aspiring Congressman. Level II, 16K $9.95

**Backgammon** by Scott Adams. Level II, 16K $7.95

**Chess Companion** by M. Kelleher. Combines chess clock features with ability to record your moves while action is fast and furious. Level II, 16K $7.95

**Sargon Chess** by Dan & Kathe Spracklen. Winner of the 1978 San Jose Microcomputer Chess Tournament. Level II, 16K $19.95

**Mastermind II.2** by Lance Micklus. Lets you and the computer take turns making and breaking codes. Level II, 16K $7.95

**PERSONAL**

**RPN Calculator** by Russell Starkey. A self-documenting calculator program. Uses Reverse Polish Notation with 4-level stack, 100 memories, scientific functions. Level II, 16K $9.95

**Home Financial Management** by M. Kelleher. Turns your computer into a personal financial advisor. Level II, 16K $9.95

**Tarot** by Frank B. Rowlett, Jr. Probably the best future-gazing type program ever written. Try it -- you'll like it! Level I or II, 16K $9.95

**Ham Radio** by M. Kelleher. Amateur Frequency Allocations, ID Timer, Q-signal File, Amateur Log Routine, Propogation Forecasting. L II, 16K $9.95. **Special Disk-enhanced version, 32K $24.95**

**Educator Assistant** by Steve Reisser. Five programs of value to educators. Compute percentage, individual student averages, class averages, standard test scores, final grades. L II, 16K $9.95 D, $14.95

**Electronic Assistant** by John Adamson. A group of 9 subprograms designed to solve problems such as tuned circuits and active and passive filters. L II, 16K $9.95

**Personal Finance** by Lance Micklus. 33 different budgets can be easily adapted by user to fit his individual needs. A 2-part program, entry and search. Level II, 16K $9.95

**Advanced Personal Finance** by Lance Micklus. Same as above with advanced analysis routine. **Supports Disk Files** D, 32K $24.95

**TsE TRS-80 Software Exchange**
**1-603-673-5144**
17 BRIAR CLIFF DRIVE MILFORD, NEW HAMPSHIRE 03055

# MAGAZINE SECTION NEW!

## SoftSide™

SoftSide is for pioneers . . . those hardly souls who have adopted a TRS-80, installed it in their living-room or office, and unleashed their imaginations.

SoftSide helps you discover the endless variety of tasks your new friend will do for you, as you build a unique partnership of human being and machine.

We publish software for the partners. Every month we publish games, household application programs, educational aids, business programs. We help you realize your expectations, fantasies, and dreams.

SoftSide means Software!

## PROG/80

A bi-monthly magazine for the serious programmer who wants to know HOW his computer works and WHY. PROG/80 emphasizes technique rather than canned programs. The subjects include machine language, construction projects and specialized applications software, not just for the advanced computer hobbyist, but for the computerphile who wants the most from his machine.

## Apple Seed

If you're an Apple II pioneer, you've been longing for a software publication and hoping someone would get around to it.

We have. Apple Seed is to the Apple II what SoftSide is to the TRS-80. And it's brand new. The first issue will roll off the press in August or September. Apple II enthusiasts will eat up this special introductory offer!

**DEALER INQUIRIES INVITED**

For further details call:
603-673-5144

**SOFTSIDE**
☐ 1 Year - 12 issues $18.00

**PROG/80**
☐ 1 Year - 6 issues $20.00

**APPLE SEED**
☐ 1 Year - 12 issues $15.00

**PO Box 68 Milford, NH 03055**

☐ USA first class $25.00 - 1 yr.
☐ APO/OVERSEAS surface $25 - 1 yr.
☐ CANADA/MEXICO $25 - 1 yr.
☐ OVERSEAS airmail $30 - 1 yr.

**Exp. Date** ____________ **Interbank # [M/C only]** ____________
**Signature** ____________
**Name** ____________
**Address** ____________
**City** ____________ **State** ________ **Zip** ____________

Telephone orders accepted for Master Charge or VISA accounts. Call Monday through Friday, 9:30 to 5:30 EST at **603-673-5144**

## OCTOBER 1979

*October 1-3*
**Second Annual Symposium on Small Systems**, Hilton Inn, Dallas TX. The symposium will consist of a blend of paper and panel discussions with major emphasis on microcomputer applications. Both hardware and software topics presenting state-of-the-art and state-of-the-industry aspects will be included. Contact Gerald Kane, Southern Methodist University, Dallas TX.

*October 2-4*
**NEPCON Central '79**, O'Hare Exposition Center, Rosemont IL. This tenth annual exhibition and conference of electronic and microelectronic packaging and production equipment will feature displays of electronic and microelectronic materials, hardware, tools, supplies and test instruments. Contact Industrial and Scientific Conference Management Inc, 222 W Adams St, Chicago IL 60606.

*October 14-17*
**International Data Processing Conference and Business Exposition**, Town and Country Hotel, San Diego CA. Contact Data Processing Management Association, 505 Busse Highway, Park Ridge IL 60068.

*October 15-18*
**Sixth Information Management Exposition and Conference**, New York Coliseum, New York NY. Contact Clapp and Poliak Inc, 245 Park Ave, New York NY 10017.

*October 15-19*
**CPEUG 79**, San Diego CA. This is the fifteenth meeting of the Computer Performance Evaluation Users Group sponsored by the National Bureau of Standards. Contact Judith G Abilock, The Mitre Corp, Metrek Div, 1820 Dolley Madison Blvd, McLean VA 22102.

*October 16-18*
**Understanding and Using Computer Graphics**, Washington DC. See August 22-24 for details.

*October 21-23*
**New York State Association for Educational Data Systems Annual Conference**, Granit Hotel, Kerhonksen NY. The theme of this conference is "Instructional Computing — Hardware/Software/Courseware." Contact Mary E Heagney, 9201 Shore Rd, Brooklyn NY 11209.

*October 22-24*
**Computers in Aerospace Conference II**, Hyatt House Hotel, Los Angeles CA. The conference theme, "Computer Technology for Space and Aeronautical Systems in the Eighties," will be carried out by a series of panels, invited presentations, and contributed papers which will bring computer system technologists together with specialists in the application of embedded computers in space and aeronautics. Contact American Institute of Aeronautics and Astronautics, 1290 Ave of the Americas, New York NY 10019.

*October 22-25*
**ISA/79**, O'Hare Exposition Center, Chicago IL. The conference theme, "Instrumentation for Energy Alternatives," will emphasize current practices in instrumentation design and implementation. Contact Instrument Society of America, 400 Stanwix St, Pittsburgh PA 15222.

*October 28-30*
**The Tenth North American Computer Chess Championship**, Detroit Plaza, Detroit Michigan. Sponsored by the Association for Computing Machinery, this is a four round, Swiss style tournament, with the first two rounds to be played on October 28th (1 PM and 7:30 PM), the third on October 29th (7:30 PM), and the final round on Tuesday, October 30th (7:30 PM). Contact Monroe Newborn, McGill University, School of Computer Science, 805 Sherbrooke St W, Montreal PQ, CANADA H3A 2K6.

*October 29 - November 2*
**Applied Interactive Computer Graphics**, University of Maryland, College Park MD. This course is designed to cover the most important facets of graphics that are necessary to develop general graphic applications. Systems considerations including configuration selection criteria, and the pros and cons of off-the-shelf software are stressed. The most important factors and techniques are described for hardware, software, and geometric modeling. Contact UCLA Extension, 10995 Le Conte Ave, Los Angeles CA 90024.

*October 30 - November 1*
**Interface West**, Anaheim Convention Center, Anaheim CA. This third annual West Coast small computer and office automation systems conference and exposition will feature over 100 company exhibits and 60 conference sessions covering a variety of data processing, word processing, data communications, management hardware, software, and service topics. Contact the Interface Group, 160 Speen St, Framingham MA 01701.

# Clubs and Newsletters

### Sacramento Microcomputer Users Group

According to *Push & Pop*, the newsletter of the Sacramento Microcomputer Users Group, this organization meets the fourth Tuesday of every month at 7:30 PM at the SMUD Training Facilities on 59th St. Their mailing address is POB 161513, Sacramento CA 95816.

### Northwest Computer Society Meets Twice a Month

The Northwest Computer Society meets at Seattle University in the Library Auditorium, Room 115. The University is on 12th Ave between E Madison St and E Cherry St. Meetings are held the first and third Thursday of each month at 7:30 PM. The first meeting of the month usually features a formal presentation by a speaker or speakers. The second meeting is usually more informal with free-wheeling discussion and problem solving. Membership in the Northwest Computer Society, which includes the impressive *Northwest Computer News*, is $7. For more information, write the club at POB 4193, Seattle WA 98104, or call (206) 284-6109 for recorded information.

### The Computer Hobbyist Group of North Texas

*The Printed Circuit* is a well organized, informative newsletter published by The Computer Hobbyist Group of North Texas. In a recent issue there were reports from various user groups within the club, a list of coming attractions, a reprint of an article about the Tandy and Texas Instruments' race for the home computer, an S-100 bus article, new products, and more. *The Printed Circuit* may be obtained by joining the group at a rate of $7 per year. Dues should be sent to Warren Bean, 2405 Briarwood, Carrollton TX 76006.

### Denver Amateur Computer Society

The Denver Amateur

Computer Society has recently increased the printing of their newsletter, *Interrupt*, to 1,000 copies, and has begun over-the-counter distribution at local computer stores. First class mailings of the newsletter will be restricted to paid members only. The club meets the third Wednesday of the month at 7:30 PM at 1380 S Santa Fe, Denver CO. Many user groups within the club meet at different times and locations. For further information, write to Mike Dymtrasz, president of the society, at the above address.

## Computers in Psychiatry and Clinical Psychology

*Computers in Psychiatry/Psychology* (formerly *Micro-Psych*), a bi-monthly newsletter for professionals interested in the use of computers in psychiatry and clinical psychology, is beginning its second year of publication. It addresses itself in an informal, scientific style to clinical users of the computer. Three pages of each issue are devoted to a description of the computer related activities of subscribers. Each issue of the 13 page newsletter contains summaries and reviews of recently published articles and books as well as an ongoing bibliography and a program catalogue. Recent additions include a clearinghouse for information on training opportunities in the field and a new hardware column. Subscriptions to Volume 2 can be obtained by sending $15 to *Computers in Psychiatry/-Psychology*, 26 Trumbull St, New Haven CT 06511. *The Best of Micro-Psych* — Volume 1, a 52 page compilation of articles and information from Volume 1, is also available for $12.

## The New England Computer Society

The New England Computer Society meets on the first Wednesday of each month to exchange computer hobbyist information and sponsor activities. The NECS is the oldest and one of the largest clubs in the Boston area, with over 200 members. Within the club are 8080, 6502, TRS-80, 6800, PET, Apple and Digital user groups. The meetings start at 7 PM and are held at the Mitre Corp cafeteria, Route 6, east of Route 3, Bedford MA. For additional information, write to the New England Computer Society, POB 198, Bedford MA 01730.

## Heath Company Newsletter

*Buss* is an independent newsletter of Heath Company computers. It contains Heath product information and user reports. The price for 12 issues is $8 ($10 overseas). Contact Charles Floto, 325 Pennsylvania Ave SE, Washington DC 20003.

## Publication for the Computer Professional

*The Data Processing Digest (DPD)* is written for the computer professional and the manager who uses computer technology for planning, control and production. The editors of *DPD* regularly search through numerous business and industrial periodicals and reports to locate articles on all aspects of computer technology and its application to operations and management. Concise summaries of these articles, reviews of books on data processing, and listings of current professional meetings and seminars appear in each issue. The subscription rates are $57 for one year; $108 for two years; and $153 for three years. Contact Data Processing Digest Inc, 6820 La Tijera Blvd, Los Angeles CA 90045.

# A Message to our Subscribers

From time to time we make the BYTE subscriber list available to other companies who wish to send our subscribers promotional material about their products. We take great care to screen these companies, choosing only those who are reputable, and whose products, services, or information we feel would be of interest to you. Direct mail is an efficient medium for presenting the latest personal computer goods and services to our subscribers.

Many BYTE subscribers appreciate this controlled use of our mailing list, and look forward to finding information of interest to them in the mail. Used are our subscribers' names and addresses only (no other information we may have is ever given).

While we believe the distribution of this information is of benefit to our subscribers, we firmly respect the wishes of any subscriber who does not want to receive such promotional literature. Should you wish to restrict the use of your name, simply send your request to BYTE Publications Inc, Attn: Circulation Department, 70 Main St, Peterborough NH 03458. Thank you.

### Australian Tandy Users Club and Software Exchange

*8th Bit* is the main medium by which Software Exchange members keep informed of what is happening in Australia. This newsletter contains information on what is for sale and the location,contributions from members, and information of general significance. Membership in the Exchange is $10 per year. Contact Pitt St Microcomputer Centre, Second Floor, 373-375 Pitt St, Sydney 2000 AUSTRALIA.

### Detroit Personal Computer Network

Andrew Fellman has written to inform us that the Detroit Personal Computer Network will be meeting in August. This organization was formed to help microcomputer users discover and exchange ideas on user projects, to promote business or financial gain, and for enjoyment. More information may be obtained by writing to Andrew at 13043 McNichols, Detroit MI 48219, or calling (313) 865-4374.

### Software of the Month Club

Creative Discount Software has announced the opening of its new Software of the Month Club. The new club will have separate branches for users of the Apple II, TRS-80, Ohio Scientific, Exidy, PET and CP/M based systems. Members will select division memberships such as business applications, education applications, high level languages, games and fun applications, and personal and home management applications. Membership enrollment applications are available from Creative Discount Software, Software of the Month Department, POB 24-B-67, Los Angeles CA 90024.

### The Physicians Microcomputer Report

The *Physicians Microcomputer Report* is a monthly publication for doctors who wish to become better informed about the computer and its application in the field of medicine. Some of the features include software news, calculator corner, computers in patient health care, microcomputer hardware news, the bargain market, and computer articles of special interest to the physician. Additionally, the report contains articles on nonmedical applications such as linking your computer to a stock portfolio information center. Another intent of this publication is to facilitate the exchange of information between physicians who own computers. For this purpose, the magazine has a listing of user groups.

The *Physicians Microcomputer Report* is available for $25 a year, $12.50 for students. Contact Dr Gerald M Orosz, POB 6483, Lawrenceville NJ 08648.

## BYTE's Bits

### Call for Papers

The International Society for Mini and Microcomputers (ISMM) will hold an international symposium on microcomputers and their application January 30 to February 1 1980 in Monterey CA. The symposium will highlight technology, hardware, software engineering, languages, systems architecture, design methodology, computer networks, performance evaluations, concurrent processing, real time processing, operating systems, portability for software systems, systems security, digital signal processing, education, and applications. Send three camera ready copies of 200 word abstracts to Secretary, MIMI-80 (Monterey), POB 2481, Anaheim CA 92804 by September 1 1979. Notification of acceptance will be sent by October 1. Camera ready copies of accepted papers are due December 15 1979. Additionally, proposals for half day and one day tutorials are solicited in the above areas and should be received by September 1 1979.

## Exidy to Sponsor Software Contest

Exidy Inc, the makers of the Sorcerer microcomputer, are sponsoring a contest for microcomputer programs this summer. Four Sorcerer computers will be awarded as grand prizes. The purpose of the contest is to encourage people who have written good programs to share their programs with other computer owners. Exidy will publish a book featuring the best programs entered in the contest. The contest is open to all BASIC language computer programs which will run on the Sorcerer. Prizes of free computers will be awarded to the program judged best in each of four categories: business, education, fun and games, and home and personal management. Every entrant will receive a free poster and a professionally written program in exchange for the program they submit. The contest runs from June 1 thru August 31 1979. For further information, contact Paul Terrell, Marketing Communications, Exidy Inc, 969 W Maude Ave, Sunnyvale CA 94086.

## Department of Missing Authors

Once again an author of a yet-to-be-published article has moved and neglected to inform us of his new address. We therefore request that James Cherry, whose last known address was 28 The Fenway, Boston MA 02215, please contact us with his current address and telephone number.

## Call for Papers for Fifth International Conference on Computer Communications

Technical papers for the Fifth International Conference on Computer Communications to be held October 27 thru 30 1980 in Atlanta GA are being solicited for presentation at the regular conference sessions and publication in the official proceedings. The conference is held biannually by the International Council for Computer Communications as an interdisciplinary forum for discussing social, economic, political and technological implications of computer communication networks.

Topics for 1980 may include a wide range of subjects and issues relevant to the development and use of computer communications and its effect on human affairs. All papers must be original, written and presented in English, and cannot exceed 5,000 words. Specific suggested subjects are: broad needs and requirements, social implications, applications, and technology. Manuscripts must be typed, double spaced, and on one side of the paper only. A cover page must give the title, the full names of the author(s), the affiliation of each author, and the name, address, and telephone number of the primary author. A 100 to 200 word abstract and a full set of illustrations must accompany the manuscript.

Six copies of all material should be sent by March 1 1980 to Dr J Salz, Program Chairman, ICCC '80, Bell Laboratories 1G-509, Holmdel NJ 07733. The Program Committee would also appreciate advance notice of the intention to submit a paper. ■

# LISP Applications in Boolean Logic


Richard Weyhrauch
Stanford Artificial Intelligence Laboratory
Stanford University
Stanford CA 94305
and
Henson Graves
Dept of Mathematics
San Jose State University
San Jose CA 95192


In LISP, some data structures can be viewed two different ways, either as data or program. This feature makes LISP unique among high level languages. When seen as a program, LISP expressions can be executed and return a value: when seen as data, they may be used as arguments for other programs. This means that if we think about a LISP program as a piece of data we can write programs directly in LISP which transform them into more useful programs.

We use LISP to imitate the manipulations that are done by engineers when designing combinatorial circuits. In this sense LISP can be used as a *calculator* for Boolean logic.

The examples presented here are well known to anyone who has studied a little hardware design. The purpose of this article is to give beginners with LISP some idea of what LISP programs look like and how some interesting symbolic manipulations can be represented in a natural way using LISP. It is written primarily with novices in mind. For this reason there are some elementary remarks about how LISP actually works. The code in this article was written as examples of LISP style programming. What we have tried to do is present some programs as they might be written in existing LISP systems. Of course the style is ours.

We illustrate the use of the recursive data structures, lists and S-expressions, and the use of lambda abstraction as a control structure to facilitate recursive transformations on them.

## Combinatorial Circuits as Boolean Logic

One learns in circuit theory that combinatorial circuits, those with no feedback, may be represented as Boolean or propositional expressions. Although these are the simplest circuits that an engineer might use, this article is meant to give simple examples of how LISP can be used. For example the circuit in figure 1 is represented by the Boolean expression:

$$(\overline{X1} \wedge \overline{X2}).$$

We may view this expression as specifying a Boolean function. We may also think of this expression as a Boolean program which may be evaluated using the ordinary rules of logic. There are, of course, many different expressions which have the same behavior.

A circuit's behavior can be described by a Boolean function. The Boolean function for $(\overline{X1} \wedge \overline{X2})$ may be represented by:

| X1 | X2 | F (X1,X2) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## Representing Boolean Expressions

Both the circuit diagrams and Boolean expresions are concrete representations of an abstract data structure, which we refer to as WFFs (*well-formed propositional formulas*). In LISP we use a concrete representation of well-formed propositional formulas as lists. For example, we represent the expression:

$$(P \vee Q) \wedge R$$

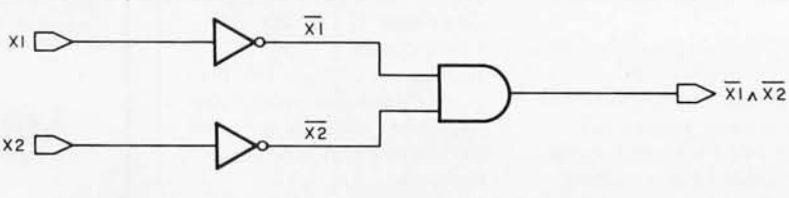



*Figure 1: A simple digital circuit whose function can be defined by the Boolean expression $(\overline{X1} \wedge \overline{X2})$*

as the list:

```
(AND (OR P Q) R)
```

We follow usual programming language practice and describe the lists which represent well-formed propositional formulas using a BNF(Backus Naur form) grammar as in table 1.

We can recognize which lists represent well-formed propositional formulas by writing a LISP program which takes a list as input and whose value is T if the list represents a WFF and NIL otherwise. This program can be viewed as a *parser* for the language generated by this grammar. It has a recursive definition which parallels the grammar:

```
(DEFINE ISWFF (E)
(COND ((ISCONST E) T)
      ((ISVAR E) T)
      ((ISUNARY E) (ISWFF (body E)))
      ((ISBINARY E) (AND (ISWFF (lhs E))
                         (ISWFF (rhs E)))
      (T NIL) )) )
```

The subfunctions body, lhs, rhs, ISCONST, ISVAR, ISUNARY and ISBINARY must also be defined. Their definition reflects our specific representation of well-formed propositional formulas in LISP. For example:

```
(DEFINE ISUNARY (E)
(EQ (CAR E) (QUOTE NOT)) )
```

Evaluation of these *defining* programs has the side effect of storing the function definition in memory. Subsequently, the name ISWFF may itself be used in a program. LISP represents function application by evaluating the list whose first element is the function and the remaining elements are the arguments. Evaluating the program:

```
(ISWFF (QUOTE (AND (OR P Q) R)))
```

returns the value T.

For any expression A the evaluation of (QUOTE A) is simply A. This is how we make LISP treat A as data. Thus in the above program the argument to ISWFF is treated as data.

### Representing Boolean Programs

If we consider T as representing true and NIL as false then we can represent the usual Boolean expressions as LISP programs using COND. COND is LISP's version of IF-THEN-ELSE.

```
(DEFINE NOT (A)
   (COND (A NIL) (T T)) )

(DEFINE OR (A B)
   (COND (A T) (T B)) )

(DEFINE AND (A B)
   (COND (A B) (T NIL)) )

(DEFINE IMPLIES (A B)
   (OR (NOT A) B) )
```

```
<wff>     := <const> | <var> | <unary> | <binary>
<const>   := T | NIL
<var>     := <identifier>
<unary>   := (NOT <wff>)
<binary>  := (AND <wff> <wff>) | (OR <wff> <wff>) |
             (IMPLIES <wff> <wff>) | (EQUIV <wff> <wff>)
```

*Table 1: In LISP, list representations for WFFs (well-formed propositional formulas) are described using a Backus Naur form of grammar. In LISP, T and NIL are generally used as the constants for true and false respectively. These correspond to 1 and 0 in digital circuit diagrams.*

| (w,v) | $\overline{w}$ | $w \vee v$ | $w \wedge v$ | $w \equiv v$ | $w \supset v$ |
|---|---|---|---|---|---|
| f,f | t | f | f | t | t |
| f,t | t | f | f | f | t |
| t,f | f | t | f | f | f |
| t,t | f | t | t | t | t |

*Table 2: Examples of truth tables for Boolean algebra. For two inputs (w and v) Boolean results are shown for the negated value of w, w OR v, w AND v, equality, and implication.*

```
(DEFINE EQUIV (A B)
   (OR (AND A B) (AND (NOT A) (NOT B)) )
```

Notice that we have defined IMPLIES, and EQUIV in terms of NOT, AND, and OR. These definitions mean that well-formed propositional formulas like:

```
(AND (OR T NIL) T)
```

are valid LISP programs whose evaluation returns a truth value (ie: T or NIL). These values correspond to those determined by the usual truth table evaluation of Boolean expressions as reviewed in table 2.

For example, if in the well-formed propositional formula (AND (OR P Q) R), we replace P by T, Q by NIL, and R by T: by observing that $(t \vee f) \equiv t$ and $(t \wedge t) \equiv t$, we calculate the value of this well-formed propositional formula as T. Logicians call this kind of assignment of truth values to the atoms an *interpretation* of the well-formed propositional formula.

One question we should ask is what happens if we try to evaluate a well-formed propositional formula which contains variables rather than simply T and NIL. For example:

```
(AND (OR P Q) R))
```

will return an error message saying that P is an undefined variable.

One thing we can use to make the substitution of T and NIL to these variables is the lambda construction. Evaluation of:

```
((LAMBDA (P Q R) (AND (OR P Q) R) (T NIL T))
```

will result in T.

### Viewing Programs as Data

Evaluation of a Boolean program corresponds to a *simulation* of the circuit represented by the program. We may also want to use LISP to answer questions about our circuits. We will consider two standard questions asked about programs for these circuit programs:

- When do two programs compute the same function? (analysis)
- Given an I/O (input/output) specification construct a program with this behavior. (synthesis)

### Analysis

Analysis of a program starts with the question—what is its behavioral description? One may then consider questions of efficiency. The complete input/output description is expressed by the Boolean function. Above we have called this the *function computed by* the program. In logic this function is just the set of all interpretations of the well-formed propositional formula. The Boolean function for the expression $(X \wedge \overline{Y}) \vee Z$ expressed as a table is:

| (X,Y,Z) | $(X \wedge \overline{Y}) \vee Z$ |
|---|---|
| 0,0,0 | 0 |
| 0,0,1 | 1 |
| 0,1,0 | 0 |
| 0,1,1 | 1 |
| 1,0,0 | 1 |
| 1,0,1 | 1 |
| 1,1,0 | 0 |
| 1,1,1 | 1 |

If a well-formed propositional formula, w, has n variables then there are $2^n$ interpretations. Thus the I/O table has $2^n$ entries. Complete behavioral knowledge could be obtained by making the $2^n$ possible evaluations. Often only partial behavioral knowledge is needed and this may sometimes be obtained without complete simulation.

Two programs are called *equivalent* when they compute the same function, i.e., they have the same behavior. A well-formed propositional formula which evaluates to T under all interpretations is called a *tautology*. The well-formed propositional formula (IMPLIES (AND P Q) (OR R P)) is a tautology. Two well-formed propositional formulas w1 and w2 are called *equivalent* if (EQUIV w1 w2) is a tautology. This means that w1 and w2 have the same I/O behavior. Thus for circuit programs the notion of equivalence coincides with the logic notion of equivalence.

One simple way to determine if a well-formed propositional formula is a tautology is to compute all its interpretations. This brute force technique can be improved upon by using an algorithm introduced by Quine in 1950. Our experience with the FOL project at the Stanford Artificial Intelligence Laboratory indicates that this algorithm represents considerable improvement over the listing of all cases. It is informally described as follows.

Choose one variable p and make two new expressions, one obtained by substituting t for p in the well-formed propositional formula and the other obtained by substituting f for p in the well-formed propositional formula. Take the conjunction of the two expressions, and use the following simplification rules.

*Listing 1: A program can be written to look for tautologies. Two well-formed propositional formulas are said to be equivalent if they both exhibit the same behavior.*

```
(DE TAUT (WFF) (TAUT1 (SIMP WFF)))

(DE TAUT1 (W)
 (COND
  ((ISCONST W) W)
  (T (TAUT
      ((LAMBDA (X) (MKAND (SUBST T X W)
                          (SUBST NIL X W)))
       (FIRSTVAR W)) ))))

(DE SIMP (W)
  (COND ((OR (ISCONST W) (ISVAR W)) W)
        ((ISNOT W) (SIMPNOT (SIMP (body W))))
        ((ISOR W)  (SIMPOR (SIMP (lhs W))
                           (SIMP(rhs W))))
        ((ISAND W) (SIMPAND (SIMP (lhs W))
                            (SIMP(rhs W))))
        ((ISIMPLIES W) (SIMPIMP (SIMP (lhs W))
                                (SIMP(rhs W))))
        ((ISEQUIV W)  (SIMPEQUIV  (SIMP (lhs W))
                                  (SIMP(rhs W)))) ))

(DE SIMPNOT (W) (COND ((ISFALSE W) T)
                      ((ISTRUE W) NIL)
                      (T (MKNOT W))))

(DE SIMPOR   (W1 W2) (SIMPANDOR ' OR W1 W2 W1 W2))

(DE SIMPAND (W1 W2) (SIMPANDOR ' AND W1 W2 W2 W1))

(DE SIMPIMP   (W1 W2) (SIMPOR (SIMPNOT W1) W2))

(DE SIMPEQUIV (W1 W2)
   (SIMPAND (SIMPIMP W1 W2)(SIMPIMP W2 W1)))

(DE SIMPANDOR (OP W1 W2 V1 V2)
                  (COND ((ISTRUE W1) V1)
                        ((ISTRUE W2) V2)
                        ((ISFALSE W1) V2)
                        ((ISFALSE W2) V1)
                        (T (MKOP OP W1 W2))))

(DE FIRSTVAR (W1)
       (COND ((ISVAR W1) W1)
             ((UNARY W1) (FIRSTVAR (body W1)))
             ((FIRSTVAR (lhs W1)))
             (T (FIRSTVAR (rhs W1)))) )

(DE ISIMPLIES (X) (EQ X T))

(DE ISFALSE (X) (EQ X NIL))

(DE ISNOT (X) (EQ (CAR X) (QUOTE NOT)))

(DE ISOR (X) (EQ (CAR X) (QUOTE OR)))

(DE ISAND (X) (EQ (CAR X) (QUOTE AND)))

(DE ISIMPLIES (X) (EQ (CAR X) (QUOTE IMPLIES)))

(DE ISEQUIV (X) (EQ (CAR X) (QUOTE EQUIV)))

(DE ISEQOR (X) (EQ X (QUOTE OR)))

(DE lhs (WFF) (CADR WFF))

(DE rhs (WFF) (CADDR WFF))

(DE body (WFF) (CADR WFF))

(DE MKOP (OP X Y) (LIST OP X Y))

(DE MKAND (X Y) (MKOP (QUOTE AND) X Y))

(DE MKNOT (X) (LIST (QUOTE NOT) X))

(DE ISCONST (W) (OR (EQ W T) (EQ W NIL)))
```

*Listing 1 continued on page 210*

$$
\begin{aligned}
&\overline{f} := t\\
&t \supset w := w\\
&f \supset w := t\\
&t \vee w := t\\
&t \wedge w := w
\end{aligned}
$$

$$
\begin{aligned}
&\overline{t} := f\\
&w \supset t := t\\
&w \supset f := \overline{w}\\
&f \vee w := w\\
&f \wedge w := f
\end{aligned}
$$

Repeat the branching and simplifying until all branches consist of either t or f. If all branches terminate in t, the well-formed propositional formula is a tautology, otherwise it is not. Applying the Quine algorithm to the well-formed propositional formula, $(p \wedge q) \supset (r \vee p)$ yields:

$$
\begin{gathered}
((t \wedge q) \supset (r \vee t)) \wedge ((f \wedge q) \supset (r \vee f))\\
(q \supset t) \wedge (f \supset r)\\
t \wedge t\\
t
\end{gathered}
$$

The LISP program in listing 1 represents the Quine algorithm.

The evaluation of:

(TAUT (QUOTE (IMPLIES (AND P Q) (OR R P))))

returns T. Notice we have used the Boolean functions IMPLIES, AND, and OR in these definitions.

## Synthesis

Consider the problem of synthesizing a program with its I/O behavior specified by the table:

| X | Y | F(X,Y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

This table may be represented by the list:

```
( (X    Y)
  (0    0    0)
  (0    1    1)
  (1    0    1)
  (1    1    0)  )
```

A well-formed propositional formula which has this behavior may be constructed by observing that:

$$
\begin{aligned}
F(X,Y) = 1 \text{ if either } & X = 0 \text{ and } Y = 1\\
& \text{or}\\
& X = 1 \text{ and } Y = 0.
\end{aligned}
$$

This Boolean function may be realized by the well-formed propositional formula $(\overline{X} \wedge Y) \vee (X \wedge \overline{Y})$. This well-formed propositional formula has a very special form. Well-formed propositional formulas which are

*Text continued on page 211*

*Listing 1 continued from page 209:*

```
(DE ISVAR (W) (AND (ATOM W) (NOT (NUMBERP W))))

(DE UNARY (W) (EQ (CAR W) (QUOTE NOT)))

(DE BINARY (W)
   (OR (OR (OR (EQ (CAR W) (QUOTE AND))
                (EQ (CAR W) (QUOTE OR)))
            (EQ (CAR W) (QUOTE IMPLIES)))
        (EQ (CAR W) (QUOTE EQUIV))))
```

```
(DEFINE SYNTHESIS (L)
  (mkor (REVERSE (CAR L)) (REVERSE (CDR L))))

(DEFINE mkand (V L)
  (PROG (X)
        (COND ((EQUAL (CAR L) 0) (RETURN NIL)))
        (SETQ L (CDR L))
        (SETQ X (COND ((EQUAL (CAR L) 0) (LIST (QUOTE NOT) (CAR V)))
                      (T (CAR V))))
  L1    (SETQ V (CDR V))
        (SETQ L (CDR L))
        (COND ((NULL L) (RETURN X)))
        (SETQ X
              (CONS (QUOTE AND)
                    (CONS (COND
                             ((EQUAL (CAR L) 0) (LIST (QUOTE NOT) (CAR V)))
                             (T (CAR V))) (LIST X))))
        (GO L1)))

(DEFINE mkor (V L)
  (PROG (X)
        (SETQ X (mkand V (REVERSE (CAR L))))
  L1    (SETQ L (CDR L))
        (COND ((NULL L) (RETURN X)))
        (SETQ X (CONS (QUOTE OR) (CONS (mkand V (REVERSE (CAR L))) (LIST X))))
        (GO L1)))
```

*Listing 2: A well-formed propositional formula which is a sum of products with each summand having literal factors is said to be in disjunctive normal form. Any Boolean function F(X1, . . . Xn) of n variables may be described by a well-formed propositional formula in disjunctive normal form. This program constructs a well-formed propositional formula in disjunctive normal form.*

```
(DE PN (WFF Z)
 (COND ((ATOM WFF) (COND ((ISEQOR Z) (MKNOT WFF)) (T WFF)))
       ((ISNOT WFF) (PN (body WFF) (FLIP Z)))
       ((ISEQUIV WFF)
        (MKOP Z
           (MKOP (FLIP Z)
                 (PN (lhs WFF) (QUOTE OR))
                 (PN (rhs WFF) (QUOTE AND)))

           (MKOP (FLIP Z)
                 (PN (lhs WFF) (QUOTE AND))
                 (PN (rhs WFF) (QUOTE OR)))))
       ((ISIMPLIES WFF) (MKOP (FLIP Z)
                              (PN (lhs WFF) (FLIP Z))
                              (PN (rhs WFF) Z)))
       ((ISAND WFF) (MKOP Z
                          (PN (lhs WFF) Z)
                          (PN (rhs WFF) Z)))
       ((ISOR WFF) (MKOP (FLIP Z)
                         (PN (lhs WFF) Z)
                         (PN (rhs WFF) Z))) ))

(DE FLIP (Z) (COND ((EQ Z (QUOTE OR)) (QUOTE AND)) (T (QUOTE OR))) )
```

*Listing 3: Any well-formed propositional formula may be transformed into disjunctive normal form. This recursive LISP program uses the rules described in the text to complete the transformation.*

*Text continued from page 209:*
either variables or the negation of variables are called *literals*. The above well-formed propositional formula is an example of a sum of products where the factors of each summand is a literal. A well-formed propositional formula of this type is said to be in DNF (disjunctive normal form).

The well-formed propositional formula $(\overline{X} \wedge Y) \vee (X \wedge \overline{Y})$ was constructed by looking at each row of the above table which has the value 1. For each such row we form a conjunction containing those variables with value 1 and the negation of those with value 0. We finish by taking the disjunction of all these conjunctions. Any Boolean function F(X1,...Xn) of n variables may be realized by a well-formed propositional formula in disjunctive normal form in this way. The code in listing 2 uses the list representation of function tables displayed above and constructs a well-formed propositional formula in disjunctive normal form. Every well-formed propositional formula may be put into disjunctive normal form. The following transformation rules applied to a well-formed propositional formula w as long as any simplifications can be made to yield a disjunctive normal form equivalent to w.

$$(w1 \equiv w2) := ((w1 \supset w2) \wedge (w2 \supset w1))$$
$$\overline{(w1 \supset w2)} := ((\overline{w1}) \vee w2)$$
$$\overline{((\overline{w1}))} := w1$$
$$\overline{(w1 \wedge w2)} := \overline{(w1)} \vee \overline{(w2)}$$
$$\overline{(w1 \vee w2)} := \overline{(w1)} \wedge \overline{(w2)}$$
$$(w1 \wedge (w2 \vee w3)) := ((w1 \wedge w2) \vee (w1 \wedge w3))$$
$$((w1 \vee w2) \wedge w3) := ((w1 \wedge w3) \vee (w2 \wedge w3))$$

These rules may also be converted into a recursive LISP program as in listing 3.

The program PN (push negation) removes EQUIV and IMPLIES, pushes all negations *into* the well-formed propositional formula so that NOTs only appear as part of a literal. PN works by "remembering" how many NOTs it has seen. This is kept track of by a flag which is AND when the number is even and OR if it is odd.

DNF1 then applies the distributive law until the formula is in disjunctive normal form. Thus we compute the disjunctive normal form of a well-formed propositional formula, w, by evaluating:

(DNF (QUOTE w)).

## Conclusion

In this short paper we have given some examples of using LISP data structures in several different ways at once with examples from circuit design. These are not the only examples we could have chosen. A natural extension is the set of programs which deal not only with synthesis and analysis but with the optimization of circuits. That is, construct a program with a specified behavior which is by some measure *best*. For example, we could write code to compute the minimal sum of products representation of a circuit where each product is a *prime* implicant. This is the typical kind of thing studied in courses on combinatorial circuits. ■

# Programming Quickies

## Assembly Language Switching

**Ira Chayut**
**Bell Laboratories**
**Naperville IL 60540**

When programming in assembly language, it is often useful to borrow the tools commonly available to high level language programmers. One such tool is the switch construct, or multi-way jump. A switch steers program execution to one of a number of memory locations, depending on a test value. The switch may be implemented as a series of compares and conditional jumps. An alternate implementation is to create the switch with a subroutine and case tables. A case table can be of variable length; it lists values to be tested for and the associated addresses to which program control may be passed. In addition, a default address is included in the list. If the test value is not equal to any of the values in the list, program execution continues at the default address.

One possible use of the switch is to decode 1-character commands and jump to

*Listing 1: SWITCH, a program to perform multi-way jumps. SWITCH is entered via a jump with register A containing the test value and register pair HL containing the starting address of a case table. The format of the case table is any number of 3 byte case entries followed by a 3 byte default entry. Each case entry consists of a 1 byte case value followed by a 2 byte address. The default entry consists of a byte containing hexadecimal FF followed by a 2 byte address. If the test value contained in register A is equal to a case entry, a jump to the associated address is executed. If no match is found, a jump to the address of the default entry is executed. Since the default value is hexadecimal FF a case value of FF is not allowed.*

*Routine SWITCH does not execute a return itself. If it is entered via a call instruction, the routine indicated in the case table should contain returns to the calling program.*

```
SWITCH:  MOV  B,M        ; get case value
         INX  H          ; point to case address
         CMP  B          ; case and test values equal?
         JZ   SW01       ;  -yes, prepare to jump
         INR  B          ;  -no, case entry equals FF?
         JZ   SW01       ; --yes, prepare to jump
         INX  H          ; --no, point to next case entry
         INX  H
         JMP  SWITCH     ; try next case
SW01:    MOV  B,M        ; get low byte of case address
         INX  H
         MOV  H,M        ; get high byte of case address
         MOV  L,M        ; put low byte in L
         PCHL            ; jump to case address
```

*Listing 2: Example use of SWITCH routine. The value to be tested is put in register A by the call to routine GET. In this case we are checking 1-character commands for addition and subtraction. If the character is neither a subtraction nor an addition symbol, the routine exits at the default jump.*

```
        .
        .
        .
        .
        .
      CALL GET                       ; get a character
      LXI   H,CTBL                   ; point to case table
      JMP   SWITCH                   ; decode command
        .
        .
        .
        .
ADD:  . . .                          ; add routine
        .
        .
        .
SUB:  . . .                          ; subtract routine
        .
        .
        .
ERR:  . . .                          ; invalid command handler
        .
        .
        .
      ; case table follows
CTBL:DB '+'                          ; add command
      DW add
      DB '-'                         ; subtract command
      DW sub
        .
        .
        .
      DB FFH                         ; default, error
      DW err
```

the appropriate servicing routine. The default address might be the start of a section of code to print out an error message.

Listing 1 contains the switch procedure for the 8080 processor. A section of code and a case table illustrating the switch's use appear in listing 2.■

## Turn Your KIM into a Metronome

David Kellerman
1047 Schuyler Dr
Endicott NY 13760

Using the program described in listing 1 (on page 214) and a tape recorder, readers can transform their KIM-1 computers into metronomes. The main part of the program consists of three nested timing loops used to periodically invert the line going to the tape recorder. The resulting square wave pulse is audible as a click through the tape recorder's speaker when the monitor switch is on and the tape recorder is set as if a tape were being recorded. If your recorder has no monitor switch, simply make a recording of the clicks and play it back.

To use the program, set hexadecimal memory locations 0000 and 0001 equal to the appropriate values for the desired click rate (see figure 1 on page 214). Start the program at location 0002, and have fun accompanying your computer!

| Hexadecimal Address | Opcode | Label | Instruction | Comments |
|---|---|---|---|---|
| 0002 | A0 80 | | LDY #80 | Make line to tape recorder |
| 0004 | 8C 43 17 | | STY 1743 | an output line. |
| 0007 | AD 42 17 | | LDA 1742 | Produce a click by inverting |
| 000A | 49 80 | | EOR #80 | the line that goes to the |
| 000C | 8D 42 17 | | STA 1742 | tape recorder. |
| 000F | A5 00 | | LDA 00 | |
| 0011 | 85 E0 | | STA E0 | |
| 0013 | A5 01 | L3 | LDA 01 | |
| 0015 | 85 E1 | | STA E1 | |
| 0017 | A0 FF | L2 | LDY #FF | |
| 0019 | 88 | L1 | DEY | Delay |
| 001A | D0 FD | | BNE L1 | |
| 001C | C6 E1 | | DEC E1 | |
| 001E | D0 F7 | | BNE L2 | |
| 0020 | C6 E0 | | DEC E0 | |
| 0022 | D0 EF | | BNE L3 | |
| 0024 | 4C 07 00 | | JMP 0007 | Repeat |

*Listing 1: Metronome program for the KIM-1 computer. Nested timing loops create audible clicks through a tape recorder hooked up to the computer. The period can be easily altered by the user.*

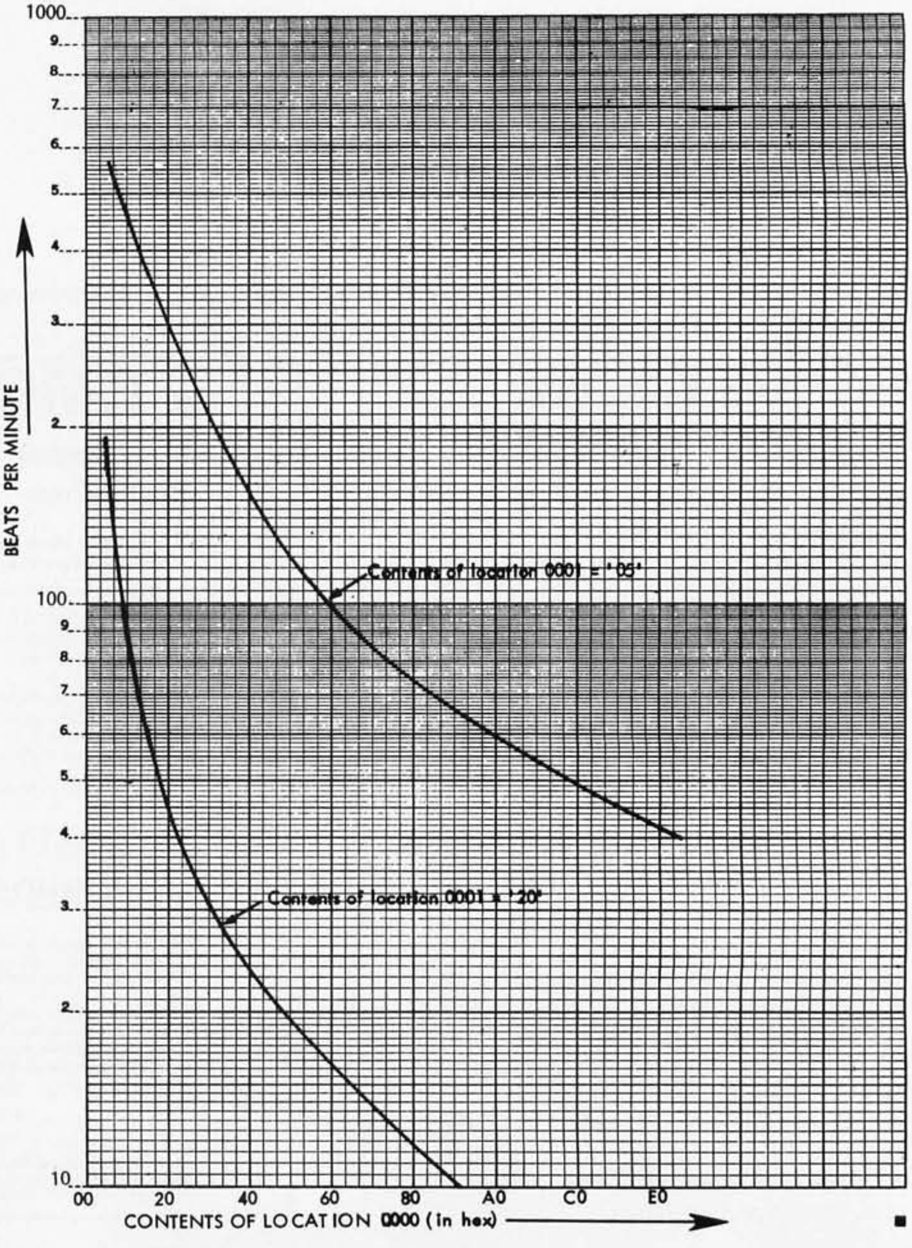



*Figure 1: Calculating the metronome's period. First, find the desired number of beats per minute on the Y axis, then read across to the two curves and enter the corresponding values for the program on the X axis into hexadecimal memory locations 0000 and 0001.*

# Memory Test Program

Frank J Caperello
1806 Kuser Rd Apt 9
Trenton NJ 08690

Did you ever have a program that ran successfully for months, only to have it suddenly bomb? Or are you getting inconsistent results from your data? It could be that your computer is losing its memory. Your problem may be due to memory locations becoming degraded because of a bit failure. With microprocessors having 4 K byte or greater amounts of memory it is almost impossible to check each and every location manually for a bad bit, unless you have a year of free time on your hands.

This wouldn't be a problem if the microprocessor had parity memory. Parity memory is implemented as an extra hardware bit that detects a bit malfunction. Unfortunately, parity memory also has a high cost factor, so it is usually unavailable on microcomputer systems. The memory test program shown here will not replace parity memory, but will assist you and save time in locating bit malfunctions.

This program is 8080 compatible and will check up to 64 K bytes of memory. Although the program was written for an IMSAI 8080 system with front panel, it can easily be modified to work on other 8080 based microcomputer systems. The program can also be modified to be placed in read only memory so a check can be run without having to manually load the program.

Basically, this program clears and sets up the internal registers, inputs the amount of memory you want to test, loads the test memory with a pattern and then checks it. If all goes well, it increments the pattern and repeats the entire process. The test pattern starts out at octal 000 and is incremented to octal 377; when it is incremented again, a pass has been completed. A pass counter is incremented and displayed in the control panel output port light emitting diodes (LEDs). On start up, the

*Text continued on page 217*

*Listing 1: Memory test program.*

```
000  START    XRA      A    257  0s to register A.
001           MOV      E,A  137  0s to pattern register.
002           MOV      D,A  127  0s to pass complete register.
003           CMA           057  377 to output to reflect 0 in light emiting diode (LED).
004           MOV      C,A  117  377 to low order half of maximum address.
005           OUT           323  Output 377 to reflect 0.
006           377           377  In output port LEDs.
007           IN            333  Input from the switches the high half of the maximum
010           377           377  address.
011           MOV      B,A  107  Move it to the high half of maximum add register.
012           INX      BC   003  Increment the register.
013  REDO     MOV      A,E  173  Move the test pattern to register A.
014           LXI      HL   041  Load the first memory location to be tested into the
015  XXA      (FIRST)       133  current address register.
016  XXB                    000
017  LOAD 1   MOV      M,A  167  Go put the test data in.
020           INX      H,L  043  Increment the address.
021           MOV      A,C  171  Get low order half of maximum address.
022           CMP      L    275  Compare it to low order half of current address.
023           JC       Z    312  It compared now go check the high order half of
                                 maximum address.
024           LOAD 2        032
025                         000
026  LOAD 3   MOV      A,E  173  Here there is still more to do.
027           JMP           303  Go get test pattern and jump back and deposit it again.
030           LOAD 1        017
031                         000
032  LOAD 2   MOV      A,B  170  Get the high order half of maximum address.
033           CMP      H    274  Compare it to low order half of current address.
034           JC       NZ   302  Jump if it does not compare.
035           LOAD #        026  This means that there is still more to do.
036                         000
037           MOV      A,E  173  Here we start to check so you get the test pattern.
040           LXI      HL   041  Reload the current address register with the first
041  XXC      (FIRST)       133  memory location to be tested.
042  XXD                    000
043  CHECK 1  CMP      M    276  Check the memory location.
044           JC       NZ   302  If they do not compare jump to the error routine.
045           ERR           107
046                         000
047           INX      HL   043  Here if they do compare, increment the current address
                                 to the next location.
050           MOV      A,C  171  Now get low order half of maximum address.
051           CMP      L    275  Compare it to low order half of current address.
052           JC       Z    312  If they are equal go jump to check the high order
053           CHECK    2    061  half.
054                         000
055  CHECK 3  MOV      A,E  173  Here if still more to check, go get the test data and jump
056           JMP           303  back to recheck it again.
057           CHECK 1       043
060                         000
061  CHECK 2  MOV      A,B  170  Get the high order half of maximum address.
062           CMP      H    274  Compare it to low order of current address.
063           JC       NZ   302  Jump if it does not compare.
064           CHECK 3       055  This means that there is still more to do.
065                         000
066           MOV      A,E  173  Get the test data.
067           INR      A    074  Increment it for the next pattern.
070           MOV      E,A  137  Save the test data.
071           CPI           376  See if the test data is equal to 0s.
072           000           000
073           JC       NZ   302  Jump if it is not — this means that we still have patterns
074           REDO          013  to do before we can complete this pass.
075                         000
076           MOV      A,D  172  Pass complete so get the pass counter.
077           INR      A    074  Increment register.
100           MOV      D,A  127  Put it back to save it.
101           CMA           057  Complement it so it looks correct in the control panel
102           OUT           323  LEDs and output it to the IO port.
103           377           377
104           JMP           303  Go back and redo the test.
105           REDO          013
106                         000
107  ERR      SHLD          042  Here if we have an error store the current address where
110           ERR 3         131  the fault occurred.
111                         000
112           STA           062  Store the correct data as it should have been read from
113           ERR 2         130  memory.
114                         000
115           MOV      A,M  176  Go retrieve the incorrect data.
116           STA           062  Store it so we can see where the error was.
117           ERR 1         127
```

*Listing 1 continued on next page*

| | | | | | |
|---|---|---|---|---|---|
| 120 | | | | 000 | |
| 121 | | MOV | A,D | 172 | Now get the number of completed passes and store this |
| 122 | | STA | | 062 | away for future use. |
| 123 | | ERR 0 | | 126 | |
| 124 | | | | 000 | |
| 125 | | HLT | | 166 | Stop. |
| 126 | ERR 0 | 000 | | 000 | Pass number. |
| 127 | ERR 1 | 000 | | 000 | Bad data. |
| 130 | ERR 2 | 000 | | 000 | Good data. |
| 131 | ERR 3 | 000 | | 000 | Low order half of failed address. |
| 132 | | 000 | | 000 | High order half of failed address. |
| 133 | FIRST | 000 | | 000 | First tested location. |

*Text continued from page 215:*
program receives the number of the 256 locations of memory to be tested via the control panel input port switches. The test will run until the stop button is depressed or until an error is detected.

Let's look at what happens when an error is detected; the machine comes to a halt — the error 0 location contains the number of successfully completed passes.

In the error 1 location is the incorrect data as retrieved from the faulty memory. In the error 2 location is the correct data as it should have been read from memory. In the error 3 location is the low order half of the offending address, while in the error 3+1 location is stored the high order half of the offending address. By comparing the data in error 1 and error 2, we can determine which bit was picked up or dropped — but what if they're the same?

You then have what is known as a "soft error," or an error that is incorrect on the first read out, but correct the second time around. A soft error can be caused by a timing problem, or a refresh problem when using dynamic memory. The program starts checking data from the lowest address to the highest. When an error is detected, the data from error 1, 2 and 3 should be recorded.

Since there is more memory to check, add 1 to the error 3 data and deposit this in locations xxA and xxC. The error 3+1 location should be entered into locations xxB and xxD. Record the next error when it occurs, continuing the same routine until no new errors are detected, or until a pattern of errors is evident.■

# Book Reviews

Practical Microcomputer Programming: The Z-80
*by W J Weller*
*Northern Technology Books*
*Evanston IL*
*481 pages*
*$29.95*

*Practical Microcomputer Programming: The Z-80* is the third volume in a series which also includes works on the 8080 and 6800 microprocessors. My review of the 8080 volume was published in BYTE, January 1978.

The most obvious differences between the Z-80 and the 8080 volumes in this series are the length and the price. The Z-80 version costs $8 more than its predecessor and it is almost 60 percent longer. There are more than 100 pages of additional text, and much more software is included. The Z-80 volume treats several new topics, among which are floating point arithmetic and graphical output.

This book is intended for two audiences: the first is the beginning assembly level programmer (as all of the textbook basics are included and identified so that the more advanced reader can skip them), and the second is the programmer who is familiar with the 8080 and wants to become skilled in the use of the Z-80. With this in mind, the mnemonics used are *not* those used by Zilog, but an 8080 compatible set. The new Z-80 instructions use forms based on the 8080 mnemonics. Unfortunately, the two sets of Z-80 mnemonics are not compatible.

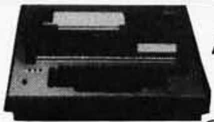
The topics which the book treats are fairly standard: moving data, arithmetic (single and multiple precision, fixed and floating point, binary, and decimal), logical operations, use of the stack pointer, tables and arrays, I/O (input/-output) programming, and the use of interrupts. I/O programming is divided into sections on polled, interrupt-driven, and graphical output. Explanations are clear, and there are many good examples.

The appendices are a nice feature. These contain documentation and listings for a debugging monitor and a conversational assembler. Both of these are written in the 8080 subset of the Z-80 instructions, so that an 8080 programmer can use them (the assembler flags non-8080 instructions). Typing in the code (either object or source) for programs of this size is very tedious, and for this reason paper tapes of the object code for both the monitor and the assembler are free by returning the coupon at the back of the book to the publisher. The assembler can take its source code either from memory or from a tape or disk. A simple line editor is included. You do not have to load the editor, load the source code, punch the source code, load the assembler and load the source code again, as is necessary with separate editors and assemblers. It looks very convenient.

In conclusion, *Practical Microcomputer Programming: The Z-80* has all of the advantages of its 8080 predecessor, while avoiding the major faults. The book is clear and complete (including the index of assembler mnemonics which was missing from the 8080 version), and the appendices are very good. I have been programming the Z-80 for a year and a half, and I wish that I had picked up the knowledge this book offers 18 months ago! ■

**John A Lehman**
**716 Hutchins #2**
**Ann Arbor MI 48103**

SURPLUS ELECTRONICS

ASCII  ASCII

**IBM SELECTRIC® BASED I/O TERMINAL WITH ASCII CONVERSION INSTALLED $645.00**

- Tape Drives • Cable
- Cassette Drives • Wire
- Power Supplies 12V15A, 12V25A, 5V35A Others, • Displays
- Cabinets • XFMRS • Heat Sinks • Printers • Components

Many other items
Write for free catalog
**WORLDWIDE ELECT. INC.**
130 NORTHEASTERN BLVD.
NASHUA, N.H. 03060
Phone orders accepted using VISA or MC. Toll Free 1-800-258-1036
In N.H. 603-889-7661

Circle 391 on inquiry card.

# TRS - 80

**SPECIAL PROMOTION SALE**

**SAVE** 10%, 15% or more on **ALL** Computers, Peripherals, Software, and **ALL** other fine Radio Shack® products.

**NO TAXES** on out-of-state shipments.

**FREE** Surface delivery in U.S.

**WARRANTIES** will be honored by your local Radio Shack® store.

Offered exclusively by
**Radio Shack® Authorized Sales Center**
**1117 Conway**
**Mission, Texas 78572**
**(512) 585-2765**

VISA

Circle 314 on inquiry card.

# SUPPLIES

- FLOPPY DISKS, MINI OR STANDARD MEMOREX OR 3M
- 3M DATA CARTRIDGES DC300A, DC100A
- 3M DIGITAL CASSETTES
- 3M OR MEMOREX AUDIO CASSETTES, C-60
- 3M DISK CARTRIDGES

WE OFFER:
- COMPETITIVE PRICING
- IMMEDIATE DELIVERIES (Any Quantity)
- UNCONDITIONAL GUARANTEE

**BETA BUSINESS SYSTEMS**
8369 VICKERS ST., #G
SAN DIEGO, CA 92111
(714) 565-4505

Circle 34 on inquiry card.

# FLOPPY DISK REPAIR

- *PerSci and Shugart*
- *Quick turnaround*
- *Factory trained on PerSci*

COMPUTER SERVICE CENTER
7501 Sunset Blvd
Hollywood CA 90046
213-851-2226

Circle 68 on inquiry card.

TRS-80 16K MEMORY EXPANSION KIT
INCLUDES 8 TESTED & GUARANTEED MSK 4116 3 16K RAMS. PROGRAMMING PLUGS & EASY-TO-FOLLOW INSTRUCTIONS.
$72.00 PER KIT

6800 64K BYTE RAM SET AND CONTROLLER CHIP SET
*MAKE 64K BYTES OF MEMORY FOR YOUR 6800 THE CHIP SETS INCLUDE:*
$295.00 PER COMPLETE SET.
32 MSK 4116-3 16K RAMS.
1 MC3480L MEMORY CONTROLLER.
1 MC3242AP MEMORY ADDRESS MULTIPLEXER/COUNTER.
DATA & APPLICATION SHEETS.
PARTS TESTED & GUARANTEED.

16K DYNAMIC RAMS
MSK 4116-3 200NSEC ACCESS TIME/375NSEC CYCLE TIME. TESTED & BURNED-IN.
$8.50 EACH/MIN. QTY. 8

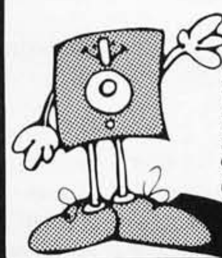
4K STATIC RAMS
EQUIV. TO TMS40L44-30 300NSEC ACCESS TIME/CYCLE TIME FOR 4MHZ Z-80 OPERATION.
$7.50 EACH/MIN. QTY. 8
TESTED & GUARANTEED

*QUANTITY DISCOUNTS AVAILABLE*
ALL ORDERS POSTPAID. U.S. FUNDS. CHECK OR MONEY ORDER. VISA, BA, MASTERCHARGE — SEND ACCOUNT NO., EXPIRATION DATE, INTERBANK NO., & SIGNED ORDER. PHONE ORDERS: 714/633-4460
MEASUREMENT SYSTEMS & CONTROLS, INC.
MEMORY DEVICES DIVISION, DEPT. B3
867 NORTH MAIN ST., ORANGE, CA 92668

Circle 216 on inquiry card.

# Memorex Floppy Discs

Lowest prices. **WE WILL NOT BE UNDERSOLD!!** Buy any quantity 1-1000. Visa, Mastercharge accepted. Call free (800)235-4137 for prices and information. All orders sent postage paid.



**PACIFIC EXCHANGES**
100 Foothill Blvd.
San Luis Obispo, CA 93401. (In Cal. call (805) 543-1037.)

Circle 294 on inquiry card.

**MAXIMUM VALUE FOR YOUR DOLLAR**

**NORTH STAR COMPUTER PRODUCTS**

HORIZON1 16K KIT........$1275.00
16K RAM BOARD KIT......$ 250.00
32K RAM BOARD KIT......$ 475.00

**VERBATIM DISCS FOR NORTH STAR**
BOX OF 10.........$29. POST PAID

**COMPLETE SYSTEMS AVAILABLE**
CUSTOM SOFTWARE FOR NORTH STAR SYSTEMS

**CASIO CALCULATORS AT DISCOUNT PRICES**

MANY OTHER SUPER VALUES
WRITE OR CALL:

**A.E.I.**
3851 HACKETT AVE.
LONG BEACH, CALIF. 90808
(213) 421-4815 (213) 429-0535

Circle 21 on inquiry card.

**save more than 20%!**
**NORTH STAR——INTERTUBE**
**THINKER TOYS——HUH**
**the smartest computers at the smartest price**

**DOUBLE DENSITY**
HORIZON-1-16K kit, list $1599 ...**$1279**
Assembled & tested, list $1899 ...**$1519**
HORIZON-2-32K kit, list $2249 ...**$1799**
Assembled & tested, list $2549 ...**$2039**
HORIZON-2-32K **QUAD DENSITY**
Assembled & tested, list $2999 ...**$2399**
PASCAL for NORTH STAR on Disk .......**$49**
Powerful NORTH STAR BASIC.........**FREE**
Measurement Systems asm memory 64K. **$640**
Thinker Toys Discus/2 D asm $1149.....**$949**
TRS80 to S-100 HUH kit list $295 .....**$260**
Assembled & tested, list $375 .....**$350**
**INTERTUBE II Terminal, list $995 .....$780**
MARYELLEN Word Processing...
$34 + $1.50 shipping
VERBATIM Disks...
10 for $29.50 + $1.50 shipping
Which Computers are best? BROCHURE...**FREE**

**AMERICAN SQUARE COMPUTERS**
**KIVETT DR, JAMESTOWN NC 27282**
**(919) 883-1105**

Circle 5 on inquiry card.

# RECYCLE(D) COMPUTERS

BUY ☆ SELL ☆ SWAP

Hardware & Software

NEW PRODUCT ANNOUNCEMENTS
22 pages

Mailed 1st Class every 3 Weeks

1yr. (18 issues) ☆ $3.75

ON_LINE

Dave Beetle, Publisher ........... Established 1973

24695 Santa Cruz Hwy. • Los Gatos, CA 95030

THE BEST WAY TO DETERMINE IF ON_LINE CAN BE OF VALUE TO YOU IS TO TRY A .....

FREE SAMPLE ISSUE

Circle 288 on inquiry card.

# An Overview of Long Division

Geoffrey Gass
5240 SW Dosch Rd
Portland OR 97201

On the very simplest level, a division problem starts with two numbers, a *dividend*, which we want to divide by a *divisor*, to obtain a third number, a *quotient*. In terms of grade school long division:

```
        Quotient      + Remainder
Divisor )Dividend
```

The quotient (integer portion) is simply the number of times the divisor can be subtracted from the dividend and still leave a positive remainder.

The simplest computer program for this calculation goes as follows:

- Put the dividend into register N.
- Put the divisor into register D.
- Clear a quotient register Q.
- Assign a remainder register R.
- Subtract D from N and put the result in R.
- Test R.
- If R is positive, increment Q, transfer R into N, and go back to the *subtract* step.
- If R is negative, exit. Q is now the (integer) quotient and N contains the remainder.

There is nothing basically wrong with this procedure, but it's not very useful. If N is 1,000,000 and D is 2, it will take 500,000 operations of the program to get Q. If D is 797,236, the program will quickly tell us the answer is 1, with a remainder.

Let us check off the chief deficiencies. First, if the two numbers are very different, the program will give us an accurate answer, but will take a long time doing it. Second, if the two numbers are very close in value, the program will be very quick, but not very precise. Third, if D is larger than N, zero is the only answer. Fourth, if D happens to be zero, the program will loop forever trying to get Q up to infinity.

What we'd prefer is a quicker program that gives us an answer correct to at least as many places as the significant digits of the numbers we put in, regardless of the magnitude of the numbers. But won't that take a more complicated program and won't a more complicated program take longer to execute? A program 2,000,000 instructions long could be quicker to execute than one which loops through six instructions 500,000 times. And it certainly won't take two million instructions to make a quite thorough, precise, accurate and quick division program.

To get speed and precision, start out just as a previous generation was taught in grade school, by juggling the decimal points around (or binary points if we are working in binary). To put it another way, multiply the divisor and dividend some number of times by the base of the number system (10 or 2, for example) until the

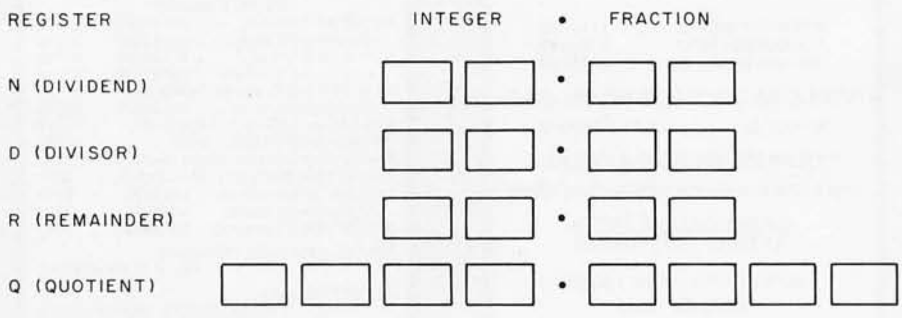



*Figure 1: Four registers for division, each with two words for integers and two for fractions, except register Q which is double size. The registers are usually strung out serially in adjacent memory locations, but it is convenient to think of them in block form as shown.*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| N | | | 0000 | nn00 | . | 0000 | 0000 | | |
| D | | | 0000 | 000d | . | 0000 | 0000 | | |
| R | | | 0000 | 0000 | . | 0000 | 0000 | | |
| Q | 0000 | 0000 | 0000 | 0000 | . | 0000 | 0000 | 0000 | 0000 |

*Table 1: Starting arrangement of the registers for division. The dividend and divisor have been loaded; all other registers are cleared.*

dividend is only slightly larger than the divisor. Note how many places it is necessary to shift the numbers so they are nearly equal. (In old-fashioned long division, the divisor is shifted until it is an integer, and the dividend is shifted the same number of times. The rest of the necessary shifting is done by relocation of the quotient with respect to a fixed location for the decimal point.)

To start, set up an array of registers large enough to hold the largest numbers we want to deal with. The quotient register is twice as large as the others, since dividing a very small fraction by a very large number produces a yet smaller fraction, and dividing a very large number by a small fraction gives an even larger quotient. Then arbitrarily define some point in each register as the decimal or binary point. A convenient place is between two memory words, as shown in figure 1. Although a more common technique is to use only three registers (no R register), using four is a little easier, and you'll never notice the slightly increased time required for putting R into N after every successful subtraction. However, extra time is only needed for BCD (binary coded decimal) division. In binary arithmetic, the extra time for an addition after every *unsuccessful* subtraction approximately balances the time wasted in transfers.

The first operation is to load in the numbers, being careful to locate them in the proper position with respect to the decimal point. If the dividend N is nn00, it will go into the word just to the left of the point in N. If the divisor is 000d, it will go in the corresponding word of register D. All other locations must be cleared to 0000, if not already done. Table 1 shows our starting arrangement. Because the program is general purpose, and must be able to operate with any kind of numbers that can be fitted into its registers, it can't "know" how big N and D are. Its first job is to find out their magnitudes so it can set them to be nearly equal.

The easiest way to do this is to start by shifting register D to the left and insert zeros at the least significant digit position of the fraction part of the register until something pops up at the most significant digit position at the left of the integer part of the register. In this operation we must set a limit to the number of shifts allowed, so when we have done 16 shifts and still get nothing at the top of the register, we can stop. Division by zero is not allowed, of course, and the computer has better things to do than spend hours shifting empty registers. Then do the same thing with register N, shifting it left until its most significant digit shows at the top of the register. We can use the same counter used for D to keep track of how many shifts it takes, starting with the count left over from counting D's shifts and counting in the opposite direction. Our final count will reflect the difference in magnitude between the two numbers. That number is saved for later. Again, with N, it is necessary to set a limit to the count or we'll be shifting forever if N happens to be zero. The limit needn't be exact (it can't be, because we don't know what number we started with in the counter), but that's not critical. All that's needed is something that will get us out if the count starts looking like infinity. A limit of $-20$ or $+20$, depending on which way the counting starts, is adequate. In the example of table 1, the saved number is 3 (the difference between the seven shifts it took to get D to the top of the register and the four shifts required for N).

Before starting subtraction, counting and shifting, a certain number of operations must be set. Since we started with possible 16-position numbers, 16 operations should give 16 position answers, which is what we were looking for. We will be moving quotient digits into the Q register at a point 15 places to the right of the binary/decimal point. If the answer is 1, 16 shifts will put that first and only digit of the answer just to the left of the binary/decimal point in Q.

Now, with a starting count of 16, and the D and N numbers in position, subtract D from N and put the result in R. Is R negative? (If binary coded decimal notation is used D could be larger than N, and R could therefore be negative. If binary notation is used, N must equal D, so R could not in the specific example be negative; but we test for it anyway.) If R is negative, go immediately to the next operation. If R is positive, transfer R to N and increment Q. If working in binary arithmetic, go to the next operation at this point, since another subtraction cannot be done. If working in binary coded decimal, however, N could be 9 and D could be 1, and there are eight operations yet to go. So for binary coded decimal, loop back and keep on subtracting and swapping R back into N until R is finally negative, then stop. Don't transfer R or increment Q, just get on to the next operation.

At this point, the most significant digit of the quotient is in the least significant digit position of register Q. Now shift D one position to the right and shift Q one position left, marking the end of one operation in our operations counter. Keep repeating the above process until all 16 shifts have been done. At this point, the first Q digit is one position to the left of the binary or decimal point in Q. Now, go back and look at the magnitude difference count obtained at the start of the program. If it is positive, shift Q to the left that many times; if it is negative, shift Q to the right that many times. (We could have checked the magnitude difference count when the operations counter was set: if the magnitude difference was negative, set that many fewer operations for the program. We would *not* have added any positive number, however; that would set up a divide by zero for the 17th operation.) Register Q now has the correct quotient.

We neglected the small problem of loading the digits into the registers in their proper positions, and didn't get into fine detail on how a subtract or shift operation might be performed in a multiword register; however, the general outline of the algorithm can now be imagined, and that's half the battle. And there are some details of it that can help us along to the next step.

When the numbers were shifted up to the tops of their registers in the earlier example, we were actually going through the process of converting fixed point to floating point numbers, by normalizing the digits, with a saved exponent indicating how far they'd been shifted. In that specific example, we saved only the difference in exponents, but this gave us the information needed to create a conventional notation number from our floating point answer in Q.

Our next step is to establish a full floating point format in order to avoid the magnitude limitations forced on us by fixed point data. Because most processors are equipped with binary coded decimal arithmetic aids, there is no need to bother with binary coded decimal to binary conversions (and vice versa) when handling numbers input via the keyboard. Also, battling with the attendant conversion problems can be avoided (ie: decimal fractions that can only be approximated by binary fractions and rounding operations which don't come out the same in binary coded decimal and binary).

In floating point format, every number is stored as a string of digits, with the most significant nonzero digit at the top of the register and the decimal point *location* saved in a separate register. The programmer can arbitrarily say that the imaginary decimal point is anywhere in the normalized string of digits as long as the program is internally consistent. For ease of output in standard scientific notation, however, it's best to say that the 0 position of the decimal point is immediately following the most significant digit in the register. That is, the number stored is 1 or greater and less than 10, and is to be multiplied by 10 to the power indicated to obtain conventional notation.

The number 6045.35 is stored as:

EXP NUMBER
03 604535

with the number in EXP indicating how many places further to the right of the first digit the decimal place must be moved for conventional notation. If EXP is 00, the number is 6.04535; if EXP is FD (—3 in hexadecimal form), the number is .00604535. In addition to the number and the base exponent, we also need something to indicate the sign of the number.

In binary operations, the most significant bit of a number can be considered the *sign* bit, providing a single byte with the range of values +127 to −128 decimal. Arithmetic performed under this convention gives consistent answers (except under overflow conditions for which most processors have detection circuits and warning flags). For binary coded decimal, the topmost *digit* position is the sign *digit*: 0 for a positive number, and 9 for a negative number. Negative numbers are generally handled in *tens complement* form, obtained by subtracting the absolute value from 999999999....9 and then adding 1 to the least significant digit (this is the way many early adding machines handled subtraction).

Without going into the detail of how it got that way, simply assume that all data in our division problem will be available to us in tens complement form, in the format shown in figure 2. The exponent could be in binary coded decimal form (maximum values + and − 79, with the most significant bit used as a sign bit), but it's easier to keep it in binary form, allowing a value range of +127 to −128, limited by the program to plus and minus 99. The format gives nine significant digits, of which we may elect to hold out two or three as guard digits, and display only six or seven, rounded off according to the value of the guard digits.

There is one more complication in our division routine: *signs*. The operation we want to perform here is repeated subtraction of absolute values, not just the simple signed subtraction for which the tens complement form can give correct answers. When dividing +956 by −3, we do not want the remainder to become larger and larger! So first of all, look at the sign digits of the two numbers (if a number is negative, the 9 at the most significant digit position will set the N bit of a condition code register, just as for binary operations) and determine the proper sign for the quotient. Store this flag away for the moment.

Next, if the dividend is negative, use a tens complement routine to get its absolute value, and put it back in register N. We might also test it for 0 at this point, and do an early exit if the answer is going to be 0. This would be appropriate only if we had already checked D, since D might also be 0, and 0/0 would be an indeterminate value, not 0. So don't bother with the zero check at this point if register N is being processed first.

What we do with register D depends on the processor being used. Some processors have decimal subtract operations, or a binary coded decimal adjust instruction which is effective after a subtraction. In the Motorola 6800, the DAA instruction works properly only after an ADD operation with register A (ADD A, ADC A or ABA). For the 6800, then, the subtraction function requires

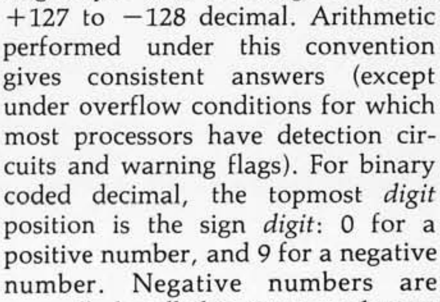



*Figure 2: Register arrangement for floating point binary coded decimal division. Note that it is no longer necessary to provide a double size register for Q. The imaginary decimal point is located immediately following the most significant digit.*

register D to be in tens complement negative form, so our subtraction can be performed with an addition instruction. For other processors, D must be in absolute (positive) form if a subtract instruction is to be used, or in complemented form for an add instruction, depending on what is available in the machine.

So we do or don't run the data in register D through a tens complement operation depending on its present form and the form required by our division routine. While checking the sign, we can also note if D is 0; if it is, we set an error flag and exit. If D is not 0, check here to see if N is 0, and exit early if it is (assuming register Q is already cleared), thus saving a little processing time.

Next, look at the exponent data to discover what the final exponent will be. Subtract the D exponent from the N exponent, but before storing it away check for overflow (a carry into the sign bit, effectively reversing the sign from what it should be), or, if we have set limits of + and −99, check for a number exceeding these limits. If the magnitude of the answer is going to be out of limits, we may choose to reject the problem, set a warning flag, or simply set Q to 0 or 999999E99 to indicate that the result is beyond the capacity of the machine if the program is simply a calculator program without programmability or other exotic features. For a scientific program, this sort of thing could lead to serious and probably undetectable errors, and would never do. For an interpreter program, the exponent overflow should spring out to an error message and halt the program. If the exponent is within limits, store it as the tentative exponent for Q, subject to later adjustment.

Now, we're finally ready to divide. We can skip the procedure done earlier in running data up to the tops of the registers. First, set up a count of nine (the number of digits desired). Subtract D from N, with binary coded decimal adjustment as required, and store the difference in R. If R is positive (checking byte 1 in R), increment the least significant bit in register Q (no need for binary coded decimal adjustment here — the digit will never exceed nine), transfer R to N and repeat until R is finally negative. Leave R alone this time and do not increment Q. Shift Q one digit (four bits) left, starting at the least significant byte of the register and shifting it one bit left, repeating the process four times. Then shift D one digit (four bits) right, starting at the most significant byte of the register and going through it four times. One more operation must be remembered when working with D in tens complement form and doing additions: the sign digit of D must be extended back to the top of the register. Do this by adding 90 to the most significant byte after we have completed the shifting above. When we get down to the last operation, register D should be all 9s except for the least significant digit.

Before going back to the subtract operation, step the operations counter by one, and exit if the counter indicates completion. When the subtracting is done, check the most significant digit of register Q. If it is 0, the result of the first subtraction was no good and the initially assigned exponent for Q was too large. Under these circumstances we shift Q one more digit to the left and reduce the exponent that was calculated earlier by 1.

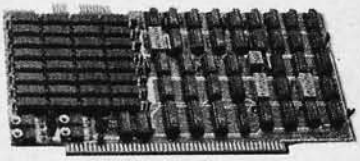
Now, everything is taken care of except the sign. If we have a simple calculator program, we can just look at the sign flag stored away and either do or don't output a minus sign, followed by the register Q data in absolute form. However, for most applications, Q will have to be stored away for future use in machine usable form (as previously discussed in figure 2), just as we got the N and D data to start with.

So look at the sign flag. If it says Q is negative, send Q through the tens complement routine, then store the result wherever it belongs. If Q is to be positive, store it as is, with 0 for a sign digit. In either case, "park" the exponent data next door, so it can be retrieved along with Q's digits whenever needed.

Well, we did it. A whole long division program in binary coded decimal, with a constant precision answer. Of course, we haven't actually formatted the digits for output, or converted our binary exponent to signed ASCII, or decided whether to output the number in conventional or scientific notation (there really isn't room on the average printer for 99 zeros). We also haven't figured out how to use the exponent to locate the decimal point in the printout of conventional notation data. But these things are incidental. Once past the conceptual problem of the "engine" in this dividing machine, the design of the transmission, differential, seat cushions and bumpers should be no barrier to rapid progress in any direction that suits the user. ■

# What's New?

### 16 K Byte Dynamic Programmable Memory Board



Called SupeRam, this S-100 bus compatible 64 K byte dynamic programmable memory board is available from Alpha Micro, 17881 Skypark N, Irvine CA 92714. It is completely compatible with the 16 bit Alpha AM-100 processor. SupeRam is a high density programmable memory board capable of storing up to 64 K bytes of data on a single board. Completely S-100 bus compatible, it utilizes 16 K byte dynamic programmable memories to achieve maximum bit density, minimum power dissipation, and optimum cost and performance ratio.

Refresh requirements are satisfied on the SupeRam board without support from the processor and are therefore transparent to the user. Exact and reliable internal timings are generated on board using digital delay line techniques. Memory is addressable as independent 16 K byte blocks, providing maximum capability with existing Alpha Micro Systems.

Circle 559 on inquiry card.

### New Software for Computalker Speech Synthesizer

Computalker Consultants, designers and developers of the Computalker CT-1 Speech Synthesizer (a device that enables a computer to speak) has announced the availability of the new Software Package II. Designed to expand the range of applications of the Computalker CT-1, Software Package II contains: CTEDIT, a new parameter editor; CSEDIT, an editor for the CSR1 input; CTEST, a CT-1 hardware diagnostic; PLAYDATA, to hear the data files; MEMVOICE, a vocal memory dumper; KEYPLAY, a subroutine to play letters and digits; and PIANO, a simple musical keyboard.

Software Package II is written in 8080 assembly language and includes the source code. It is priced at $45 and is available on CP/M format 8 inch floppy disk; North Star and Micropolis disks; Tarbell, CUTS, MITS ACR cassette formats; and paper tape. For further information, contact Computalker Consultants, 1730 21st St, Suite A, Santa Monica CA 90404.

Circle 560 on inquiry card.

### Z-80 Assembler Package

ZASSEMBL is a package of software designed for development of Z-80 assembly language programs. ZASSEMBL is written in North Star BASIC with critical routines implemented in Z-80 machine code. Zilog suggested mnemonics are used exclusively for all 696 standard Z-80 instructions. The package consists of three BASIC programs:

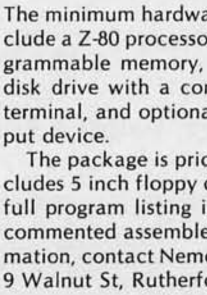
| | |
|---|---|
| Editor | enters and edits source text |
| Assembler | one pass file oriented assembler with back-patching of forward references. Inserts ASCII hexadecimal representation of the op code into the source code |
| Loader | generates binary executable code and loads it into either programmable memory or a file |

The minimum hardware requirements include a Z-80 processor, 32 K bytes of programmable memory, one 5 inch floppy disk drive with a controller, interactive terminal, and optional printer as an output device.

The package is priced at $35 which includes 5 inch floppy disk, a manual with full program listing in BASIC, and Z-80 commented assembler. For further information, contact Nemco Data Processing, 9 Walnut St, Rutherford NJ 07070.

Circle 561 on inquiry card.

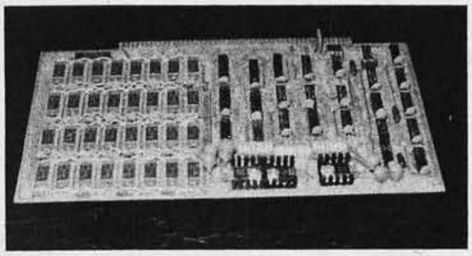
### Programming the 6502




*Programming the 6502* by Rodney Zaks is an educational text designed to teach programming from the ground up. It will show the reader both the advantages and disadvantages of using the 6502. The knowledge of programming gained with this book may be applied to other microprocessors. Structured from simple to complex, this 310 page text may be used by the person who has never programmed as well as by programmers wishing to familiarize themselves with the 6502. The book is priced at $10.95 and is available from Sybex, 2020 Milvia St, Berkeley CA 94704.

Circle 562 on inquiry card.

### 64 K Byte Programmable Memory Card



This 64 K byte programmable memory card will reduce system card count by using only one S-100 card slot. It uses the same power as the standard 16 K byte programmable memory card, thus lowering power requirements. Buffered signal lines mean less loading on buses. Memory is expandable in 16 K byte increments up to 64 K bytes and memory may be disabled in 256 byte blocks for read only memory programs. The fast cycle time of the new 16 by 1 dynamic programmable memory means no wait states are needed for reads, writes or refreshing. The memory card handles refresh. For more information, contact Microcosm Inc, 534 W 9460 S, Sandy UT 84070.

Circle 563 on inquiry card.

**Attention Readers and Vendors...**

**Where Do New Products Items Come From?**

*The information printed in the new products pages of BYTE is obtained from "new product" or "press release" copy sent by the promoters of new products. If in our judgment the information might be of interest to the personal computing experimenters and homebrewers who read BYTE, we print it in some form. We openly solicit releases and photos from manufacturers and suppliers to this marketplace. The information is printed more or less as a first in first out queue, subject to occasional priority modifications. While we would not knowingly print untrue or inaccurate data, or data from unreliable companies, our capacity to evaluate the products and companies appearing in the "What's New?" feature is necessarily limited. We therefore cannot be responsible for product quality or company performance.*

The

TO ORDER CALL TOLL FREE 800-223-7318

# COMPUTER FACTORY

## NEW PET BUSINESS SYSTEM


### PET 2001 - 16/32K

The PET is now a truly sophisticated Business System with the announcement of the Floppy Disk and Printer. This is an ideal business system for most professional and specialized fields: medicine, law, dental, research, engineering, toolmaking, printing, education, energy conservation, etc. . . .
The PET Business System as a management tool, delivers information to all levels of Business, previously attainable only with equipment many times more expensive. The PET Business System is one of the most cost efficient business tools today. Here are just a few of the cost-saving uses in the corporation, professional office or small business: stock control, purchasing, forecasting, manufacturing costing, customer records, mailing lists, etc. . . .

### Features Include:

- 16 or 32K bytes RAM user memory
- 14K ROM operating system including a machine language monitor
- Full-sized Business Keyboard
- Upper/Lower case and 64 graphics characters
- 9-inch CRT
- 8K ROM expansion sockets
- File management in operating system

**16K - $995,**
**32K - $1295**


Model 2040
**$1295**

### DUAL DRIVE FLOPPY DISK 2040

The Dual Drive Floppy is the latest in Disk technology with extremely large storage capability and excellent file management. As the Commodore disk is an "Intelligent" peripheral, it uses none of the RAM (user) memory of the PET. The Floppy Disk operating system used with the PET computer enables a program to read or write data in the background while simultaneously transferring data over the IEEE to the PET. The Floppy Disk is a reliable low cost unit and is convenient for high speed data transfer.
Due to the latest technological advances incorporated in this disk, a total of 360K bytes are available in the two standard 5¼-inch disks, without the problems of double tracking or double density. This is achieved by the use of two microprocessors and fifteen memory IC's built into the disk unit.

### Features Include:

- 360K bytes storage
- 6504 microprocessor-controlled
- 8K operating system in ROM
- 4K encoder and decoder in ROM
- 4K RAM
- Uses single or double sided floppies

**Model 2040A Single Disk Unit - $895**

***(Next day delivery available.)***


MODEL 2022
**$995**

### TRACTOR FEED PRINTER 2022

The Tractor Feed Printer is a high specification printer that can print onto paper (multiple copies) all the PET characters - letters (upper and lower case), numbers and graphics available in the PET. The tractor feed capability has the advantage of accepting mailing labels, using standard preprinted forms (customized), check printing for salaries, payables, etc.
The PET is programmable, allowing the printer to format print for: width, decimal position, leading and trailing zero's, left margin justified, lines per page, etc. It accepts 8½-inch paper giving up to four copies.

### Features Include:

- 150 cps
- 6504 microprocessor-controlled
- ½K RAM buffer
- Bottom and rear tractor feed
- 4K operating system in ROM

**Model 2023 (Friction Feed) - $849**

CABLE FROM PET TO DISK OR PRINTER - $39
CABLE FROM PRINTER TO DISK - $49

***FULL SYSTEM NOW IN STOCK FOR IMMEDIATE DELIVERY!***

## SUPER WORD PROCESSING SYSTEM

**$2395 complete with software**

System Includes:
- 16K PET with Full Sized Keyboard
- Tape Drive Unit
- Anderson Jacobson 841 Selectric Printer
- Interface
- Super Word Processing Software


The Super Word Processing System is written in 6502 Assembly Language! It allows anyone to use the PET computer for such tasks as typing letters, reports, and manuscripts, for producing mailing lists, and for filling out forms. The software is written to support any inexpensive printer and even high performance printers with incremental and proportional letter spacing. The Super Word Processor easily creates, edits, rearranges, and formats text. It can merge multiple texts. In addition to search commands, the system has automatic text wrap-around, which eliminates the need for carriage returns. The system uses a holding buffer to rearrange blocks anywhere on the page. Up and down screen scrolling makes editing a breeze! Commands include end-to-end cursor line SCAN, INDENT, TAB, soft HYPHEN) for splitting syllables at the end of a line,( and four-direction cursor control. Output formatting includes dynamic print control, indentation, right justification, line width and line-to-line spacing and proportional letterspacing.
Also included are programming capabilities for performing such useful tasks as direct-mail form-letter typing, multiple column printing, and automatic multiple forms entry. You may obtain this system in a cassette or disk drive version.

### BUSINESS SOFTWARE FOR PET BUSINESS SYSTEM

- Super Word Processing Package (disk & tape versions)
- Real Estate
- Statistics
- Banking & Finance
- Mail List Management
- Data Base System
- Small Business Package (A/R, A/P, G/L)
- General Ledger
- Super Random Access
- Cash Receipts & Disbursements
- Inventory Control (for manufacturers)

### PERIPHERALS FOR PET

| | |
|---|---|
| • 24K Memory Expansion | $499 |
| • 16K Memory Expansion | 399 |
| • PET to RS232 Serial | 169 |
| • 2 Way Serial/Communication | 229 |
| • Modem Board for PET | 375 |
| • Analog to Digital Board for 16 Devices | 275 |
| • Second Cassette Drive | 95 |
| • Parallel Printer Interface | 199 |

### NEW! from Eventide

### AUDIO SPECTRUM ANALYZER

- Mounts inside the PET
- Third-Octave audio spective analysis
- Complete with software and documentation
- Replaces equipment costing thousands of dollars

**$595**

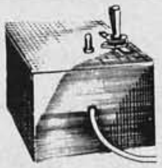
master charge · VISA

**Min Credit Card Order $75**

N.Y. residents add 8% sales tax • Same day shipment on prepaid and credit card orders • Add $5 shipping for computers, $3 for boards, $.25 each cassette tape.

**Open Mon-Fri 10-6 Sat 10-4**

### PET MUSIC BOX



Add music and sound effects to your programs. Compose, play, and hear music on your PET. Completely self-contained (no wiring). Free 3 programs including; Star Wars theme, sound effects, etc. $39.

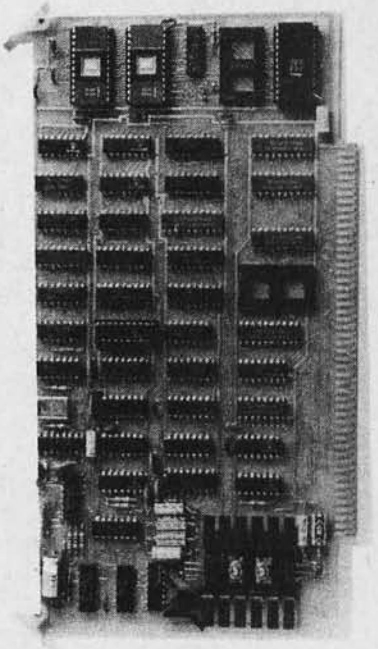
### MARK SENSE CARD READER $750

- Automatic turn-on and card feed
- Ideal for marking test scores
- Accepts any length card
- Perfect for schools & business

**TO ORDER CALL TOLL FREE 800-223-7318**

**The COMPUTER FACTORY** 485 Lexington Avenue 750 Third Avenue (46th St.) New York, N.Y. 10017
(212) 687-5001 (212) PET-2001 Foreign order desk - Telex 640055

### Video Board Features High Density and Reverse Video



A higher density version of the Flashwriter Video Board, featuring optionally controlled reverse video, has been announced by Vector Graphic Inc, 31364 Via Colinas, Westlake Village CA 91361. Displaying 80 characters by 24 lines, the Flashwriter II uses an 8 by 10 dot matrix to produce crisp, sharp resolution for 1920 character positions in a 2048 byte memory block. In addition to normal video, reverse video is optionally controlled by the higher order bit of the character code. As many as 256 characters can be generated by 2708/2716 erasable read only memories which may be user-programmed for special symbols or graphic displays.

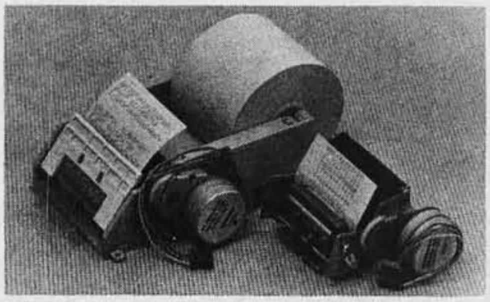
The Flashwriter II allows rapid updating of the screen via memory mapped I/O (input/output). Special circuitry prevents flashes on the screen when updating memory, and a keyboard port with latched data provides easy interface to Vector Graphic's Mindless Terminal or other parallel keyboards.

The Flashwriter II is priced at $320 assembled.

Circle 526 on inquiry card.

### Low Cost Alphanumeric Printers

These two compact, light weight, 5 by 8 dot matrix printers are being offered by

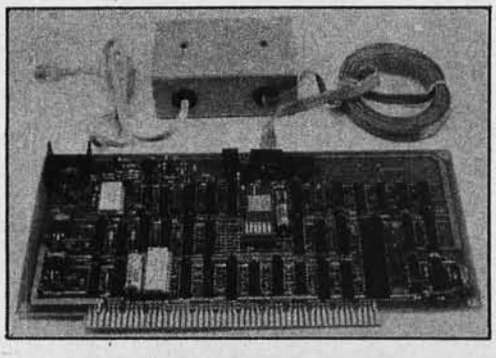


American Micro Products Inc, 6550 Tarnef, Houston TX 77074. The 12 column PL12 at $59.95 and the 20 column PL20 at $99.95 provide quiet economical hard copy output. A general specification manual, art work for a printed circuit board (available only with the PL20), parts lists, flow chart, and schematics describing the 8 bit parallel interface (Centronics type) are included with each printer. In addition, the microprocessor control device and the printed circuit board (PL20 only) are available as options. These elements of the interface are priced at $99.95 and $29.95, respectively.

Circle 527 on inquiry card.

### FCC Approved Data Modem and Communications Adapter

This S-100 bus compatible data modem and communications adapter, designated the MM-103, has been approved by the Federal Communications Commission (FCC) for direct con-



nection to the public switched telephone network without the use of a DAA (CBS or CBT). Under software control, it can originate and answer calls automatically. It can also dial the telephone automatically.

In addition to normal digital communications capabilities, the MM-103 provides auxiliary inputs and outputs that will interface with computer system power-up control (on telephone ring or external input); voice recorder announcement equipment; and alarm recognition and automatic dial equipment.

The modem is available fully assembled for $319.95 which includes an unconditional ten day return privilege and a one year limited warranty. For further information contact Potomac Micro-Magic Inc, POB 11149, Alexandria VA 22312.

Circle 528 on inquiry card.

### New Family of RS-232 Switching Units

A new family of low cost miniature switching units has been introduced by Giltronix Inc, 3156 Avalon, Palo Alto CA 94306. The family, called RS232-X, switches serial RS-232 peripherals between several driving sources. Model RS232-X3 allows three driving sources. By turning the three position switch mounted on the RS232-X3, the user can select the driving device that will exchange data with the peripheral unit. A unique arrangement allows the cascading of two or more RS232-X switches, thereby expanding the selection from three devices to five or more. Model RS232-XF is similar to the RS232-X3, but switches additional signals. Both come with 25 pin female connectors. The price of the RS232-X3 is $64.95 assembled, and $47.95 in kit form. The RS232-XF is $78.95 assembled, and $59.95 in kit form.

Circle 529 on inquiry card.

### TRS-80 Speech Synthesizer from Computalker

Computalker Consultants, developer of the Computalker CT-1 Speech Synthesizer, has announced the availability of the Model CT-1T, a speech synthesizer adapted specifically for the Radio Shack TRS-80 microcomputer equipped with Level II BASIC and a minimum of 16 K bytes of programmable memory (32 K bytes recommended). The Model CT-1T Speech Synthesizer is a completely self-contained unit with its own AC power supply. The interface circuit board contains an on board 2 W audio amplifier, an S-100 connector for the CT-1 speech synthesizer board, and a Radio Shack compatible edge connector. An interconnect cable (supplied with the Model CT-1T) connects the unit to the TRS-80 bus connector on either the keyboard or expansion interface. Standard phone jacks provide connections for external speakers, headphones or external amplifier (not provided).

The Model CT-1T can be operated in two modes: direct parameter control and phonetic, and it is supported by a growing library of software. Each unit is shipped with a hardware user manual, basic set of software consisting of CTEDIT Parameter Data Editor and speech parameter data files Hello, Letters and Digits, and the Computalker CSR1 Synthesizer-by-Rule Software program. All software is available in a choice of 5 inch disk or standard cassette.

The CT-1T is priced at $595. A special unit is available for persons who already own a Model CT-1 and is priced at $225. For further information, contact Computalker Consultants, 1730 21st St, Suite A, Santa Monica CA 90404.

Circle 530 on inquiry card.

# HOBBY WORLD®

**CALL TOLL FREE: (800) 423-5387**
**CA, HI, AK: (213) 886-9200**

## Your No.1 Source for Computer Electronics

## Computer Boards

CCS = California Computer Systems
WMC = Wameco
IA = Ithaca Audio
SSM = Solid State Music
SDS = SD Systems
SPL = Speechlab
HUH = HUH Electronics
MH = Mountain Hardware
a&t = assembled & tested

### S100 Bus

| Cat No. | Mfr/Model | Description | Price |
|---|---|---|---|
| 1601A | CCS-MXVI | 16K STATIC RAM MODULE, 450ns kit | $285 |
| 1601B | CCS-MXVI | as above, 200ns kit | $330 |
| 1602A | CCS-MXVI | as above, 450ns a&t | $330 |
| 1602B | CCS-MXVI | as above, 200ns a&t | $375 |
| 1603 | CCS-MXVI | as above, bareboard | $ 27 |
| 1600 | CCS-PT1 | WIREWRAP PROTOTYPING BOARD | $ 26 |
| 1604 | CCS-PT2 | SOLDERTAIL PROTOTYPING BOARD | $ 26 |
| 1609 | CCS-PT3 | ETCH PROTOTYPING BOARD | $ 16 |
| 1440 | SSM-PB1 | 4K/8K EPROM BOARD | $135 |
| 1442 | SSM-T1 | TERMINATOR BOARD | $ 29 |
| 1405 | SSM-MB7 | 16K STATIC RAM BOARD, kit | $329 |
| 1406 | SSM-MB7 | as above, a&t | $383 |
| 1407 | SSM-MB7 | as above, bareboard | $ 26 |
| 1425 | SSM-MB3 | 2K/4K EPROM BOARD, kit | $ 54 |
| 1426 | SSM-MB3 | as above, a&t | $108 |
| 1436 | SSM-MB9 | 4K STATIC PROM/RAM BOARD, kit | $ 64 |
| 1437 | SSM-MB9 | as above, a&t | $118 |
| 1429 | SSM-OB1 | VECTOR JUMP/PROTOTYPING CARD, kit | $47 |
| 1430 | SSM-OB1 | as above, a&t | $ 74 |
| 1431 | SSM-OB1 | as above, bareboard | $ 26 |
| 1414 | SSM-IO2 | UNIVERSAL I/O BOARD, kit | $ 48 |
| 1415 | SSM-IO2 | as above, a&t | $ 93 |
| 1416 | SSM-IO2 | as above, bareboard | $ 26 |
| 1408 | SSM-SB1 | MUSIC SYNTHESIZER, kit | $145 |
| 1409 | SSM-SB1 | as above a&t | $212 |
| 1410 | SSM-SB1 | as above, bareboard | $ 35 |
| 1432 | SSM-MT1 | 15 SLOT MOTHERBOARD, bareboard | $ 39 |
| 1411 | SSM-IO4 | 2 PARALLEL + 2 SERIAL PORTS, kit | $139 |
| 1412 | SSM-IO4 | as above, a&t | $193 |
| 1413 | SSM-IO4 | as above, bareboard | $ 26 |
| 1403 | SSM-CB1 | 8080A CPU BOARD, kit | $119 |
| 1441 | SSM-CB1 | as above, bareboard | $ 34 |
| 1404 | SSM-CB1 | as above, a&t | $164 |
| 1417 | SSM-VB1B | VIDEO INTERFACE BOARD, kit | $129 |
| 1418 | SSM-VB1B | as above, a&t | $176 |
| 1419 | SSM-VB1B | as above, bareboard | $ 26 |
| 1400 | SSM-MB6B | 8K STATIC RAM BOARD, kit | $129 |
| 1401 | SSM-MB6B | as above, a&t | $183 |
| 1402 | SSM-MB6B | as above, bareboard | $ 26 |
| 1433 | SSM-MB8A | 16K 2708 EPROM BOARD, kit | $ 88 |
| 1434 | SSM-MB8A | as above, a&t | $118 |
| 1435 | SSM-MB8A | as above, bareboard | $ 26 |
| 1438 | SSM-VB2 | VIDEO BOARD, kit | $139 |
| 1439 | SSM-VB2 | as above, a&t | $199 |
| 1420 | SSM-MB4 | STATIC RAM BOARD, kit, 2MHz | $ 89 |
| 1422 | SSM-MB4 | as above, a&t | $142 |
| 1424 | SSM-MB4 | as above, bareboard | $ 26 |
| 1324 | WMC-EPM1 | 4K EPROM BOARD, bareboard | $ 28 |
| 1325 | WMC-EPM2 | 16/32K EPROM BOARD, bareboard | $ 28 |
| 1322 | WMC-FPB1 | FRONT PANEL BOARD, bareboard | $ 48 |
| 1323 | WMC-CPU1 | 8080A CPU BOARD, bareboard | $ 28 |
| 1337 | WMC-FDC1 | FLOPPY DISK CONTROLLER BOARD, bareboard | $ 48 |
| 1509 | WMC-MEM1 | 8K STATIC RAM KIT, bareboard | $ 28 |
| 1510 | WMC-MEM1 | parts only for above | $ 80 |
| 1511 | IA | 2708/16 EPROM BOARD, bareboard | $ 28 |
| 1513 | IA | 8K STATIC RAM, bareboard | $ 28 |
| 1512 | IA | Z-80A CPU BOARD, bareboard | $ 32 |
| 1514 | IA | WIRE WRAP PROTOTYPING BOARD | $ 24 |
| 1500 | HUH-MPA | PET & APPLE TO S-100 ADAPTER, kit | $199 |
| 1501 | HUH-MPA | as above, a&t | $279 |
| 1505 | MH | 100,000 DAY CLOCK, a&t | $219 |
| 1506 | MH | INTROL | $329 |
| 1518 | SPL-20S | 32 WORD SPEECHLAB, a&t | $189 |
| 1520 | SPL-50S | 64 WORD SPEECHLAB, a&t | $299 |
| 1516 | SDS | VERSAFLOPPY, kit | $159 |
| 1517-0 | SDS | EXPANDORAM, kit | $185 |
| 1517-16K | SDS | as above. with 16K RAM | $249 |
| 1517-32K | SDS | as above. with 32K RAM | $330 |
| 1517-48K | SDS | as above. with 48K RAM | $425 |
| 1517-64K | SDS | as above. with 64K RAM | $500 |

### APPLE BUS

| Cat No. | Mfr/Model | Description | Price |
|---|---|---|---|
| 1607 | CCS | SOLDERTAIL PROTOTYPING BOARD | $ 21 |
| 1606 | CCS | WIREWRAP PROTOTYPING BOARD | $ 21 |
| 1608 | CCS | ETCH PROTOTYPING BOARD | $ 18 |
| 1519 | SPL | 32 WORD SPEECHLAB | $189 |

## FINALLY! PANAVISE at a Discount

LOW PROFILE BASE
Cat No. PV305
~~$13.49~~ $10.50

BASE TRAY
Cat No. PV312
~~$ 9.95~~ $ 8.50

VACUUM BASE
Cat No. PV380
~~$18.49~~ $14.50

STANDARD BASE
Cat No. PV300
~~$13.49~~ $10.50

BASE MOUNTING CLAMP
Cat No. PV311
~~$12.95~~ $10.00

WIDE OPENING VISE HEAD
Cat No. PV366
~~$14.49~~ $11.25

STANDARD VISE HEAD
Cat No. PV303
~~$14.49~~ $11.25

HORIZONTAL VISE HEAD
Cat No. PV304
~~$15.49~~ $12.00

PC BOARD VISES
Cat No. PV315 Standard ~~$18.98~~ $14.75
Cat No. PV315 S-100 ~~$21.95~~ $16.75

## IC Sockets Penny-A-Pin

- *Texas Instruments*
- *Solder tail*
- *Package quantities only*

Order by Cat No. 1117 and pins

| | |
|---|---|
| 8 pin | 20 for $1.60 |
| 14 pin | 10 for $1.40 |
| 16 pin | 10 for $1.60 |
| 18 pin | 8 for $1.44 |
| 20 pin | 8 for $1.60 |
| 22 pin | 8 for $1.76 |
| 24 pin | 3 for .72 |
| 28 pin | 3 for .84 |
| 40 pin | 2 for .80 |

## TRS232 Printer Serial Interface $42

Software driven RS232 output port. Interface printers such as DIABLO, TELETYPE, TI SILENT, etc. Easy to install, with instructions.
Cat No. 1199 $42.00

## 16K Memory Add-On $95

- *For TRS-80, Apple II*

Guaranteed memory expansion! Includes everything you need...chips, jumpers, and step by step instructions. No special tools, no soldering!
Cat No. 1156

## TRS 80 Level III Basic

Loads on top of Level II, turns your TRS-80 into a powerful system. Gives disk commands without the DOS. Solves loading problems, cures keyboard bounce. Software cassette, with the power of a hardware modification. Guaranteed satisfaction.
Cat No. 1332 $49.00

**$49**

## "LIFT-IT" PC Duplicating Kit $25

Copy PC patterns from magazines quickly and easily! No additional materials required, no camera needed. With 10x12" sheet of film. We stock all refills. With step-by-step instructions.
Cat No. 1203 $25

## TTL's

*Order by type number*

| | | | |
|---|---|---|---|
| 7400 | .15 | 7490 | .45 |
| 7402 | .20 | 7491 | .55 |
| 7403 | .20 | 7492 | .45 |
| 7404 | .20 | 7493 | .45 |
| 7405 | .20 | 7495 | .65 |
| 7406 | .30 | 7496 | .65 |
| 7407 | .30 | 74107 | .35 |
| 7408 | .20 | 74109 | .55 |
| 7409 | .20 | 74121 | .35 |
| 7410 | .20 | 74122 | .40 |
| 7411 | .25 | 74123 | .60 |
| 7412 | .25 | 74126 | .40 |
| 7413 | .35 | 74132 | .65 |
| 7414 | .60 | 74141 | .80 |
| 7416 | .25 | 74145 | .65 |
| 7417 | .25 | 74150 | .90 |
| 7420 | .20 | 74151 | .60 |
| 7423 | .25 | 74153 | .60 |
| 7425 | .25 | 74154 | 1.00 |
| 7426 | .25 | 74155 | .70 |
| 7427 | .25 | 74157 | .60 |
| 7430 | .25 | 74160 | .85 |
| 7432 | .25 | 74161 | .75 |
| 7437 | .25 | 74163 | .75 |
| 7438 | .25 | 74165 | .95 |
| 7440 | .20 | 74166 | 1.25 |
| 7441 | .70 | 74170 | 1.60 |
| 7442 | .50 | 74173 | 1.10 |
| 7443 | .50 | 74174 | .85 |
| 7444 | .25 | 74175 | .70 |
| 7445 | .75 | 74176 | .80 |
| 7446 | .75 | 74177 | .75 |
| 7447 | .65 | 74180 | .65 |
| 7450 | .20 | 74181 | 1.75 |
| 7451 | .20 | 74190 | 1.25 |
| 7453 | .20 | 74191 | .95 |
| 7454 | .20 | 74192 | .75 |
| 7470 | .30 | 74193 | .75 |
| 7472 | .25 | 74194 | .90 |
| 7473 | .35 | 74195 | .70 |
| 7474 | .35 | 74198 | 1.49 |
| 7475 | .45 | 74221 | .85 |
| 7476 | .35 | 74251 | .80 |
| 7481 | 1.20 | 74273 | 1.00 |
| 7483 | .60 | 74365 | .65 |
| 7485 | .75 | 74366 | .65 |
| 7486 | .35 | 74367 | .65 |
| 7489 | 1.75 | 74368 | .65 |

## HUH 8100 TRS-80 to S100 Bus Adaptor

- *6 Slot motherboard*

Includes options such as serial RS232/20ma I/O, parallel input and output, space for 4K or 16K RAM, and more. Basic unit includes Bus interface and connectors only.

| | | |
|---|---|---|
| Cat No. 1460 | Bus interface kit | $167 |
| Cat No. 1461 | Ram Support kit | $ 42 |
| Cat No. 1462 | I/O Option kit | $ 77 |
| Cat No. 1463 | 5 connectors + guides | $ 42 |
| Cat No. 1464 | Complete Package | $266 |
| Cat No. 1465 | Bus Interface a&t | $220 |
| Cat No. 1466 | Complete Pkg. a&t | $340 |

## Voltage Regulators

| Type | Equal to | CASE | PRICE |
|---|---|---|---|
| 7805K | LM340K + 5 | TO-3 | $1.40 |
| 7812K | LM340K + 12 | TO-3 | $1.40 |
| 7815K | LM340K + 15 | TO-3 | $1.40 |
| 7818K | LM340K + 18 | TO-3 | $1.40 |
| 7805T | LM340T + 5 | TO-220 | .95 |
| 7812T | LM340T + 12 | TO-220 | .95 |
| 7815T | LM340T + 15 | TO-220 | .95 |
| 7818T | LM34oT + 18 | TO-220 | .95 |
| 7905K | LM320K-5 | TO-3 | $1.75 |
| 7912K | LM430K-12 | TO-3 | $1.75 |
| 7905T | LM320T-5 | TO-220 | $1.25 |
| 7912T | LM320T-12 | TO-220 | $1.25 |
| 7915T | LM320T-15 | TO-220 | $1.25 |

## LEEDEX 12" Video Monitor $139

- *TRS-80 compatible*

*(no interfacing required)*

- *High resolution*

Accepts standard composite video input, no RF modulator necessary. Stable & sharp. Lightweight. Shipping $6 gnd, $12 air, USA only.
Cat No. 1204 $139.00

## Send for FREE CATALOG Featuring

The best selection of computer accessories add-ons, factory fresh IC's, led's, semi's, software, PC aids, prototyping aids, books, test equipment, and more! Always updated! Dozens of new products every issue!

Pay by check, COD, Visa, or Mastercharge. Order by phone or mail. Please include phone number and magazine issue you are ordering from. USA: add $2 for shipping/handling, $3 for air. Foreign: add $3 for surface, $6 for air. COD's also add $1. Guaranteed satisfaction for 120 days or your money back!

**19355 Business Center Dr. 6B8**
**Northridge CA 91324**

Circle 170 on inquiry card.

asap computer products, inc.

11542-1 KNOTT STREET
GARDEN GROVE, CALIFORNIA 92641
(714) 891-2663

## MICROBYTE Z80/I-O

- A complete single board Z80A CPU and serial/paralle I/O system
- Fully S-100 Bus compatible, IMSAI, ALTAIR
- Z80A CPU (4 MHz version of the Z80)
- 158 instructions — superset of and upward compatible from the 8080's 78 instructions
- Provision for up to 4K on board monitor program using 1K (2708), 2K (2716), 4K 2732
- On board EPROM can be hardware and/or software deselected
- 2 MHz or 4 MHz operation is switch selectable
- 0 or 1 wait state for all cycles is switch selectable
- 2 RS-232C serial ports with 8251 USARTs
- Serial baud rates switch selectable
- 24 programmable parallel I/O lines (uses 8255)
- Gold Contacts for higher reliability
- Power requirements: +8V @ 800mA, +16V, @ 86mA, − 16V @ 100mA
- Operating temperature 0°-55°C
- Will operate with or without IMSAI/ALTAIR front panel
- Low power shottky tri-state buffers on all address and data lines
- Fully warranted for 120 days from date of shipment

**$325.⁰⁰**

## IMSAI CONN.

100 PIN-SOLDERTAIL
GOLD CONTACTS
**$3.⁰⁰ each or 10/2.60 each**

## 8251

PROGRAMMABLE/U-ART
TESTED @ 4 MHZ
**$5.⁰⁰ each**

## TRS - 80

Floppy disk drive with cabinet & pwr. supply compatible with Radio Shack interface. Assembled & tested with 1 yr. warranty on parts & labor.

Mfg. by Lobo Drive **$385.⁰⁰**

Interface Cable Available

## SPECIAL

.1 @ 12 VOLTS
CERAMIC CAP
**10¢ each**
**or**
**100 / $9.⁰⁰**

## MICROBYTE 16K STATIC RAM BOARD

- Fully S100 Bus Compatible, IMSAI, SOL, ALTAIR, ALPHA MICRO
- Uses National's Low Power 5257 4K × 1 Static Rams
- 2 MHz or 4 MHz operation
- On board single 5 amp regulator
- Thermally designed heat sink (board operating temperature 0° — 70°C)
- Inputs fully low power Shottky Schmitt Trigger buffered on all address and data lines
- Phantom is jumper selectable to pin 67
- Each 4K bank addressable to any 4K slot with in a 64K boundary.
- 4K hardware or software selectable
- Selectable port address
- 4K banks can be selected or disabled on power on clear or reset
- Will operate with or without front panel
- Compatible with ALPHA MICRO, with extended memory management for selection beyond 64K
- No DMA restriction
- Low power consumption 1.3 amp
- Fully warranted for 120 days from date of shipment
- **Extended addressing up to 1 megabyte of addressable ram**

**450 NS $320.⁰⁰**
**300 NS $340.⁰⁰**

## SHUGART 801-Disk Drive

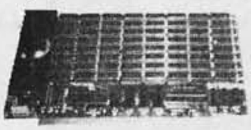
WITH CABINET & POWER SUPPLY
ASSEMBLED & TESTED
1 YR PARTS & LABOR
Mfg. by Lobo Drive
**$585.⁰⁰**
Dual Cabinet & Drives Available

## 2708's

LOW POWER
450 NS.
**$8.⁷⁵ each**
**8 for $66.⁰⁰**

## MICROBYTE 32K STATIC RAM BOARD

- Fully S100 Bus Compatible, IMSAI, SOL, ALTAIR, ALPHA MICRO
- Uses National's Low Power 5257 4K × 1 Static Rams
- 2 MHz or 4 MHz operation
- On board single 5 amp regulator
- Thermally designed heat sink (board operating temperature 0° — 70°C)
- Inputs fully low power Shottky Schmitt Trigger buffered on all address and data lines
- Phantom is jumper selectable to pin 67
- Each 4K bank addressable to any 4K slot with in a 64K boundary.
- 4K hardware or software selectable
- One on board 8-bit output port enables or disables the 32K in 4K blocks
- Selectable port address
- 4K banks can be selected or disabled on power on clear or reset
- Will operate with or without front panel
- Compatible with ALPHA MICRO, with extended memory management for selection beyond 64K
- No DMA restriction
- Low power consumption 2.3 — 2.5 amps
- Fully warranted for 120 days from date of shipment.
- **Extended addressing up to 1 megabyte of addressable ram**



**450 NS $620.⁰⁰**
**300 NS $650.⁰⁰**

## SHUGART SA400

DISK DRIVE INCLUDES CABINET, NO PWR SUPPLY, CUTOUTS FOR SWITCH, FUSE, & INTERFACE CABLE
Mfg. by Lobo Drive
**$325.⁰⁰**

## 2716

5 VOLT ONLY
LOW POWER
450 ns
**$40.⁰⁰**

## SCANBE / RN SOCKETS — LO PROFILE

(tin)

| | 1-24 | 25-99 | 100-499 | 500 up |
|---|---|---|---|---|
| 14 PIN | .16 | .15 | .14 | .12 |
| 16 PIN | .17 | .16 | .15 | .14 |
| 18 PIN | .20 | .19 | .18 | .16 |
| 20 PIN | .29 | .28 | .26 | .25 |
| 24 PIN | .34 | .32 | .30 | .28 |
| 40 PIN | .60 | .58 | .56 | .52 |

## REGULATORS

| | 1-9 | 10-49 | 50up |
|---|---|---|---|
| 320 T-5 | 1.25 | 1.15 | 1.05 |
| 320 T-12 | 1.00 | .90 | .85 |
| 340 T5 | .75 | .70 | .65 |
| 340 T-12 | .75 | .70 | .65 |
| 78 H05 | 6.00 | 5.70 | 5.40 |

## MICROBYTE MOTHERBOARD

- Active Diode termination
- Slot for IMSAI front panel
- Terminal block connection for easy hook-up
- Extra wide ground plane
- Silk screen and solder mask
- Assembled and tested

**9 slot kit $70.⁰⁰ A&T $100.⁰⁰**
**20 slot kit $125.⁰⁰ A&T $155.⁰⁰**
**Bare Board 9 slot $30.⁰⁰ 20 slot $50.⁰⁰**

## NEW PRODUCTS

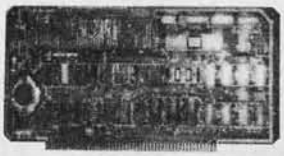
8086 — CPU BOARD
8088 — CPU BOARD
Double Density Controller
CALL OR WRITE
FOR DEALER INFO

## CABLE ASSEMBLY

**for 8" disk drives**
(2) 50 PIN CARD-EDGE CONNECTORS ON 4ft. RIBBON CABLE
**$20.⁰⁰ea.**
extra conn. $7.⁰⁰ ea.

## MICROBYTE DISK CONTROLLER

- IBM 3740 Soft Sectored Compatible
- Z80 or 8080 compatible on S-100 Bus
- Single density runs both mini and full size drives, runs CPM, on Shugart, Persci, Memorex etc.
- Selectable port/address
- On board 2708/2716 for bootstrap or monitor program
- No hardware jumpers, uses plug in modules for different drives
- Uses 1771B-01 controller chip
- Assembled and tested
- **Specify disk drive used when ordering by mail**



**$225.⁰⁰**

**ORDERING INFORMATION:**
Name, Address, Phone
Ship by: UPS or P.P.
Shipping Charge: Add $2.50 up to 5 lbs., all excess shipping charges will be refunded. Credit cards will be charged appropriate freight.

**TERMS:**
We accept cash, check, money orders, Visa, and Master Charge cards. (U.S. Funds Only).
COD's: on approval only
Open Acct's: companies may inquire for net terms.
Tax: add 6% for Calif. residents only

Circle 11 on inquiry card.

## TRS-80 USERS

Loweco Computer Introduces 3 Diskless TRS-80 Programs

- Telephone/Address/Mailing list program - sorts by name or zip code. Retrieves telephone # from name and visa versa. Access time is under 2 seconds, over 100 listings, Level II 16K - $30.00
- Checkbook program - hard electronic copy, easily accessible. 170 listings, Level II 16K - $30.00
- Mandalas for the Cybernectic Age I & II - amazing graphic programs, better than TV - lasts hours without repeating or commercials. 2 sets of four interweaving designs, Level I or II 4K RAM - $30.00
- Introductory offer - all 3 for $75.00

**LOWECO COMPUTOR**
**1803 Rodney**
**LOS ANGELES CA 90027**
**213-660-7530**

6% Tax in California
Cashier Check Speeds Delivery

Circle 207 on inquiry card.

**Radio Shack® DEALER**

MICRO MANAGEMENT SYSTEMS

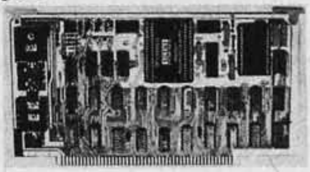
## Up To 15% Discount on

# TRS-80's

MICRO-COMPUTER SPECIALIST
LARRY OWENS
**COMPUTER CENTER**

MINI MALL
DOWNTOWN SHOPPING CENTER
CAIRO, GEORGIA 31728
912-377-7120

## NEW S-100 A/D



- S-100 Bus Compatible A/D Converter
- 12 Bit Accuracy
- 16 Channel Analog Input
- Programmable Gain Amplifier with Sample-and-Hold
- High Quality Commercial/ Industrial Construction

2 and 4 Channel, 12 Bit D/A Boards also available.

CDC
CALIFORNIA DATA CORPORATION
*3475 Old Conejo Road, Suite C10*
*Newbury Park, California 91320*
*(805) 498-3651*

Circle 41 on inquiry card.

# KIM SOFTWARE

**9K MICROSOFT BASIC**
**Includes:**

- Over 55 Commands
- Full String Handling
- 9 Digit Precision
- Hypertape Built-In
- 70 Page Manual

**SPECIAL**
**INCLUDES "DATA/SAVE"**
**(added commands to record both programs and data!)**
**KIM CASSETTE & MANUAL**
**$100.00 prepaid**

***UPDATE KIT & MANUAL FOR KIM BASIC WITHOUT MICRO-Z FEATURES . . . $35.00***

**MICRO-Z COMPANY**
**Box 2426**
**Rolling Hills, CA 90274**

Circle 234 on inquiry card.

APPLE ][
## RESET KEY PROTECTOR

RESET

*FAST RELIEF for the ACCIDENTAL RESET BLUES !*

- * *Prevents reset when reaching for = and return*
- * *Allows deliberate usage*
- * *Custom, precision molded part* * *Easily installed*
- * *"Original equipment" look*

*only $2.25 - incl. ship.& hndl.*
*(calif. res. add 6% tax)*
*send check or mon. order to:*
**pkc. inc.** *dept. B*
*2003 Quail St., N.B. CA 92660*

Circle 299 on inquiry card.


# The best choice in mainframes !

- S100 CARD FRAME
- 22 MHz 12" CRT MONITOR
- 18 AMP POWER SUPPLY
- UPPER & LOWER CASE ASCII KEY BOARDS
- AXIAL BLOWER
- ASSEMBLED & TESTED
- READY FOR YOUR CARDS
- $995.00

*VERY ATTRACTIVE O.E.M. AND DEALER DISCOUNTS AVAILABLE*

**INFINITE INCORPORATED**
*Celebrating Our 11th Year.*
819 E. STRAWBRIDGE, MELBOURNE, FL 32901 – (305) 724-1588

Circle 174 on inquiry card.

**BYTE's New Toll-free Subscriber W.A.T.S. Line (800) 258-5485**

To further improve service to our customers we have installed a toll-free WATS line in our Peterborough, New Hampshire office. If you would like to order a subscription to BYTE, or if you have a question related to a BYTE subscription, you are invited to call (800) 258-5485 between 8:30 AM and 4:30 PM Eastern Time. This applies to calls from within the continental US only.

*We thank you and look forward to serving you.*

## 16K RAMS & RAM CONTROLLERS

16k × 1 DYNAMIC RAM MK4116P3
- 200 NSEC ACCESS/375 NSEC CYCLE TIMES
- 16 PIN CERAMIC DEVICE/TTL COMPATIBLE
- ALL CHIPS BURNED IN 8 125°C AND FULLY TESTED, BOTH DYNAMICALLY AND STATICALLY
- PRICE (WITH DATA SHEET):
- $68.00 IN QTY OF 8/THAT'S $8.50 EACH

DYNAMIC MEMORY CONTROLLER MC3480L
- GENERATES RAS/CAS & REFRESH TIMING FOR 16K TO 64K BYTE MEMORIES
- PRICE (WITH DATA SHEET): $13.95 EACH

MEMORY ADDRESS MUX/COUNTER MC3242AP
- MUX ADDRESS & REFRESH COUNTER FOR 16K TO 64K BYTE MEMORIES
- PRICE (WITH DATA SHEET): $12.50 EACH

QUANTITY DISCOUNTS AVAILABLE

ALL ORDERS POSTPAID. U.S. FUNDS ON INTERNATIONAL ORDERS, CHECK OR MONEY ORDER, VISA/BA/MC ALSO ACCEPTED. SEND ACCT. NO., EXPIRATION DATE, & INTERBANK NO. WITH SIGNED ORDER. CALIF. RESIDENTS PLEASE ADD 6% SALES TAX. PHONE ORDERS: (714) 633-4460.

**MEASUREMENT SYSTEMS & CONTROLS, INC.**
MEMORY DEVICES DIVISION
867 NORTH MAIN ST., ORANGE, CA 92668

Circle 217 on inquiry card.

## 6800 MICROPROCESSOR PARTS

(Brand New Limited Quantities)

M6800 CPU . . . . . . . . . . . . . $8.95
M6821 PIA . . . . . . . . . . . . . . $4.95
M6810 RAM . . . . . . . . . . . . . $4.95
2708 EPROM (8K) . . . . . . $9.95
2716 EPROM (16K) . . . . $24.95

Complete 6800 computer board kit with M6800, M6821, M6810, 2708 EPROM, p.c. board, and power supply ready for operation on 115 VAC with full instructions (with 2716 EPROM, add $10.00) . . . . . . . . . . . $49.95

**ODS INC.**
**P.O. BOX 2346**
**Gaithersburg, MD 20760**

Circle 286 on inquiry card.

Circle 4 on inquiry card.

### Extended FORTRAN Preprocessor

X4 is an extended FORTRAN preprocessor for use in the Cromemco CDOS environment. X4 translates programs into standard FORTRAN from a readable, well structured language providing modern control structures for conditionals and iteration that virtually eliminate the need for statement numbers and GOTO statements. X4 also provides automatic file inclusion, a macro facility, mixed upper and lower case input, and the expansion of quoted strings into numeric character codes where desired. X4 is available with complete documentation on CDOS (CP/M) format 5 inch floppy disk for $59.95. Contact Modular Systems Inc, 4005 Seven Mile Ln, Pikesville MD 21208.

Circle 531 on inquiry card.

### Word Processor For TRS-80 Disk Operating System

Word-III is a text processor for the TRS-80 disk operating system. Requiring 16 K bytes of memory, it accepts lines of text interspersed with lines of format control information and then formats the text into a displayable document. Word-III features automatic line adjusting, margin right justification, page numbering, centering, title, page size, line width, indentation, and vertical spacing control. It is written in TRS-80 Disk BASIC for easy loading and expansion. Word-III is disk based with a size limited by disk storage. It uses the printer interface that already exists in the expansion module. Instructions are given to make software modification to other printers not using 1PRINT command.

The price of Word-III is $39 complete with source code. For further information contact Micro Architect, 96 Dothan St, Arlington MA 02174.

Circle 532 on inquiry card.

### Business Software Series in BASIC

The *Standard Software Library* is a series of books containing listings of programs written in BASIC with complete documentation. Each volume in the series is devoted to a single application. The first three volumes deal with accounting programs for small computers. Volume I, *General Ledger* enables a small business to set up a fully automated general ledger system with a complete chart of accounts. Included are programs for editing, sorting, merging and posting of transactions. A trial balance report is available in either summary or detail at the user's option. Income statement and balance sheet reports may be obtained at the close of each accounting period with both current and year to date totals and percentages.

Volume 2, *Accounts Receivable* provides a fully automated system for dealing with customer accounts. Volume 3, *Payroll* enables a business to automate all of the normal payroll functions. All of the programs are written in a level of BASIC common to practically all of the current microprocessors and minicomputers. The modular nature of the programs and the accompanying documentation make it easy to revise the program to meet special user requirements.

The documentation includes an overall view of the program, a list of the variables used, a description of the required user inputs and an illustrative example with sample output reports. Annotated comments are contained in all of the programs.

Contact Creative Computer Consultants Inc, POB 2111, Norwalk CT 06852.

Circle 535 on inquiry card.

### 6502 Robot Language

Written in 6502 machine language, Robot is an interactive programming language for the control of robots. The robot may be a Turtle, plotter, or video cursor. The heart of Robot is a command processing module designed to allow the user to design a language of personalized commands and command subroutines to suit a particular application.

The version of Robot that is being offered includes a command set and subroutine package for the control of a video robot. The subroutines are designed specifically for the TVT-6 video interface, but will work with any memory mapped video display and can be adapted by the user for varying formats. Robot takes slightly more than 1 K bytes of programmable memory and comes with a user manual and a completely commented source listing.

Robot is priced at $5 (add $3 for KIM-1 Hypertape cassette). For further information contact Michael Allen, 6025 Kimbark, Chicago IL 60637. This vendor also offers a 6502 tiny editor and assembler.

Circle 533 on inquiry card.

### Free Monthly Review of Software Products Available

Users of Northstar BASIC can receive a free subscription to John Dvorak's *Software Review*. Each month the software review examines and reviews new software packages and reports on the relative merits and value of the product. At the moment the mailing list has focused on users of Northstar BASIC but plans are in the works to introduce a newsletter for users of CP/M oriented systems, TRS-80 and eventually Apple users. For a free subscription, write to J Dvorak, 704 Solano Av, Albany CA 94706.

Circle 534 on inquiry card.

### Microcomputer Text Editor

Edit-80 is a random access, line oriented editor for 8080 and Z-80 systems. It provides almost instantaneous access to any record of the file, even if the available memory space is considerably smaller than the file being edited. In addition to the standard line commands to insert, delete, print or replace lines of text, Edit-80 offers many other features such as automatic line renumbering, global find and substitute, multiple page files and ability to read in files without Edit-80 line numbers. Edit-80's alter mode provides a complete set of intraline subcommands to edit portions of individual lines. With Edit-80, the edited file is not written to disk until a write command is given, and the original file is always saved as back-up.

The Edit-80 Text Editing Package includes a file compare utility program called FILCOM which compares source or binary files and outputs differences between them.

Edit-80 runs on any 8080 or Z-80 system with the CP/M operating system. The price for the Edit-80 Text Editing Package is $120 and the manual is available for $10. For further information contact Microsoft, 300 San Mateo NE, Suite 819, Albuquerque NM 87108.

Circle 536 on inquiry card.

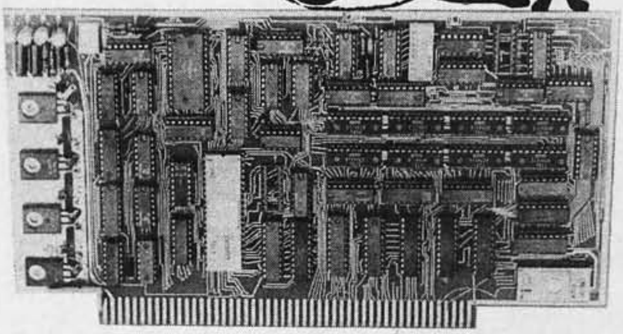
### The Realty Expense Analysis Program

REAP is designed for the property owner or manager and provides complete expense information for each building in payment-by-payment and summary format which includes tax ready totals for IRS filing. The building payee report displays expenses for any building, for all or selected payees. The utility summary report displays yearly, year-to-date, or monthly average utility expenses for each building under the categories electric, gas, water, and trash. The tax totals report displays totals for each building under the categories utilities, insurance, repairs and property tax. Special accounts may be set up to track auto, general office management, advertising, telephone or any other expense type. Complete data inputing, editing, and sorting capabilities, all with extensive error recovery, provide easy data file maintanence. Expense data may be added to the file and the latest reports run at any time interval.

REAP is available on cassette with complete documentation for the TRS-80 Level I and II, Apple, and PET computers. Each 16 K bytes of user memory will handle 500 yearly expense payments. Larger data files are possible by using disk data storage. REAP is priced at $25. Documentation only with sample reports is $2.50. For further information contact Realty Software Co, 2045 Manhattan Av, Hermosa Beach CA 90254.

Circle 537 on inquiry card.

# Smart & Foxy



## Venus 2001 Video Board

**Assembled & Tested $259.95 • Complete Unit with 4K Memory and Video Driver on Eprom assembled and tested $339.95**

*OPTIONAL:* • Sockets $10.00
• 2K Memory $30.00
• 4K Memory $60.00
• Video Driver Eprom $20.00

**kit $199⁹⁵**

***• S-100 plug-in • Parallel keyboard port***

On board 4K Screen Memory (Optional). On board Eprom (Optional) for Video Driver or Text Editor Software.

***Up and down scrolling through video memory*** Reverse Video, Blinking Characters.

***Display :*** 128 ASC11 Characters 64 X 32 or 32 X 16 Screen format (Jumper Selectable). 7 by 11 Dot Matrix Characters.

***American or European TV Compatible***
**(CRT Controls Programable)** *Dealer Inquiries Invited*

## 32-K Static RAM $499. KIT

• S-100 Plug-In • Kit includes P.C. board, all parts and assembly manual • Uses 2114L, 450 nS.
I.C. sockets - $20.00
P.C. BOARD BY S-100 CO.

## 16-K Static RAM $249. KIT

• S-100 Plug-In Kit includes P.C. board, all parts and assembly manual. Uses 2114L 450 nS.
Sockets - $10.00
Add $40.00 for 300 nS (4MHz) RAMS
P.C. BOARD BY WAMECO

## Z-80 CPU $125. KIT

• S-100 Plug-In Kit includes P.C. boards, all parts and assembly manual.
**FEATURES:** 2MHz operation • S-100 plug-in • Power-on jump • On board provision for 2708 (optional at $12.95).
P.C. BOARD BY ITHACA AUDIO

## *ASCII Keyboard Kit $79.95*

Assembled and Tested $ 95.95

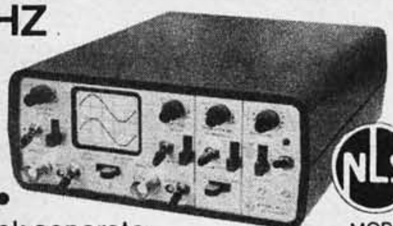
• Single +5V Supply • Full ASCII Set (Upper and Lower Case) • Parallel Output • Positive and Negetave Strobe • 2 Key Rollover • 3 User Definable Keys • P.C. Board Size: 17-3/16" X 5" • Control Characters Molded on Key Caps • Optional Provision For Serial Output
*OPTIONAL:* Metal Enclosure $27.50 • Edge Con. $2.00 • Sockets $4.00 • Upper Case Lock Switch $2.50 • Shift Register (For Serial Output) $2.00
*Dealer Inquiries Invited*

## *Apple II I/O Board Kit*

Plugs intc Slot of Mother Board
•1 8 Bit Parallel Output Port (Expands to 3 Ports) • 1 Input Port • 15mA Output Current Sink or Source • Can be used for peripheral equipment such as printers, floppy discs, cassettes, paper tapes, etc. • 1 free software listing for SWTP PR40 or IBM selectric.
*PRICE:* 1 Input and 1 Output Port $49.00
1 Input and 3 Output Ports $64.00
*Dealer Inquiries Invited*

## new ! A DREAM COME TRUE !

Introducing: **30 MHZ**
***DUAL TRACE PORTABLE SCOPE*** for an amazing **$555.**



NLS
MODEL MS 230

• Dual trace 2-channel; separate, chopped or alternate modes. • 30 megahertz bandwidth. • External and internal trigger.
• Time base - 0.05 , Microseconds to 0.2 SEC/div 21 settings • Battery or line operation.
• Line synchronization mode.
• Power consumption less than 50W. • Vertical gain - 0.1 to 50 volts/div- 12 settings. • Size: 2.9" H 6.4" W 8.5" D. • Weighs only 3.5 lbs. with batteries.
• Complete with input cable and rechargeable batteries and charger unit.
*OPTIONAL:* Leather case $45.00 • 10:1 probe $27.00 (2 for $49.00)

**MS-215**
15 MHZ ***Dual Trace Portable Scope*** **$399.**

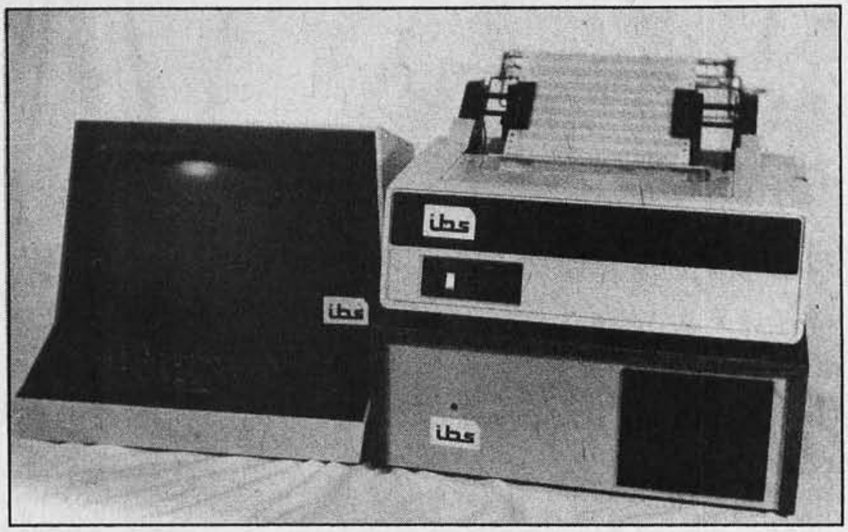
**MS-15** 15 MHZ ***Single Trace Scope*** **$299.**

SHIPPING $3.50 / California residents add 6% sales tax

**ELECTRONICS WAREHOUSE Inc.**
15820 Hawthorne Boulevard
Lawndale, CA 90260
(213) 370-5551

master charge VISA

### New Fully Implemented Pascal System



The Independent Business System's Betasystem is a complete operating system that features the UCSD implementation of Pascal. The operating system contains a powerful screen oriented text editor, a fast Pascal compiler, file and library handling systems, linker, Z-80 assembler and more. This Z-80 microprocessor comes complete with 48 K byte programmable memory, dual quad density (630 K byte formatted) disk drives, serial and parallel ports, 60 character per second dot matrix printer with tractor feed, and intelligent terminal with addressable cursor. It sells for $5485. For further information contact Independent Business Systems Inc, 5476 Cleo Ct, Livermore CA 94550.

Circle 635 on inquiry card.

### 16 K Byte Programmable Read Only Memory Board

Electronic Solutions 16 K byte programmable read only memory board is compatible with the Intel SBC 80 bus and single board computer. The PROM-16 accepts sixteen 2708 erasable read only memories. The board has a convenient addressing scheme allowing jumper selection of the board base address at the beginning of any 4 K block. Any number of 1 K byte memory blocks may be deselected by jumper removal, thus freeing these 1 K byte memory addresses for the processor, programmable memory, etc. When fully loaded with sixteen 2708 erasable read only memories, the board typically draws 0.31 A (from +5 V), 0.48 A (from −5 V), and 0.80 A (from +12 V). For further information, contact Electronic Solutions Inc, 7969 Engineer Rd, San Diego CA 92111.

Circle 558 on inquiry card.





Circle 354 on inquiry card.

# 7400 TTL

| | | | | | |
|---|---|---|---|---|---|
| SN7400N | .16 | SN7470N | .29 | SN74160N | .89 |
| SN7401N | .18 | SN7472N | .29 | SN74161N | .89 |
| SN7402N | .18 | SN7473N | .35 | SN74162N | 1.95 |
| SN7403N | .18 | SN7474N | .35 | SN74163N | .89 |
| SN7404N | .18 | SN7475N | .49 | SN74164N | .89 |
| SN7405N | .20 | SN7476N | .35 | SN74165N | .89 |
| SN7406N | .29 | SN7479N | 5.00 | SN74166N | 1.25 |
| SN7407N | .29 | SN7480N | .50 | SN74167N | 1.95 |
| SN7408N | .20 | SN7482N | .99 | SN74170N | 1.59 |
| SN7409N | .20 | SN7483N | .59 | SN74172N | 6.00 |
| SN7410N | .18 | SN7485N | .79 | SN74173N | 1.25 |
| SN7411N | .25 | SN7486N | .35 | SN74174N | .89 |
| SN7412N | .25 | SN7489N | 1.75 | SN74175N | .79 |
| SN7413N | .40 | SN7490N | .45 | SN74176N | .79 |
| SN7414N | .70 | SN7491N | .59 | SN74177N | .79 |
| SN7416N | .25 | SN7492N | .43 | SN74179N | 1.95 |
| SN7417N | .25 | SN7493N | .43 | SN74180N | .79 |
| SN7420N | .20 | SN7494N | .65 | SN74181N | 1.95 |
| SN7421N | .29 | SN7495N | .65 | SN74182N | .79 |
| SN7422N | .39 | SN7496N | .65 | SN74184N | 1.95 |
| SN7423N | .25 | SN7497N | 3.00 | SN74185N | 1.95 |
| SN7425N | .29 | SN74100N | .89 | SN74186N | 9.95 |
| SN7426N | .29 | SN74107N | .35 | SN74188N | 3.95 |
| SN7427N | .25 | SN74109N | .59 | SN74190N | 1.25 |
| SN7429N | .39 | SN74116N | 1.95 | SN74191N | 1.25 |
| SN7430N | .20 | SN74121N | .35 | SN74192N | .79 |
| SN7432N | .25 | SN74122N | .39 | SN74193N | .79 |
| SN7437N | .25 | SN74123N | .49 | SN74194N | .89 |
| SN7438N | .25 | SN74125N | .49 | SN74195N | .69 |
| SN7439N | .25 | SN74126N | .49 | SN74196N | .89 |
| SN7440N | .20 | SN74132N | .75 | SN74197N | .89 |
| SN7441N | .89 | SN74136N | .75 | SN74198N | 1.49 |
| SN7442N | .49 | SN74141N | .79 | SN74199N | 1.49 |
| SN7443N | .75 | SN74142N | 2.95 | SN74S200 | 4.95 |
| SN7444N | .75 | SN74143N | 2.95 | SN74251N | 1.79 |
| SN7445N | .75 | SN74144N | 2.95 | SN74279N | .79 |
| SN7446N | .69 | SN74145N | .79 | SN74283N | 2.25 |
| SN7447N | .59 | SN74147N | 1.95 | SN74284N | 3.95 |
| SN7448N | .79 | SN74148N | 1.29 | SN74285N | 3.95 |
| SN7450N | .20 | SN74150N | .89 | SN74365N | .69 |
| SN7451N | .20 | SN74151N | .59 | SN74366N | .69 |
| SN7453N | .20 | SN74152N | .59 | SN74367N | .69 |
| SN7454N | .20 | SN74153N | .59 | SN74368N | .69 |
| SN7459A | .25 | SN74154N | .99 | SN74390N | 1.95 |
| SN7460N | .20 | SN74155N | .79 | SN74393N | 1.95 |
| | | SN74156N | .79 | | |
| | | SN74157N | .65 | | |

# C/MOS

| | | | | | |
|---|---|---|---|---|---|
| CD4000 | .23 | | | CD4070 | .55 |
| CD4001 | .23 | | | CD4071 | .23 |
| CD4002 | .23 | CD4028 | .89 | CD4072 | .49 |
| CD4006 | 1.19 | CD4029 | 1.19 | CD4076 | 1.39 |
| CD4007 | .25 | CD4030 | .49 | CD4081 | .23 |
| CD4009 | .49 | CD4035 | .99 | CD4082 | .23 |
| CD4010 | .49 | CD4040 | 1.19 | CD4093 | .99 |
| CD4011 | .23 | CD4041 | 1.25 | CD4098 | 2.49 |
| CD4012 | .25 | CD4042 | .99 | MC14409 | 14.95 |
| CD4013 | .39 | CD4043 | .89 | MC14410 | 14.95 |
| CD4014 | 1.39 | CD4044 | .89 | MC14411 | 14.95 |
| CD4015 | 1.19 | CD4046 | 1.79 | MC14419 | 4.95 |
| CD4016 | .49 | CD4047 | 2.50 | MC14433 | 19.95 |
| CD4017 | 1.19 | CD4048 | 1.35 | MC14506 | .75 |
| CD4018 | .99 | CD4049 | .49 | MC14507 | .99 |
| CD4019 | .49 | CD4050 | .49 | MC14562 | 14.50 |
| CD4020 | 1.19 | CD4051 | 1.19 | MC14583 | 3.50 |
| CD4021 | 1.39 | CD4053 | 1.19 | CD4508 | 3.95 |
| CD4022 | 1.19 | CD4056 | 2.95 | CD4510 | 1.39 |
| CD4023 | .23 | CD4059 | 9.95 | CD4511 | 1.29 |
| CD4024 | .79 | CD4060 | 1.49 | CD4515 | 2.95 |
| CD4025 | .23 | CD4066 | .79 | CD4518 | 1.29 |
| CD4026 | 2.25 | CD4068 | .39 | CD4520 | 1.29 |
| CD4027 | .69 | CD4069 | .45 | CD4566 | 2.25 |

# 74C00

| | | | | | |
|---|---|---|---|---|---|
| 74C00 | .39 | | | 74C163 | 2.49 |
| 74C02 | .39 | | | 74C164 | 2.49 |
| 74C04 | .39 | 74C85 | 2.49 | 74C173 | 2.60 |
| 74C08 | .49 | 74C90 | 1.95 | 74C192 | 2.49 |
| 74C10 | .39 | 74C93 | 1.95 | 74C193 | 2.49 |
| 74C14 | 1.95 | 74C95 | 1.95 | 74C195 | 2.49 |
| 74C20 | .39 | 74C107 | 1.25 | 74C922 | 5.95 |
| 74C30 | .39 | 74C151 | 2.90 | 74C923 | 6.25 |
| 74C42 | 1.95 | 74C154 | 3.00 | 74C925 | 8.95 |
| 74C48 | 2.49 | 74C157 | 2.15 | 74C926 | 8.95 |
| 74C73 | .89 | 74C160 | 2.49 | 80C95 | 1.50 |
| 74C74 | .89 | 74C161 | 2.49 | 80C97 | 1.50 |

# LINEAR

| | | | | | |
|---|---|---|---|---|---|
| 78MG | 1.75 | | | LM710N | .79 |
| LM106H | .99 | | | LM711N | .39 |
| LM300H | .80 | LM340K-18 | 1.35 | LM723N/H | .55 |
| LM301CN/H | .35 | LM340K-24 | 1.35 | LM733N | 1.00 |
| LM302H | .75 | LM340T-5 | 1.25 | LM739N | 1.19 |
| LM304H | 1.00 | LM340T-6 | 1.25 | LM741CN/H | .35 |
| LM305H | .60 | LM340T-8 | 1.25 | LM741-14N | .39 |
| LM307CN/H | .35 | LM340T-12 | 1.25 | LM747N/H | .79 |
| LM308CN/H | 1.00 | LM340T-15 | 1.25 | LM748N/H | .39 |
| LM309H | 1.10 | LM340T-18 | 1.25 | LM1310N | 2.95 |
| LM309K | 1.25 | LM340T-24 | 1.25 | LM1458CN/H | .59 |
| LM310CN | 1.15 | LM358N | 1.00 | MC1488N | 1.39 |
| LM311N/H | .90 | LM370N | 1.95 | MC1489N | 1.39 |
| LM312H | 1.95 | LM373N | 3.25 | LM1496N | .95 |
| LM317K | 6.50 | LM377N | 4.00 | LM1556V | 1.75 |
| LM318CN/H | 1.50 | LM380N | 1.25 | MC1741SCP | 3.00 |
| LM319N | 1.30 | LM380CN | .99 | LM2111N | 1.95 |
| LM320K-5 | 1.35 | LM381N | 1.79 | LM2901N | 2.95 |
| LM320K-5.2 | 1.35 | LM382N | 1.79 | LM3053N | 1.50 |
| LM320K-12 | 1.35 | NE501N | 8.00 | LM3065N | 1.49 |
| LM320K-15 | 1.35 | NE510A | 6.00 | LM3900N(3401) | .49 |
| LM320K-18 | 1.35 | NE529A | 4.95 | LM3905N | .89 |
| LM320K-24 | 1.35 | NE531H/V | 3.95 | LM3909N | 1.25 |
| LM320T-5 | 1.25 | NE536T | 6.00 | MC5558V | .59 |
| LM320T-5.2 | 1.25 | NE540L | 6.00 | 8038B | 4.95 |
| LM320T-8 | 1.25 | NE544N | 4.95 | LM75450N | .49 |
| LM320T-12 | 1.25 | NE550N | 1.30 | 75451CN | .39 |
| LM320T-15 | 1.25 | NE555V | .39 | 75452CN | .39 |
| LM320T-18 | 1.25 | NE556N | .99 | 75453CN | .39 |
| LM320T-24 | 1.25 | NE560B | 5.00 | 75454CN | .39 |
| LM323K-5 | 5.95 | NE561B | 5.00 | 75491CN | .79 |
| LM324N | 1.80 | NE562B | 5.00 | 75492CN | .89 |
| LM339N | .99 | NE565N/H | 1.25 | 75493N | .89 |
| LM340K-5 | 1.35 | NE566CN | 1.75 | 75494CN | .89 |
| LM340K-6 | 1.35 | NE567V/H | .99 | RC4136 | 1.25 |
| LM340K-8 | 1.35 | NE570N | 4.95 | RC4151 | 2.85 |
| LM340K-12 | 1.35 | LM703CN/H | .69 | RC4194 | 5.95 |
| LM340K-15 | 1.35 | LM709N/H | .29 | RC4195 | 4.49 |

# 74LS00TTL

| | | | | | |
|---|---|---|---|---|---|
| 74LS00 | .29 | | | 74LS138 | .89 |
| 74LS01 | .29 | | | 74LS139 | .89 |
| 74LS02 | .29 | 74LS47 | .89 | 74LS151 | .89 |
| 74LS03 | .29 | 74LS51 | .29 | 74LS155 | .89 |
| 74LS04 | .35 | 74LS54 | .29 | 74LS157 | .89 |
| 74LS05 | .35 | 74LS55 | .29 | 74LS160 | 1.15 |
| 74LS08 | .29 | 74LS73 | .45 | 74LS161 | 1.15 |
| 74LS09 | .35 | 74LS74 | .45 | 74LS162 | 1.15 |
| 74LS10 | .29 | 74LS75 | .59 | 74LS163 | 1.15 |
| 74LS11 | .75 | 74LS76 | .45 | 74LS164 | 1.25 |
| 74LS13 | .59 | 74LS78 | .49 | 74LS175 | .99 |
| 74LS14 | 1.25 | 74LS83 | .89 | 74LS181 | 2.49 |
| 74LS15 | .35 | 74LS85 | 1.25 | 74LS190 | 1.15 |
| 74LS20 | .29 | 74LS86 | .45 | 74LS191 | 1.15 |
| 74LS21 | .35 | 74LS90 | .59 | 74LS192 | 1.15 |
| 74LS22 | .35 | 74LS92 | .75 | 74LS193 | 1.15 |
| 74LS26 | .35 | 74LS93 | .75 | 74LS194 | 1.15 |
| 74LS27 | .35 | 74LS95 | .99 | 74LS195 | 1.15 |
| 74LS28 | .35 | 74LS96 | 1.15 | 74LS253 | .99 |
| 74LS30 | .29 | 74LS107 | .45 | 74LS257 | .89 |
| 74LS32 | .35 | 74LS109 | .45 | 74LS258 | 1.75 |
| 74LS37 | .45 | 74LS112 | .45 | 74LS260 | .69 |
| 74LS40 | .35 | 74LS123 | 1.25 | 74LS279 | .75 |
| 74LS42 | .89 | 74LS125 | .89 | 74LS367 | .75 |
| | | 74LS132 | .99 | 74LS368 | .75 |
| | | 74LS136 | .49 | 74LS670 | 2.49 |

# EXCITING NEW KITS

## NEW JE600 HEXADECIMAL ENCODER KIT

FEATURES:
- Full 8 bit latched output for microprocessor use
- 3 User Define keys with one being bistable operation
- Debounce circuit provided for all 19 keys
- LED readout to verify entries
- Easy interfacing with standard 16 pin IC connector
- Only +5VDC required for operations

FULL 8 BIT LATCHED OUTPUT—19 KEYBOARD

The JE600 Encoder Keyboard provides two separate hexadecimal digits produced from sequential key entries to allow direct programming for 8 bit microprocessor or 8 bit memory circuits. Three (3) additional keys are provided for user operations with one having a bistable output available. The outputs are latched and monitored with LED readouts. Also included is a key entry strobe.

**JE600 .......... $59.95**

## Digital Thermometer Kit

NEW

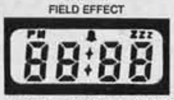
- Dual sensors—switching control for indoor/outdoor or dual monitoring
- Continuous LED .8" ht. display
- Range: -40°F to 199°F / -40°C to 100°C
- Accuracy: ±1° nominal
- Set for Fahrenheit or Celsius reading
- Sim. walnut case - AC wall adapter incl.
- Size: 3-1/4"Hx6-5/8"Wx1-3/8"D

**JE300 ....... $39.95**

# DISCRETE LEDS

| | | | | | |
|---|---|---|---|---|---|
| .200" dia. | | | .125" dia. | | |
| XC556R | red | 5/$1 | XC209R | red | 5/$1 |
| XC556G | green | 4/$1 | XC209G | green | 4/$1 |
| XC556Y | yellow | 4/$1 | XC209Y | yellow | 4/$1 |
| XC556C | clear | 4/$1 | | | |
| .200" dia. | | | .185" dia. | | |
| XC22R | red | 5/$1 | XC526R | red | 5/$1 |
| XC22G | green | 4/$1 | XC526G | green | 4/$1 |
| XC22Y | yellow | 4/$1 | XC526Y | yellow | 4/$1 |
| .170" dia. | | | XC526C | clear | 4/$1 |
| MV10B | red | 4/$1 | .190" dia. | | |
| .085" dia. | | | XC111R | red | 5/$1 |
| MV50 | red | 6/$1 | XC111G | green | 4/$1 |
| INFRA-RED LED 1/4"x1/4"x1/16" flat | | 5/$1 | XC111Y | yellow | 4/$1 |
| | | | XC111C | clear | 4/$1 |

TIMEX T1001
LIQUID CRYSTAL DISPLAY
CLASS II
FIELD EFFECT



4 DIGIT - .5" CHARACTERS
THREE ENUNCIATORS
2.00" X 1.20" PACKAGE
INCLUDES CONNECTOR

T1001-Transmissive $7.95
T1001A-Reflective 8.25

# DISPLAY LEDS

| TYPE | POLARITY | HT | PRICE | TYPE | POLARITY | HT | PRICE |
|---|---|---|---|---|---|---|---|
| MAN 1 | Common Anode-red | .270 | 2.95 | MAN 6730 | Common Anode-red ± 1 | .560 | .99 |
| MAN 2 | 5 x 7 Dot Matrix-red | .300 | 4.95 | MAN 6740 | Common Cathode-red-D.D. | .560 | .99 |
| MAN 3 | Common Cathode-red | .125 | .25 | MAN 6750 | Common Cathode-red ± 1 | .560 | .99 |
| MAN 4 | Common Cathode-red | .187 | 1.95 | MAN 6760 | Common Anode-red | .560 | .99 |
| MAN 7G | Common Anode-green | .300 | 1.25 | MAN 6780 | Common Cathode-red | .560 | .99 |
| MAN 7Y | Common Anode-yellow | .300 | .99 | DL701 | Common Anode-red ± 1 | .300 | .99 |
| MAN 72 | Common Anode-red | .300 | .99 | DL704 | Common Cathode-red | .300 | .99 |
| MAN 74 | Common Cathode-red | .300 | 1.25 | DL707 | Common Anode-red | .300 | .99 |
| MAN 82 | Common Anode-yellow | .300 | .99 | DL728 | Common Cathode-red | .500 | 1.49 |
| MAN 84 | Common Cathode-yellow | .300 | .99 | DL741 | Common Anode-red | .600 | 1.25 |
| MAN 3620 | Common Anode-orange | .300 | .99 | DL746 | Common Anode-red ± 1 | .630 | 1.49 |
| MAN 3630 | Common Anode-orange ± 1 | .300 | .99 | DL747 | Common Anode-red | .600 | 1.49 |
| MAN 3640 | Common Cathode-orange | .300 | .99 | DL749 | Common Cathode-red ± 1 | .630 | 1.49 |
| MAN 4610 | Common Anode-orange | .300 | .99 | DL750 | Common Cathode-red | .600 | 1.49 |
| MAN 4640 | Common Cathode-orange | .400 | .99 | DL338 | Common Cathode-red | .110 | .35 |
| MAN 4710 | Common Anode-red | .400 | .99 | FND70 | Common Cathode | .250 | .69 |
| MAN 4730 | Common Anode-red ± 1 | .400 | .99 | FND358 | Common Cathode ± 1 | .357 | .99 |
| MAN 4740 | Common Cathode-red | .400 | .99 | FND359 | Common Cathode | .357 | .75 |
| MAN 4810 | Common Anode-yellow | .400 | .99 | FND503 | Common Cathode(FND500) | .500 | .99 |
| MAN 4840 | Common Cathode-yellow | .400 | .99 | FND507 | Common Anode (FND510) | .500 | .99 |
| MAN 6610 | Common Anode-orange-D.D. | .560 | .99 | 5082-7730 | Common Anode-red | .300 | 1.30 |
| MAN 6630 | Common Anode-orange ± 1 | .560 | .99 | HDSP-3400 | Common Anode-red | .800 | 2.10 |
| MAN 6640 | Common Cathode-orange-D.D. | .560 | .99 | HDSP-3403 | Common Cathode red | .800 | 2.10 |
| MAN 6650 | Common Cathode-orange ± 1 | .560 | .99 | 5082-7300 | 4 x 7 sgl. Digit-RHDP | .600 | 19.95 |
| MAN 6660 | Common Anode-orange | .560 | .99 | 5082-7302 | 4 x 7 Sgl. Digit-LHDP | .600 | 19.95 |
| MAN 6680 | Common Cathode-orange | .560 | .99 | 5082-7304 | Overrange character (±1) | .600 | 15.00 |
| MAN 6710 | Common Anode-red-D.D. | .560 | .99 | 5082-7340 | 4 x 7 Sgl. Digit-Hexadecimal | .600 | 22.50 |

# RCA LINEAR

| | | | |
|---|---|---|---|
| CA3013T | 2.15 | CA3082N | 2.00 |
| CA2023T | 2.56 | CA3083N | 1.60 |
| CA3035T | 2.48 | CA3086N | .85 |
| CA3039T | 1.35 | CA3089N | 3.75 |
| CA3046N | 1.30 | CA3130T | 1.39 |
| CA3059N | 3.25 | CA3140T | 1.25 |
| CA3060N | 3.25 | CA3160T | 1.25 |
| CA3080T | .85 | CA3401N | .49 |
| CA3081N | 2.00 | CA3600N | 3.50 |

# CALCULATOR CHIPS/DRIVERS

| | |
|---|---|
| MM5725 | $2.95 |
| MM5738 | 2.95 |
| DM8864 | 2.00 |
| DM8865 | 1.00 |
| DM8887 | .75 |
| DM8889 | .75 |
| 9374 7 seg. C.A. LED driver | 1.50 |

# CLOCK CHIPS

| | |
|---|---|
| MM5309 | $4.95 |
| MM5311 | 4.95 |
| MM5312 | 4.95 |
| MM5314 | 4.95 |
| MM5316 | 6.95 |
| MM5318 | 9.95 |
| MM5369 | 2.95 |
| MM5387/1998A | 4.95 |
| MM5841 | 9.95 |

# MOTOROLA

| | |
|---|---|
| MC1408L7 | $4.95 |
| MC1408L8 | 5.75 |
| MC1439L | 2.95 |
| MC3022P | 2.95 |
| MC3061P | 3.50 |
| MC4016(74416) | 7.50 |
| MC4024P | 3.95 |
| MC4040P | 6.95 |
| MC4044P | 4.50 |

# IC SOLDERTAIL — LOW PROFILE (TIN) SOCKETS

| | 1-24 | 25-49 | 50-100 | | 1-24 | 25-49 | 50-100 |
|---|---|---|---|---|---|---|---|
| 8 pin LP | .$17 | .16 | .15 | 22 pin LP | $ .37 | .36 | .35 |
| 14 pin LP | .20 | .19 | .18 | 24 pin LP | .38 | .37 | .36 |
| 16 pin LP | .22 | .21 | .20 | 28 pin LP | .45 | .44 | .43 |
| 18 pin LP | .29 | .28 | .27 | 36 pin LP | .60 | .59 | .58 |
| 20 pin LP | .34 | .32 | .30 | 40 pin LP | .63 | .62 | .61 |

## SOLDERTAIL STANDARD (TIN)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 14 pin ST | $.27 | .25 | .24 | 28 pin ST | $ .99 | .90 | .81 |
| 16 pin ST | .30 | .27 | .25 | 36 pin ST | 1.39 | 1.26 | 1.15 |
| 18 pin ST | .35 | .32 | .30 | 40 pin ST | 1.59 | 1.45 | 1.30 |
| 24 pin ST | .49 | .45 | .42 | | | | |

## SOLDERTAIL STANDARD (GOLD)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 pin SG | $.30 | .27 | .24 | 24 pin SG | $ .70 | .63 | .57 |
| 14 pin SG | .35 | .32 | .29 | 28 pin SG | 1.10 | 1.00 | .90 |
| 16 pin SG | .38 | .35 | .32 | 36 pin SG | 1.65 | 1.40 | 1.26 |
| 18 pin SG | .52 | .47 | .43 | 40 pin SG | 1.75 | 1.59 | 1.45 |

## WIRE WRAP SOCKETS (GOLD) LEVEL #3

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 pin WW | $.39 | .38 | .31 | 22 pin WW | $ .95 | .85 | .75 |
| 10 pin WW | .45 | .41 | .37 | 24 pin WW | 1.05 | .95 | .85 |
| 14 pin WW | .39 | .38 | .37 | 28 pin WW | 1.40 | 1.25 | 1.10 |
| 16 pin WW | .43 | .42 | .41 | 36 pin WW | 1.59 | 1.45 | 1.30 |
| 18 pin WW | .75 | .68 | .62 | 40 pin WW | 1.75 | 1.55 | 1.40 |

# 1/4 WATT RESISTOR ASSORTMENTS – 5%

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ASST. 1 | 5 ea. | 10 OHM / 27 OHM | 12 OHM / 33 OHM | 15 OHM / 39 OHM | 18 OHM / 47 OHM | 22 OHM / 56 OHM | 50 PCS | $1.75 |
| ASST. 2 | 5 ea. | 68 OHM / 180 OHM | 82 OHM / 220 OHM | 100 OHM / 270 OHM | 120 OHM / 330 OHM | 150 OHM / 390 OHM | 50 PCS | 1.75 |
| ASST. 3 | 5 ea. | 470 OHM / 1.2K | 560 OHM / 1.5K | 680 OHM / 1.8K | 820 OHM / 2.2K | 1K / 2.7K | 50 PCS | 1.75 |
| ASST. 4 | 5 ea. | 3.3K / 8.2K | 3.9K / 10K | 4.7K / 12K | 5.6K / 15K | 6.8K / 18K | 50 PCS | 1.75 |
| ASST. 5 | 5 ea. | 22K / 56K | 27K / 68K | 33K / 82K | 39K / 100K | 47K / 120K | 50 PCS | 1.75 |
| ASST. 6 | 5 ea. | 150K / 390K | 180K / 470K | 220K / 560K | 270K / 680K | 330K / 820K | 50 PCS | 1.75 |
| ASST. 7 | 5 ea. | 1M / 2.7M | 1.2M / 3.3M | 1.5M / 3.9M | 1.8M / 4.7M | 2.2M / 5.6M | 50 PCS | 1.75 |

ASST. 8R Includes Resistor Assortments 1-7 (350 PCS.) $9.95 ea.

$10.00 Min. Order — U.S. Funds Only
Calif. Residents Add 6% Sales Tax
Postage — Add 5% plus $1 Insurance (if desired)

Spec Sheets — 25¢
1979 Catalog Available — Send 41¢ stamp

FREE A 1979 CATALOG

**Jameco ELECTRONICS**

PHONE ORDERS WELCOME (415) 592-8097

*MAIL ORDER ELECTRONICS – WORLDWIDE*
1021 HOWARD AVENUE, SAN CARLOS, CA 94070
ADVERTISED PRICES GOOD THRU AUGUST

# TELEPHONE/KEYBOARD CHIPS

| | | |
|---|---|---|
| AY-5-9100 | Push Button Telephone Dialler | $14.95 |
| AY-5-9200 | Repertory Dialler | 14.95 |
| AY-5-9500 | CMOS Clock Generator | 4.95 |
| AY-5-2376 | Keyboard Encoder (88 keys) | 14.95 |
| HD0165 | Keyboard Encoder (16 keys) | 7.95 |
| 74C922 | Keyboard Encoder (16 keys) | 5.95 |

# ICM CHIPS

| | | |
|---|---|---|
| ICM7045 | CMOS Precision Timer | 24.95 |
| ICM7205 | CMOS LED Stopwatch/Timer | 19.95 |
| ICM7207 | Oscillator Controller | 7.50 |
| ICM7208 | Seven Decade Counter | 19.95 |
| ICM7209 | Clock Generator | 6.95 |

# NMOS READ ONLY MEMORIES

| | | |
|---|---|---|
| MCM6571 | 128 X 9 X 7 ASCII Shifted with Greek | 13.50 |
| MCM6574 | 128 X 9 X 7 Math Symbol & Pictures | 13.50 |
| MCM6575 | 128 X 9 X 7 Alphanumeric Control Character Generator | 13.50 |

# MISCELLANEOUS

| | | |
|---|---|---|
| TL074CN | Quad Low Noise bi-fet Op Amp | 2.49 |
| TL494CN | Switching Regulator | 4.49 |
| TL496CP | Single Switching Regulator | 1.75 |
| 11C90 | Divide 10/11 Prescaler | 19.95 |
| 95H90 | Hi-Speed Divide 10/11 Prescaler | 11.95 |
| 4N33 | Photo-Darlington Opto-Isolator | 3.95 |
| MK50240 | Top Octave Freq. Generator | 17.50 |
| DS0026CH | 5Mhz 2-phase MOS clock driver | 3.75 |
| TIL308 | .27" red num. display w/integ. logic chip | 10.50 |
| MM5320 | TV Camera Sync. Generator | 14.95 |
| MM5330 | 4½ Digit DPM Logic Block (Special) | 3.95 |
| LD110/111 | 3½ Digit A/D Converter Set | 25.00/set |

**LITRONIX ISO-LIT 1**
Photo Transistor Opto-Isolator
(Same as MCT 2 or 4N25)
**2/99¢**

**SN 76477**
SOUND GENERATOR
Generates Complex Sounds
Low Power - Programmable
**3.95 each**

**TV GAME CHIP AND CRYSTAL**
AY-3-8500-1 and 2.01 MHZ Crystal (Chip & Crystal includes score display, 6 games and select angles, etc. **7.95/set**

# EXAR

| | | | | | |
|---|---|---|---|---|---|
| XR205 | $8.40 | | | XR2242CP | 1.50 |
| XR210 | 4.40 | | | XR2264 | 4.25 |
| XR215 | 4.40 | | | XR2556 | 3.20 |
| XR320 | 1.55 | JE2206KA | 14.95 | XR2567 | 2.99 |
| XR-L555 | 1.50 | JE2206KB | 19.95 | XR3403 | 1.25 |
| XR555 | .39 | XR1800 | 3.20 | XR4136 | 1.25 |
| XR556 | .99 | XR2206 | 4.40 | XR4151 | 2.85 |
| XR567CP | .99 | XR2207 | 3.85 | XR4194 | 4.95 |
| XR567CT | 1.25 | XR2208 | 5.20 | XR4202 | 3.60 |
| XR1310P | 1.30 | XR2209 | 1.75 | XR4212 | 2.05 |
| XR1468CN | 3.85 | XR2211 | 5.25 | XR4558 | .75 |
| XR1488 | 1.39 | XR2212 | 4.35 | XR4739 | 1.15 |
| XR1489 | 1.39 | XR2240 | 3.45 | XR4741 | 1.47 |

# DIODES

| TYPE | VOLTS | W | PRICE | TYPE | VOLTS | W | PRICE |
|---|---|---|---|---|---|---|---|
| | | | | 1N4002 | 100 PIV | 1 AMP | 12/1.00 |
| | | | | 1N4003 | 200 PIV | 1 AMP | 12/1.00 |
| 1N746 | 3.3 | 400m | 4/1.00 | 1N4004 | 400 PIV | 1 AMP | 12/1.00 |
| 1N751 | 5.1 | 400m | 4/1.00 | 1N4005 | 600 PIV | 1 AMP | 10/1.00 |
| 1N752 | 5.6 | 400m | 4/1.00 | 1N4006 | 800 PIV | 1 AMP | 10/1.00 |
| 1N753 | 6.2 | 400m | 4/1.00 | 1N4007 | 1000 PIV | 1 AMP | 10/1.00 |
| 1N754 | 6.8 | 400m | 4/1.00 | 1N3600 | 50 | 200m | 6/1.00 |
| 1N757 | 9.0 | 400m | 4/1.00 | 1N4148 | 75 | 10m | 15/1.00 |
| 1N759 | 12.0 | 400m | 4/1.00 | 1N4154 | 35 | 10m | 12/1.00 |
| 1N959 | 8.2 | 400m | 4/1.00 | 1N4305 | 75 | 25m | 15/1.00 |
| 1N965 | 15 | 400m | 4/1.00 | 1N4734 | 5.6 | 1w | 28 |
| 1N5232 | 5.6 | 500m | 28 | 1N4735 | 6.2 | 1w | 28 |
| 1N5234 | 6.2 | 500m | 28 | 1N4736 | 6.8 | 1w | 28 |
| 1N5235 | 6.8 | 500m | 28 | 1N4738 | 8.2 | 1w | 28 |
| 1N5236 | 7.5 | 500m | 28 | 1N4742 | 12 | 1w | 28 |
| 1N5242 | 12 | 500m | 28 | 1N4744 | 15 | 1w | 28 |
| 1N5245 | 15 | 500m | 28 | 1N1183 | 50 PIV | 35 AMP | 1.60 |
| 1N456 | 25 | 40m | 6/1.00 | 1N1184 | 100 PIV | 35 AMP | 1.70 |
| 1N458 | 150 | 7m | 6/1.00 | 1N1185 | 150 PIV | 35 AMP | 1.70 |
| 1N485A | 180 | 10m | 5/1.00 | 1N1186 | 200 PIV | 35 AMP | 1.80 |
| 1N4001 | 50 PIV | 1 AMP | 12/1.00 | 1N1188 | 400 PIV | 35 AMP | 3.00 |

# SCR AND FW BRIDGE RECTIFIERS

| | | | |
|---|---|---|---|
| C36D | 15A @ 400V | SCR(2N1849) | $1.95 |
| C38M | 35A @ 600V | SCR | 1.95 |
| 2N2328 | 1.6A @ 300V | SCR | .50 |
| MDA 980-1 | 12A @ 50V | FW BRIDGE REC. | 1.95 |
| MDA 980-3 | 12A @ 200V | FW BRIDGE REC. | 1.95 |

# TRANSISTORS

| | | | | | |
|---|---|---|---|---|---|
| C106B1 | .50 | | | 2N3904 | 4/1.00 |
| MPSA05 | .30 | 2N3055 | .89 | 2N3905 | 4/1.00 |
| MPSA06 | 5/1.00 | MJE3055 | 1.00 | 2N3906 | 4/1.00 |
| TIS97 | 6/1.00 | 2N3392 | 5/1.00 | 2N4013 | 3/1.00 |
| TIS98 | 6/1.00 | 2N3398 | 5/1.00 | 2N4123 | 6/1.00 |
| 40409 | 1.75 | PN3567 | 3/1.00 | PN4249 | 4/1.00 |
| 40410 | 1.75 | PN3568 | 4/1.00 | PN4250 | 4/1.00 |
| 40673 | 1.75 | PN3569 | 4/1.00 | 2N4400 | 4/1.00 |
| 2N918 | 4/1.00 | MPS3638A | 5/1.00 | 2N4401 | 4/1.00 |
| 2N2219A | 2/1.00 | MPS3702 | 5/1.00 | 2N4402 | 4/1.00 |
| 2N2221A | 4/1.00 | 2N3704 | 5/1.00 | 2N4403 | 4/1.00 |
| 2N2222A | 5/1.00 | MPS3704 | 5/1.00 | 2N4409 | 5/1.00 |
| PN2222 Plastic | 7/1.00 | 2N3705 | 5/1.00 | 2N5086 | 4/1.00 |
| 2N2369 | 5/1.00 | MPS3705 | 5/1.00 | 2N5087 | 4/1.00 |
| 2N2369A | 4/1.00 | 2N3706 | 5/1.00 | 2N5088 | 4/1.00 |
| MPS2369 | 5/1.00 | MPS3706 | 5/1.00 | 2N5089 | 4/1.00 |
| 2N2484 | 4/1.00 | 2N3707 | 5/1.00 | 2N5129 | 5/1.00 |
| 2N2906 | 4/1.00 | 2N3711 | 5/1.00 | PN5134 | 5/1.00 |
| 2N2907 | 5/1.00 | 2N3724A | .65 | PN5138 | 5/1.00 |
| PN2907 Plastic | 7/1.00 | 2N3725A | 1.00 | 2N5139 | 5/1.00 |
| 2N2925 | 5/1.00 | 2N3772 | 2.25 | 2N5210 | 5/1.00 |
| MJE2955 | 1.25 | 2N3823 | 1.00 | 2N5449 | 3/1.00 |
| 2N3053 | 2/1.00 | 2N3903 | 5/1.00 | 2N5951 | 3/1.00 |

# CAPACITOR CORNER

## 50 VOLT CERAMIC DISC CAPACITORS

| | 1-9 | 10-99 | 100+ | | 1-9 | 10-99 | 100+ |
|---|---|---|---|---|---|---|---|
| 10 pf | .05 | .04 | .03 | .001µF | .05 | .04 | .035 |
| 22 pf | .05 | .04 | .03 | .0047µF | .05 | .04 | .035 |
| 47 pf | .05 | .04 | .03 | .01µF | .05 | .04 | .035 |
| 100 pf | .05 | .04 | .03 | .022µF | .06 | .05 | .04 |
| 220 pf | .05 | .04 | .03 | .047µF | .06 | .05 | .04 |
| 470 pf | .05 | .04 | .035 | .1µF | .12 | .09 | .075 |

## 100 VOLT MYLAR FILM CAPACITORS

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| .001mf | .12 | .10 | .07 | .022mf | .13 | .11 | .08 |
| .0022 | .12 | .10 | .07 | .047mf | .21 | .17 | .13 |
| .0047mf | .12 | .10 | .07 | .1mf | .27 | .23 | .17 |
| .01mf | .12 | .10 | .07 | .22mf | .33 | .27 | .22 |

## +20% DIPPED TANTALUMS (SOLID) CAPACITORS

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| .1/35V | .28 | .23 | .17 | 1.5/35V | .30 | .26 | .21 |
| .15/35V | .28 | .23 | .17 | 2.2/25V | .31 | .27 | .22 |
| .22/35V | .28 | .23 | .17 | 3.3/25V | .31 | .27 | .22 |
| .33/35V | .28 | .23 | .17 | 4.7/25V | .32 | .28 | .23 |
| .47/35V | .28 | .23 | .17 | 6.8/25V | .36 | .31 | .25 |
| .68/35V | .28 | .23 | .17 | 15/25V | .63 | .50 | .40 |
| 1.0/35V | .28 | .23 | .17 | | | | |

## MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS

| Axial Lead | | | | Radial Lead | | | |
|---|---|---|---|---|---|---|---|
| .47/50V | .15 | .13 | .10 | .47/25V | .15 | .13 | .10 |
| 1.0/50V | .16 | .14 | .11 | .47/50V | .16 | .14 | .11 |
| 3.3/50V | .14 | .12 | .09 | 1.0/16V | .15 | .13 | .10 |
| 4.7/25V | .15 | .13 | .10 | 1.0/25V | .16 | .14 | .11 |
| 10/25V | .15 | .13 | .10 | 1.0/50V | .16 | .14 | .11 |
| 10/50V | .16 | .14 | .12 | 4.7/16V | .15 | .13 | .10 |
| 22/25V | .17 | .15 | .12 | 4.7/25V | .15 | .13 | .10 |
| 22/50V | .24 | .20 | .18 | 4.7/50V | .16 | .14 | .11 |
| 47/25V | .19 | .17 | .15 | 10/16V | .14 | .12 | .09 |
| 47/50V | .25 | .21 | .19 | 10/25V | .15 | .13 | .10 |
| 100/25V | .24 | .20 | .18 | 10/50V | .16 | .14 | .12 |
| 100/50V | .35 | .30 | .28 | 47/50V | .24 | .21 | .19 |
| 220/25V | .32 | .28 | .25 | 100/16V | .19 | .15 | .14 |
| 220/50V | .45 | .41 | .38 | 100/25V | .24 | .20 | .18 |
| 470/25V | .33 | .29 | .27 | 100/50V | .35 | .30 | .28 |
| 1000/16V | .55 | .50 | .45 | 220/16V | .23 | .17 | .16 |
| 2200/16V | .70 | .62 | .55 | 470/25V | .31 | .28 | .26 |



Circle 200 on inquiry card.

# COMPUCOLOR II IS HERE!!

**HERE IS A SURPRISINGLY AFFORDABLE COMPUTER THAT MAKES THE COMPETITION LOOK TWICE !!**

## STANDARD FEATURES

- 13" COLOR CRT
- SPECIAL GRAPHICS PKG.
- EXTENDED DISK BASIC
- MINI DISK DRIVE
- 8K RAM MEMORY
- 72 KEY KEYBOARD

## PRICE & OPTIONS

| | |
|---|---|
| MODEL 3 — 8K USER RAM | 1495.00 |
| MODEL 4 — 16K USER RAM | 1695.00 |
| MODEL 5 — 32K USER RAM | 1995.00 |
| 2nd DISK DRIVE | 400.00 |
| EXPANDED KEYBOARD | 135.00 |
| FORMATTED DISKETTE | 5.00 |

MANY PROGRAMS AVAILABLE ★

## TO ORDER

SEND CHECK OR MONEY ORDER
CALIF. RES. add 6% TAX
shipping 1% all orders
CATALOG — 50¢

**HOLLYWOOD SYSTEMS**

9100 SUNSET BLVD.
SUITE 112
L.A. CALIF. 90069

---

# BECKIAN ENTERPRISES

*All Prime Quality — New Parts Only*
**Satisfaction Guaranteed**

**EDGE CARD CONNECTORS: GOLD PLATED.** (Not Gold Flash)
BODY: Non brittle, Solvent res., G.E. Valox.
CONTACTS: Bifurcated; Phos/Bronze: Gold over Nickel.
ABBREVIATIONS: S/T Solder Tail; S/E Sold. Eyelet: W/W Wire Wrap 3; SW/W Short W/Wrap;

| PART # | Description | Row Sp. | 1-4 | 5-9 | 10-24 |
|---|---|---|---|---|---|
| 5010 | 50/100 S/T ALTAIR | .140 | 3.75 | 3.50 | 3.30 |
| 5020 | 50/100 S/T IMSAI | .250 | 3.95 | 3.75 | 3.50 |
| 5030 | 50/100 W/W IMSAI | .250 | 4.10 | 3.90 | 3.70 |
| 5040 | 50/100 S/E ALT/IMSAI | .140 | 5.00 | 4.50 | 4.25 |
| 5050 | 50/100 S/T CROMEMCO | .250 | 6.25 | 6.00 | 5.75 |
| 1450 | IMSAI CARD GUIDES | | 0.16 | 0.14 | 0.12 |
| **.100" Contact Center Connectors.** | | | | | |
| 1020 | 13/26 S/E Imsai MIO: | .140 | 2.10 | 1.85 | 1.75 |
| 1040 | 25/50 S/E | .140 | 2.95 | 2.75 | 2.50 |
| 1050 | 25/50 S/T | .140 | 3.00 | 2.80 | 2.60 |
| 1060 | 36/72 W/W Vector. | .200 | 4.80 | 4.60 | 4.30 |
| 1065 | 36/72 S/T Vector. | .200 | 4.00 | 3.75 | 3.50 |
| 1070 | 40/80 S/E PET | .140 | 4.80 | 4.50 | 4.30 |
| 1075 | 40/80 W/W PET | .200 | 5.00 | 4.65 | 4.35 |
| 1080 | 40/80 S/T PET | .140 | 4.90 | 4.60 | 4.25 |
| 1085 | 43/85 S/E Cos.ELF | .140 | 5.00 | 4.75 | 4.50 |
| 1090 | 43/86 S/T Cos.ELF | .140 | 5.10 | 4.85 | 4.60 |
| 1093 | 43/86 S/T Cos.ELF | .200 | 4.95 | 4.70 | 4.45 |
| 1095 | 43/86 W/W Cos.ELF | .200 | 5.50 | 5.20 | 4.90 |
| POLARIZING KEYS: For Above | | | 0.10 | 0.10 | 0.10 |
| **.156" Contact Center Connectors.** | | | | | |
| 1550 | 6/- S/E PET,Etc | .140 | 1.30 | 1.10 | 0.90 |
| 1560 | 6/12 S/T PET:NSC. | .140 | 1.35 | 1.15 | 0.95 |
| 1575 | 12/24 S/E PET | .140 | 2.15 | 1.95 | 1.75 |
| 1580 | 12/24 S/T PET | .140 | 2.10 | 1.90 | 1.70 |
| 1590 | 15/30 S/E GRI Keybd. | .140 | 2.25 | 2.05 | 1.85 |
| 1620 | 18/36 S/E | .140 | 2.40 | 2.20 | 2.00 |
| 1650 | 22/44 S/E KIM,VECTOR | .140 | 2.20 | 2.00 | 1.80 |
| 1660 | 22/44 S/T KIM,VECTOR | .140 | 2.00 | 1.80 | 1.70 |
| 1670 | 22/44 W/W KIM,VECTOR | .200 | 2.40 | 2.20 | 2.00 |
| 1690 | 36/72 W/W | .200 | 3.90 | 3.75 | 3.50 |
| 1710 | 36/72 S/E | .140 | 3.50 | 3.30 | 3.10 |
| 1720 | 36/72 S/T | .200 | 3.30 | 3.10 | 2.90 |
| 1730 | 43/86 S/T Mot. 6800 | .140 | 4.40 | 4.15 | 3.90 |
| 1740 | 43/86 S/T Mot. 6800 | .200 | 4.35 | 4.10 | 3.85 |
| 1750 | 43/86 W/W Mot. 6800 | .200 | 4.45 | 4.25 | 4.10 |
| POLARIZING KEYS: For Above | | | 0.10 | 0.10 | 0.10 |

**RS232 & 'D' TYPE SUBMINIATURE CONNECTORS:**

| QUANTITY | | 1-4 | 5-9 | 10-24 |
|---|---|---|---|---|
| DE9P Male | | 1.45 | 1.35 | 1.25 |
| DE9S Female | | 1.93 | 1.80 | 1.70 |
| DE110963-1 | 2pc. Grey Hood | 1.20 | 1.10 | 1.00 |
| DA15P Male | | 1.95 | 1.80 | 1.70 |
| DA15S Female | | 2.80 | 2.60 | 2.40 |
| DA51211-1 | 1pc. Grey Hood | 1.25 | 1.15 | 1.10 |
| DA110963-2 | 2pc. Grey Hood | 1.22 | 1.10 | 1.05 |
| DB25P Male | | 2.20 | 2.10 | 1.90 |
| DB25S Female | | 3.20 | 3.00 | 2.70 |
| DB51212-1 | 1pc. Grey Hood | 1.30 | 1.20 | 1.10 |
| DB51226-1A | 2pc. Black Hood | 1.40 | 1.30 | 1.20 |
| DB110963-3 | 2pc. Grey Hood | 1.35 | 1.25 | 1.15 |
| DC37P Male | | 3.70 | 3.50 | 3.35 |
| DC37S Female | | 4.90 | 4.70 | 4.40 |
| DC110963-4 | 2pc. Grey Hood | 1.95 | 1.85 | 1.75 |
| DD50P Male | | 4.40 | 4.30 | 4.10 |
| DD50S Female | | 4.90 | 4.70 | 4.50 |
| DD51216-1 | 1pc. Grey Hood | 2.30 | 2.10 | 1.90 |
| DD110963-5 | 2pc. Grey Hood. | 2.40 | 2.20 | 2.00 |
| D20418-2 Hardware Sets | | 0.75 | 0.70 | 0.65 |

**I.C. SOCKETS. GOLD. WIRE WRAP 3 TURN.**
14 pin $0.36 ea.
16 pin 0.38 ea.

**I.C. SOCKETS. Dip Solder. Tin.**
14 pin $0.15 ea.
16 pin 0.17 ea.

**8080 PRIME**
$8.00 ea.

**2708 EPROMS PRIME**
$14.00 ea.

**CONNECTORS FOR CENTRONICS 700 SERIES:**
**Amhpenol 57-30360 For Back of Centronics '700' Series:**
Price: $9.00ea. 5 pcs. $7.50ea.

**WHISPER FANS:** Excellent for Computer cabinet cooling. Extremely quiet.
Dim. 4-3/4" × 1-1/2" thick. U.L. Listed.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $22.00 | $19.00 | $18.00 |

*WRITE FOR LARGER QUANTITY DISCOUNTS. DEALER INQUIRIES ARE WELCOME.*

*WE ARE CONNECTOR (EDGE CARD) SPECIALISTS. IF YOU DO NOT SEE WHAT YOU NEED IN THIS ADVERTISEMENT, PLEASE WRITE US. WE WILL REPLY.*

*TERMS: Minimum Order $10.00: Add $1.25 for handling and shipping. All orders over $25.00 in USA and Canada: WE PAY THE SHIPPING.*
*NOTE: CA residents please add 6% sales tax.*
*NO C.O.D. SHIPMENTS OR ORDERS ACCEPTED.*

*MAIL ORDERS TO:* ***Beckian Enterprises***
***P.O. Box 3089***
***Simi Valley, CA 93063***

# wirewrap

## PRECUT WIRE

### #30 WIRE KITS

| #1 $7.95 | | | | #2 $19.95 | | | |
|---|---|---|---|---|---|---|---|
| 250 | 3″ | 100 | 4½″ | 250 | 2½″ | 250 | 5″ |
| 250 | 3½″ | 100 | 5″ | 500 | 3″ | 100 | 5½″ |
| 100 | 4″ | 100 | 6″ | 500 | 3½″ | 250 | 6″ |
| | | | | 500 | 4″ | 100 | 6½″ |
| | | | | 250 | 4½″ | 100 | 7″ |

| #3 $24.95 | | | | #4 $44.95 | | | |
|---|---|---|---|---|---|---|---|
| 500 | 2½″ | 500 | 4½″ | 1000 | 2½″ | 1000 | 4½″ |
| 500 | 3″ | 500 | 5″ | 1000 | 3″ | 1000 | 5″ |
| 500 | 3½″ | 500 | 5½″ | 1000 | 3½″ | 1000 | 5½″ |
| 500 | 4″ | 500 | 6″ | 1000 | 4″ | 1000 | 6″ |

Choose One Color or Random Assortment:
Red, Blue, Green, Yellow, White, Orange, Black.

* #26 Prices on Request

#30 Kynar stripped 1″ on each end. Lengths are overall.
Colors: Red, Blue, Green, Yellow, Black, Orange, White.
Wire packaged in plastic bags. Add 25¢/length for tubes.

| In. | 100 | 500 | 1000 | 5000 |
|---|---|---|---|---|
| 2½ | 1.04 | 2.98 | 5.16/K | 4.67/K |
| 3 | 1.08 | 3.22 | 5.65/K | 5.06/K |
| 3½ | 1.13 | 3.46 | 6.14/K | 5.46/K |
| 4 | 1.18 | 3.20 | 6.62/K | 5.87/K |
| 4½ | 1.23 | 3.95 | 7.12/K | 6.25/K |
| 5 | 1.28 | 4.20 | 7.61/K | 6.62/K |
| 5½ | 1.32 | 4.48 | 8.10/K | 7.03/K |
| 6 | 1.37 | 4.72 | 8.59/K | 7.43/K |
| 6½ | 1.60 | 5.37 | 9.84/K | 8.48/K |
| 7 | 1.66 | 5.63 | 10.37/K | 8.91/K |
| 7½ | 1.73 | 5.89 | 10.91/K | 9.33/K |
| 8 | 1.78 | 6.15 | 11.44/K | 9.79/K |
| 8½ | 1.82 | 6.41 | 11.97/K | 10.19/K |
| 9 | 1.87 | 6.76 | 12.51/K | 10.62/K |
| 9½ | 1.92 | 6.93 | 13.04/K | 11.05/K |
| 10 | 1.99 | 7.26 | 13.57/K | 11.48/K |
| Addl. In. | .15 | .60 | 1.20/K | 1.00/K |

### EDGE CARD CONNECTORS

| | | | |
|---|---|---|---|
| 44 pin Solder Tail | $1.95 | $17.50/10 | ALL are Gold |
| 100 pin Solder Tail | $3.95 | $35.00/10 | 100 pin are IMSAI |
| 100 pin Wire Wrap | $3.95 | $35.00/10 | spacing. |

## WIRE WRAP TOOLS

### HOBBY WIRE WRAP TOOL


| | |
|---|---|
| **BW 630 (Back Force)** | **$34.95** |
| BT 30 Extra Bit | 2.95 |
| BT 2628 #26 Bit | 7.95 |
| Batteries & Charger | 11.00 |

### INDUSTRIAL WIRE WRAP TOOL


| | |
|---|---|
| **BW 928** | **$49.95*** |
| BW 928BF (Back Force) | 52.95* |
| #30 Bit & Sleeve | 29.50 |
| #26 Bit & Sleeve | 29.50 |
| Batteries & Charger | 11.00 |

### ELECTRICAL INDUSTRIAL WIRE WRAP TOOL


| | |
|---|---|
| **EW 7D** | **$85.00*** |
| EW 7D BF (Back Force) | 92.90* |
| #30 Bit & Sleeve | 29.50 |
| #26 Bit & Sleeve | 29.50 |

*Industrial Tools do not include Bit & Sleeve
Spring Loaded bit on Back Force models.

### INTERCONNECT CABLES

Ribbon cable connectors for connecting boards to front panels, or board to board.

| | SINGLE ENDED | | | DOUBLE ENDED | | |
|---|---|---|---|---|---|---|
| | 14 pin | 16 pin | 24 pin | 14 pin | 16 pin | 24 pin |
| 6″ | 1.24 | 1.34 | 2.05 | 2.24 | 2.45 | 3.37 |
| 12″ | 1.33 | 1.44 | 2.24 | 2.33 | 2.55 | 3.92 |
| 24″ | 1.52 | 1.65 | 2.63 | 2.52 | 2.76 | 4.31 |
| 48″ | 1.91 | 2.06 | 3.40 | 2.91 | 3.17 | 5.08 |

### OK PRODUCTS

| | | |
|---|---|---|
| WD 30 | 50 ft. Wire Dispenser Red, White, Blue, or Yellow | $3.75 |
| WD-30-TRI | TRI Color Dispenser | 5.50 |
| R-30-TRI | Refill for TRI Color | 3.75 |
| INS 1416 | 14 & 16 pin Insertion Tool | 3.25 |
| MOS 40 | 40 pin Insertion Tool | 7.50 |
| EX-1 | IC Extractor Tool | 1.49 |
| H-PCB-1 | Hobby PC Board | 4.99 |
| WSU 30 | Hand Wrap/Unwrap/Strip Tool | 6.25 |
| WSU 30M | Same as WSU30 with Modified Wrap | 7.50 |

### SOLDERLESS BREADBOARDS

**SK 10 $16.50**

2.2″ x 6.5″


**MB10 $18.95**

**SK10 mounted on board W74 Binding Posts & Rubber Feet**

### WIRE WRAP SOCKETS

| | 1-9 | 10-24 | 25-99 | 100-249 | 250-999 |
|---|---|---|---|---|---|
| 8 pin* | .40 | .36 | .34 | .31 | .27 |
| 14 pin* | — | .39 | .37 | .34 | .32 |
| 16 pin* | — | .42 | .40 | .36 | .34 |
| 18 pin* | .70 | .60 | .55 | .50 | .45 |
| 20 pin | .90 | .80 | .75 | .65 | .62 |
| 22 pin* | .95 | .85 | .80 | .70 | .65 |
| 24 pin | .95 | .85 | .80 | .70 | .65 |
| 25 pin strip | 1.25 | 1.15 | 1.00 | .95 | .90 |
| 28 pin | 1.25 | 1.15 | 1.00 | .95 | .90 |
| 40 pin | 1.65 | 1.45 | 1.35 | 1.20 | 1.10 |

**Gold 3-Level Closed Entry Design** *End & Side Stackable
2-Level Sockets Available

### LOGIC PROBE

**PRB-1**

**$34.95**


- Compatible with all Logic Families
- 10 Nsec pulse response

## PAGE DIGITAL ELECTRONICS

135 E. Chestnut St. #5 Monrovia, CA 91016 (213) 357-5005

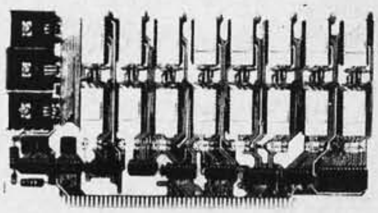
**Ordering Information:**
- Orders under $25 and COD's add $2
- All others, shipped Ppd in U.S. via UPS
- For Blue Label (Air) or 1st Class, add $1
- We accept Visa & Mastercharge

Catalog available on request.

Dealer Inquiries Invited

Circle 298 on inquiry card.

## 16K EPROM CARD-S 100 BUSS



**$59.95 KIT**

**OUR BEST SELLING KIT!**

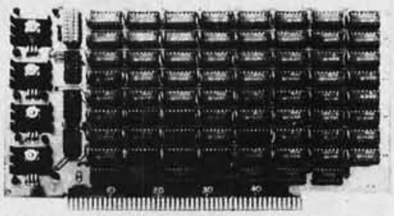
**USES 2708's!**

Thousands of personal and business systems around the world use this board with complete satisfaction. Puts 16K of software on line at **ALL TIMES!** Kit features a top quality soldermasked and silk-screened PC board and first run parts and sockets. All parts (except 2708's) are included. Any number of EPROM locations may be disabled to avoid any memory conflicts. Fully buffered and has WAIT STATE capabilities.

**OUR 450NS 2708'S ARE $8.95 EA. WITH PURCHASE OF KIT**

**ASSEMBLED AND FULLY TESTED ADD $25**

## 8K LOW POWER RAM KIT-S 100 BUSS
## 250 NS SALE!



**ADD $5 FOR 250NS!**

**$129 KIT**

**(450 NS RAMS!)**

Thousands of computer systems rely on this rugged, work horse, RAM board. Designed for error-free, NO HASSLE, systems use.

**KIT FEATURES:**

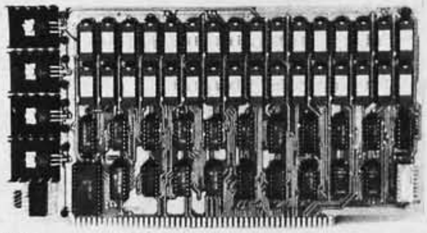
1. Doubled sided PC Board with solder mask and silk screen layout. Gold plated contact fingers.
2. All sockets included.
3. Fully buffered on all address and data lines.
4. Phantom is jumper selectable to pin 67.
5. FOUR 7805 regulators are provided on card.

Blank PC Board w/Documentation **$29.95**
Low Profile Socket Set...**13.50**
Support IC's (TTL & Regulators) **$9.75**
Bypass CAP's (Disc & Tantalums) **$4.50**

**ASSEMBLED AND FULLY BURNED IN ADD $30**

## 16K STATIC RAM KIT-S 100 BUSS

**$295 KIT**



**FULLY STATIC, AT DYNAMIC PRICES**

**WHY THE 2114 RAM CHIP?**

We feel the 2114 will be the next industry standard RAM chip (like the 2102 was). This means price, availability, and quality will all be good! Next, the 2114 is FULLY STATIC! We feel this is the **ONLY** way to go on the S-100 Buss! We've all heard the HORROR stories about some Dynamic Ram Boards having trouble with DMA and FLOPPY DISC DRIVES. Who needs these kinds of problems? And finally, even among other 4K Static RAM's the 2114 stands out! Not all 4K static Rams are created equal! Some of the other 4K's have **clocked** chip enable lines and various timing windows **just as critical** as Dynamic RAM's. Some of our competitor's 16K boards use these "tricky" devices. But not us! The 2114 is the ONLY logical choice for a trouble-free, straightforward design.

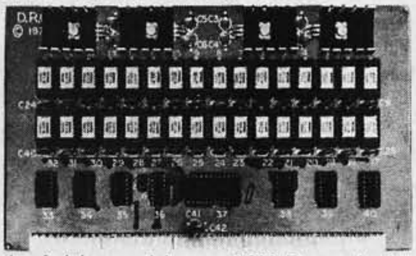
KIT FEATURES:

1. Addressable as four separate 4K Blocks.
2. ON BOARD BANK SELECT circuitry. (Cromemco Standard!). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams.
4. ON BOARD SELECTABLE WAIT STATES.
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers.
6. All address and data lines fully buffered.
7. Kit includes ALL parts and sockets.
8. PHANTOM is jumpered to PIN 67.
9. LOW POWER: under 2 amps TYPICAL from the +8 Volt Buss.
10. Blank PC Board can be populated as any multiple of 4K.

BLANK PC BOARD W/DATA—**$33**
LOW PROFILE SOCKET SET—**$12**
ASSEMBLED & TESTED—**ADD $30**
SUPPORT IC'S & CAPS—**$19.95**
2114 RAM'S—**8 FOR $69.95**

## 16K STATIC RAM SS-50 BUSS

**$295 KIT**



**FULLY STATIC AT DYNAMIC PRICES**

**FOR SWTPC 6800 BUSS!**

**KIT FEATURES:**

1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Runs at Full Speed
4. Double sided PC Board. Solder mask and silk screened layout. Gold fingers.
5. All Parts and Sockets included
6. Low Power: Under 2 Amps Typical

BLANK PC BOARD—**$33**
COMPLETE SOCKET SET—**$12**
SUPPORT IC'S AND CAPS—**$19.95**

## TM990 BUSS PROTOTYPE & WIREWRAP BOARD

For use with the Texas Instrument Series of 16 Bit Microcomputer Modules. Fully buss compatible. An inexpensive and quick way to expand the capacity of your TI computer. Made of G-10 Epoxy PC material. Gold plated contact fingers all plated through holes. High density, up to over 100 DIP's. Fully documented. **$70 each** **(OEM Discounts Available)**

## 450 NS! 2708 EPROMS

Now full speed! Prime new units from a major U.S. Mfg. 450 N.S. Access time. 1K x 8. Equiv. to 4-1702 A's in one package.

~~$15.75 ea.~~ **$9.95** ~~4 For $50.00~~

**PRICE CUT**

**ANNOUNCEMENT:**

To better serve our customers we are splitting Digital Research Corp of Texas into two operating sections. Parts and Computers. We feel this change will allow us to offer you lower prices, better service, and many more new products. Continue to order parts, clock modules, etc. from D.R.P. P.O. Box 401247 Garland, TX 75040. To order computer parts and computer kits order from Digital Research: Computers P.O. Box 401565 Garland, TX 75040.

## Z-80 PROGRAMMING MANUAL

By MOSTEK, or ZILOG. The most detailed explanation ever on the working of the Z-80 CPU CHIPS. At least one full page on each of the 158 Z-80 instructions. A MUST reference manual for any user of the Z-80. 300 pages. Just off the press. **$12.95**

**NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.**

## 16K DYNAMIC RAM CHIP

16K X 1 Bits. 16 Pin Package. Same as Mostek 4116-4. 250 NS access. 410 NS cycle time. Our best price yet for this state of the art RAM. 32K and 64K RAM boards using this chip are readily available. These are new, fully guaranteed devices by a major mfg. **VERY LIMITED STOCK!**

**8 FOR $79.50**

# Digital Research: Computers
**(OF TEXAS)**

**P.O. Box 401565 • GARLAND, TEXAS 75040 • (214) 271-2461**

**TERMS:** Add 50¢ postage, we pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa, MasterCharge, and American Express cards. Tex. Res. add 5% Tax. Foreign orders (except Canada add 20% P & H. 90 Day Money Back Guarantee on all items.

Circle 100 on inquiry card.

Circle 151 on inquiry card.

# SO THEN BILL GODBOUT SAYS TO US,

"I want to do a summer sale ad". But, we replied, the ad deadline is tomorrow and we don't have time for typesetting or photos or anything... so he says, "look, with these prices they won't mind. Just freehand it or something". OK, Bill, here it is...

# 16K MEMORY EXPANSION SET $87.20!!

For Radio Shack-80, Apple, Exidy Sorcerer

That's a lot of memory expansion for $87.20! Our set includes 250 ns low power chips, dip shunts, and full documentation that makes conversion and memory expansion a snap. At this price, there is no fancy packaging; but we've got real products, at real prices, with all the performance you or your computer will ever need.

# ECONORAM™ II CLOSEOUT!

## $129 unkit (3/$375) FULL 8K X 8! STATIC TOO!

No kidding - $129! Our brand-new Econoram IIA is out, but we still have some Econoram II unkits around and want to clear them out. Even by today's standards, the old Econoram II is an excellent memory. It offers full 2 MHz operation, low power consumption, high quality electronic + mechanical components, configuration as 2 independent 4K blocks, and one of the best track records in the industry for reliable and cost-effective operation. Our unkit has all sockets and bypass caps presoldered in place for easy, one-evening assembly. 1 year limited warranty on all components. Better send in soon if you're interested -- this is a limited qty. item! P.S. ASSM. $155

TERMS OF SALE: Both items while-they-last. For VISA® or Mastercharge® orders call our 24 hr. order desk at (415) 562-0636. California residents add sales tax. Add 5% for shipping; excess refunded. COD OK with street address for UPS delivery. For more info, see our full-page, somewhat more formal ad elsewhere in this magazine.

GODBOUT

Bill Godbout Electronics, Building 725, Oakland Airport, CA 94614

---

# FREE! UP TO $170 IN MERCHANDISE

**WITH THE PURCHASE OF ONE OF THE FOLLOWING PET-CBM ITEMS!!**

**ASK ABOUT EDUCATIONAL DISCOUNTS ON PET**

## PET SPECIALS

| | | LIST | FREE* |
|---|---|---|---|
| PET 16N | 16K full size graphics keyboard | $ 995 | $130 |
| PET 16B | 16K full size business keyboard | $ 995 | $130 |
| PET 32N | 32K full size graphics keyboard | $1295 | $170 |
| PET 32B | 32K full size business keyboard | $1295 | $170 |
| PET 16S | 16K small keyboard, integral cassette | $ 995 | $130 |
| PET 32S | 32K small keyboard, integral cassette | $1295 | $170 |
| PET 8K | 8K small keyboard, integral cassette | $ 795 | $100 |
| PET 2040 | Dual Disk Drive — 343,000 bytes | $1295 | $170 |
| PET 2040A | Single Disk Drive — 171,000 bytes | $ 895 | $115 |
| PET 2022 | Tractor Feed Printer | $ 995 | $130 |
| PET 2023 | Pressure Feed Printer | $ 849 | $110 |
| PET C2N | External Cassette Deck | $ 95 | $ 12 |

| | |
|---|---|
| IEEE - RS232 Printer Adaptor for PET | $ 79.50 |
| BETSI PET to S-100 Interface & Motherboard | $119.00 |
| PET Connectors- Parallel or IEEE | $ 2.25 |
| - Cassette Port | $ 1.60 |
| Personal Information Management System— Add $3 for PET program cassette | $ 8.90 |
| Protect-A-Pet dust cover | $ 9.50 |
| EXS 100 Floppy Disk Controller For PET | $299.00 |
| MICROCHESS for PET *(Peter Jennings)* | $ 17.90 |
| PET 4 Voice Music Board (MTUK-1002-2) | $ 49.00 |
| Music Software (K-1002-3C) for PET | $ 19.00 |
| CmC Word Processor program for PET | $ 25.00 |
| Bridge Challenger program for PET — Play and reply bridge hands against the PET | $ 13.50 |
| Graphics Utility Package for PET | $ 13.50 |
| Stimulating Simulations-Book & PET tape | $ 13.50 |
| Kite Flight - 2 player action game | $ 7.95 |

Write for PET Software List

| | |
|---|---|
| Auto-Repeat Hardware for PET | $ 24.50 |

Word Processor for PET — Machine Language version. Auto scroll, insert, delete, form letter append, etc.
8K Version $ 24.00 16K or 32K with disk $ 95.00

**WRITE FOR 6502 AND S-100 PRODUCT LIST**

# FREE!

**UP TO $170 IN MERCHANDISE**

**Minimum Order $10.00**

| | |
|---|---|
| KIM-1 $159 (Add $30 for Power Supply) SYM-1 | $229 |
| BAS-1 Microsoft ROM Basic for SYM | $139 |
| Memory Plus | $199 |
| SEA-16 New 16K Static RAM | $325 |
| Seawell Motherboard-4K RAM space | $ 99 |
| KTM-2 Synertek Keyboard and Video Interface with Graphics Capability | $290 |
| RAM 16 4MHz 16K Static S-100 RAM | $309 |
| 2114 L 450ns 4K Static RAM | $6.95 |
| 2716 EPROM (5 Volt) | $ 38 |
| 6550 RAM (for PET 8K) | $16.20 |
| 6502 Microprocessor Chip | $ 9.95 |
| 6522 VIA | $9.75 |
| 6502 PIA | $10.50 |

**BOOKS**

| | |
|---|---|
| Programming the 6502 *(Zaks)* | $ 9.90 |
| 6502 Applications Book *(Zaks)* | $11.90 |
| 6500 Programming Manual *(MOS)* | $ 6.50 |
| Programming a Microcomputer:6502 | $ 8.90 |
| Basic for Home Computers | $ 5.90 |

**SALE**

| | |
|---|---|
| 3M "Scotch" 8" disks | 10/$31 |
| 3M "Scotch" 5" diskettes | 10/$35 |
| Verbatim 5" diskettes | 10/$28 |

**(Write for quantity prices)**

Cassettes *(all tapes guaranteed)*
Premium quality, high output lownoise in 5 screw housing with labels:
C-10 10/5.95 50/25.00 100/48.00
C-30 10/7.00 50/30.00 100/57.00

*** Amount of Free Merchandise with Purchase of PET-CBM Item.**

# A B Computers

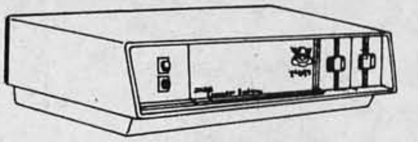
**115-A E. Stump Road
Montgomeryville, PA 18936
(215) 699-8386**



Circle 1 on inquiry card.

# JADE Computer Products

## THE PIGGY MAINFRAME



This sleak new mainframe is neatly trimmed to hold six S-100 boards, two mini-floppy drives, and is available in five colors. Power requirements: 115/220 VAC, 50/60 HZ. Weight: 27 lbs. (with drives). Dimensions: 21.375" Wide X 8.4" High X 15.875" Deep. Power Supply: 8 volts at 18 Amps unregulated. 16 volts at 3 amps unregulated. - 16 volts at 3 amps unregulated. 5 volts at 3 amps regulated. 12 volts at 3 amps regulated.

THE PIGGY IS HERE!

THE PIGGY (Without Drives) .... $475.00

## JADE ISO—BUS MOTHERBOARDS

Comes in either 6, 12, or 18 slot sizes. These boards with a special ground plane assures a silent operation

| | |
|---|---|
| **JADE 6 Slot** | |
| Kit | $49.95 |
| Assembled | $59.95 |
| Bare Board | $24.95 |
| **JADE 12 Slot** | |
| Kit | $89.95 |
| Assembled | $99.95 |
| Bare Board | $39.95 |
| **JADE 18 Slot** | |
| Kit | $129.95 |
| Assembled | $149.95 |
| Bare Board | $59.95 |

## SD SYSTEMS Z-80 STARTER KIT

Based on the powerful Z80 CPU, this kit is an ideal introduction to microprocessors. It has an on-board keyboard and display, plus cassette tape interface and expansion provisions for two S-100 connectors. This "Do-it-all" Board will also program the 2716 2K EPROM.

Kit ............... $249.95
Assmbld and Tstd $399.95

## PROTO BOARD

Includes gold plated fingers, S-100 size, holds 72-16 pin dips, accomodates all 8 thru 40 pin dip packages.
Reg.-$19.95 Special Price- $16.95

TRS-80·APPLE·SORCERER·TRS-80·APPLE·SORCERER·TRS-80·APPLE·SORCERER·TRS-80·APPLE·SORCERER·TRS-80·APPLE

## JADE MEMORY EXPANSION KITS For TRS—80, Apple, & Exidy

4116's

Everything a person needs to add 16K of memory. Chips come neatly packaged with easy to follow directions. In minutes your machine is ready for games and more advanced software.

**$82.00**

## JADE VIDEO INTERFACE

S-100 Compatible Serial Interface with Sockets Included.

| | |
|---|---|
| Kit | $117.95 |
| Assembled & Tested | $159.95 |
| Bare Board w/manual | $ 35.00 |

## THE SYM-1 $245

6502 based single board computer with keyboard/display, KIM-1 hardware compatible, complete documentation.

SYM—1 CASE $39.95

## JADE PARALLEL/SERIAL INTERFACE

S-100 compatible, 2 serial I/O ports, 1 parallel I/O.

| | |
|---|---|
| Kit JG-P/S | $124.95 |
| Assembled & Tested JG-P/SA | $179.95 |
| Bare Board W/Manual | $ 30.00 |

## Solid State Music's I/O 4

2- Serial & 2- Parallel I/O Ports. S-100 with full handshaking.

| | |
|---|---|
| KIT | $149.95 |
| Assembled | $199.95 |
| Bare Board | $29.95 |

## MEM-2
16 K Static RAM Board

| | |
|---|---|
| Kit- (450ns) | $250.00 |
| Kit- (250ns) | $285.00 |
| Assembled- (450ns) | $325.00 |
| Assembled- (250ns) | $350.00 |

## DYNAMIC RAM BOARDS EXPANDABLE TO 64K

### 32K VERSION • KITS

Uses 4115 (8K x 1, 250ns) Dynamic RAM's, can be expanded in 8K increments up to 32K

| | | | |
|---|---|---|---|
| 8K | $159.00 | 24K | $249.00 |
| 16K | $199.00 | 32K | $299.00 |

**4115 SALE 8 for $39.95**

### 64K VERSION • KITS

Uses 4116 (16K x 1, 200ns) Dynamic RAM's, can be expanded in 16K increments up to 64K:

| | | | |
|---|---|---|---|
| 16K | $249.00 | 48K | $469.00 |
| 32K | $369.00 | 64K | $569.00 |

## Summer Madness Sale
FREE!
A 12 INCH, BLACK AND WHITE, MONITOR WITH THE PURCHASE OF EVERY SORCERER!

## The Exidy SORCERER

Flexibilty is the key. The Sorcerer Computer gives you the flexibility of using ready-to-run, pre-packaged programs or doing your own thing and personalizing the programs for yourself. Which ever you choose, the Sorcerer is the personal computer that speaks your language.

The Sorcerer also provides full graphics capabilities. Each character, formed by an 8 x 8 dot cell, can be programmed as a graphic symbol set. High resolution (512 x 240 addressable points) gives a total of 122,880 locations for super animation and extremely tight plotting curves. The alphanumeric set gives 64 x 30 characters on the video screen

With 16K of memory .................................... $1150.00

## ★ STATIC RAM SPECIALS ★

2114's, low power (1024x4)

| | 1-15 | 16-99 | 100 + |
|---|---|---|---|
| 450ns | 8.00 | 6.95 | 5.50 |
| 300ns | 9.00 | 8.00 | 6.50 |
| **TMS4044/MM5257**, low power | | | |
| 450ns | 8.00 | 7.50 | 6.50 |
| 300ns | 9.95 | 8.75 | 8.00 |
| **4200A** (4K x 1, 200ns) | 9.95 | 8.50 | 8.00 |
| **410D** (4K x 1, 200 ns) | 8.25 | 7.00 | 6.75 |

## STATIC RAM BOARDS

JADE 8K

| | | |
|---|---|---|
| Kits: | 450ns | $125.95 |
| | 250ns | $149.75 |
| Assembled & Tested: | 450ns | $139.75 |
| | 250ns | $169.75 |
| Bare Board: | | $ 25.00 |

16K — Uses 2114's (low power)

| | |
|---|---|
| Assembled & Tested: | |
| RAM 16 (250ns) | $375.00 |
| RAM 16B (450ns) | $325.00 |

16K with memory management

| | |
|---|---|
| Assembled & Tested: | |
| RAM 65 (250ns) | $390.00 |
| RAM 65B (450ns) | $350.00 |

32K Static

| | |
|---|---|
| Assembled & Tested: | |
| 250ns | $795.00 |
| 450ns | $725.00 |
| 250ns Kit | $575.00 |

## SD SYSTEMS SBC-100

An S-100 single board computer. Z-80 CPU with 1024 bytes of RAM. 8 to 32K bytes of PROM. Serial I/O port.

| | |
|---|---|
| Kit | $239.95 |
| Assembled | $369.95 |

## TARBELL Cassette Interface

Plugs into your IMSAI or ALTAIR. 4 extra status lines. 37 page manual included. 4 extra control lines

| | |
|---|---|
| Kit | $99.95 |
| Assembled | $175.00 |
| Bare Board | $40.00 |
| Manual | $8.00 |

## POWER SUPPLIES

PSD-249A: For a Single 5 1/4" Disk Drive. By Power-One or Alpha Power: 5V at .7A. ............ $42.00

PSD-205A: For Single 8" Disk Drive. By Power-One. 5V at 1A, -5V at .5A, 24V at 1.5A ............ $89.95

PSD-206A: For Two 8" Disk Drives. By Power-One or Alpha Power. 5V at 2.5A, -5V at .5A, -24V at 3A ............ $125.00

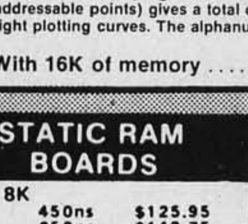
Rockwell Aim-65 Power Supply PSX-030A ............ $59.95

KIM-1 or SYM-1 Power Supply: PSX-020A ............ $59.95

## Rockwell AIM-65: The Head-Start in Microcomputers

A KIM-1 compatible machine with on-board printer and a real keyboard!

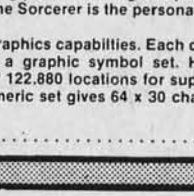
$375.00 w/1K RAM
$450.00 w/4K RAM



| | |
|---|---|
| 4K assembler/editor in ROM: | $ 80.00 |
| 8K BASIC in ROM: | $100.00 |
| Power supply: | $ 59.95 |
| Case for AIM-65: | $ 49.95 |

Special Package Price: $599.00
AIM-65 (4K), Power Supply, Case, and 8K BASIC ROM

## DIP SWITCHES



| Part No | positions | 1-9 |
|---|---|---|
| SWD 103 | 3 | $1.18 |
| SWD 104 | 4 | $1.20 |
| SWD 105 | 5 | $1.24 |
| SWD 106 | 6 | $1.28 |
| SWD 107 | 7 | $1.30 |
| SWD 108 | 8 | $1.34 |
| SWD 109 | 9 | $1.36 |
| SWD 110 | 10 | $1.38 |

## TEXTOOL ZIP* DIP® II Sockets

*ZERO INSERTION FORCE sockets

PRICES:

| | |
|---|---|
| 16 pin Zip Dip II | $6.50 |
| 24 pin Zip Dip II | $7.50 |
| 40 pin Zip Dip II | $10.25 |

## CHIPS

| MICROPROCESSORS | |
|---|---|
| F8 | $16.95 |
| Z80 (2MHz) | $10.95 |
| Z80A (4MHz) | $14.95 |
| CDP1802CD | $19.95 |
| 6502 | $11.95 |
| 6800 | $9.75 |
| 6802 | $14.00 |
| 8008-1 | $15.95 |
| 8035 | $24.00 |
| 8035-8 | $24.00 |
| 8080-A | $10.00 |
| 8085 | $23.00 |
| TMS9900TL | $49.95 |
| **8080A SUPPORT DEVICES** | |
| 8212 | $2.90 |
| 8214 | $4.65 |
| 8216 | $2.75 |
| 8224 (2MHz) | $4.30 |
| 8224-4 (4MHz) | $9.95 |
| 8226 | $2.75 |
| 8228 | $6.40 |
| 8238 | $6.40 |
| 8243 | $8.00 |
| 8251 | $7.50 |
| 8253 | $20.00 |
| 8255 | $6.40 |
| 8257 | $18.00 |
| 8259 | $18.00 |
| 8275 | $51.20 |
| 8279 | $17.70 |
| **USRT** | |
| S2350 | $10.95 |
| **UARTS** | |
| AY5-1013A | $5.25 |
| AY5-1014A | $8.25 |
| TR1602B | $5.25 |
| TMS6011 | $5.95 |
| IM6403 | $9.00 |
| **BAUD RATE GENERATORS** | |
| MC14411 | $10.00 |
| 14411 Crystal | $4.95 |
| **6800 PRODUCT** | |
| 6821P | $5.25 |
| 6828P | $9.50 |
| 6834P | $16.95 |
| 6850P | $4.80 |
| 6852P | $5.25 |
| 6860P | $9.25 |
| 6862P | $12.00 |
| 6875L | $7.30 |
| 6880P | $2.50 |
| **CHARACTER GENERATORS** | |
| 2513 Upper (-12, 5) | $6.75 |
| 2513 Lower (-12, 5) | $6.75 |
| 2513 Upper (5 volt) | $9.75 |
| 2513 Lower (5 volt) | $10.95 |
| MCM6571 up scan | $10.95 |
| MCM6571A down scan | $10.95 |
| **PROMS** | |
| 1702A | $5.00 |
| 2708 | $12.95 |
| 2716 (5, 12) | $49.00 |
| 2716 (5v) | $49.00 |
| 2758 (5v) | $30.00 |
| **DYNAMIC RAMS** | |
| 416D 4116 (200ns) | $12.50 |
| 2104 4096 | $4.00 |
| 2107B-4 | $3.95 |
| TMS4027 4096 | $4.00 |

| STATIC RAMS | 1—15 | 16—100 |
|---|---|---|
| 21L02 (450ns) | $1.50 | $1.20 |
| 21L02 (250ns) | $1.75 | $1.50 |
| 2101-1 | $2.95 | $2.60 |
| 2111-1 | $3.25 | $3.00 |
| 2112-1 | $2.95 | $2.65 |

| FLOPPY DISK CONTROLLERS | |
|---|---|
| 1771B01 | $39.95 |
| 1791 | $49.95 |
| **KEYBOARD CHIPS** | |
| AY5-2376 | $13.75 |
| AY5-3600 | $13.75 |
| MM5740 | $18.00 |



Circle 195 on inquiry card.

PLACE ORDERS TOLL FREE: **800/421-5809 Continental U.S.**
**800/262-1710 Inside California**

# THE BIG Z

## THE NEW Z-80

### CPU BOARD FROM JADE

Features Include: ■ S-100 Compatible, available in 2MHz or 4MHz versions. ■ On-board 2708, 2716, 2516, or 2532 EPROM can be addressed on any 1K, 2K, or 4K boundary, with power-on jump to EPROM. ■ On-board EPROM may be used in SHADOW mode, allowing full 64K RAM to be used. ■ Automatic MWRITE generation if front panel is not used. ■ On-board USART for synchronous or asynchronous RS232 operation (on-board baud rate generator). ■ Reverse-channel capability on USART allows use with buffered peripherals or devices with "not-ready" signal.

| 2MHz- | 4 MHz- |
|---|---|
| Kit: CPU-30200K, 2 lbs ... $149.95 | Kit: CPU-30201K, 2 lbs .. $159.95 |
| Assembled and Tested: CPU-30200A, 2 lbs ... $199.95 | Assembled and Tested CPU-30201A, 2 lbs .. $209.95 |

## JADE'S DOUBLE DENSITY

DD ™

- Single or Double Density Recording
- Full Size or Mini Floppy
- CP/M Compatible in either density
- Programmed Data Transfer, no DMA
- Controls up to 8 drives
- IBM format in either density
- Software Selectable Density

KIT: $249.00
Assmb. & Tstd: $299.00

■ This controler utilizes the proven reliability of the IBM standard format as well as the lastest phase-locked-loop for data separation ■ All clocks are generated from an on-board crystal oscillator ■ Right precompensation is used to enhance data recovery reliability in the double density mode ■ Density selection is entirely transparent to the user ■ Single and double density diskettes can be mixed on the same system.

## LEEDEX MONITOR

- 12" Black and White
- 12MHZ Bandwidth
- Handsome Plastic Case

**$139.00**

## INTEGRAL DATA SYSTEMS MODEL 440

$995.00 THE PAPER TIGER PRINTER

- Up to 198 CPS
- 1.75" to 9.5" Adjustable Tractor Feed
- Parallel and Serial Interface
- 96 Character ASCII Set
- 132 columns- 6 or 8 lines/inch
- Eight software selectable Character sizes
- 110, 300, 600, 1200 Baud Rate

FOR THE GRAPHIC OPTION WITH 2K, ADD $199.00

- Two Drives Siemens/GSI 8" Floppy
- Power Supply for Above
- Jade Double Density Board (KIT)
- CP/M Operating System with Basic E
- Package of 10 Blank 8" Diskettes (Double Density)
- Includes Interface Cables

Price if Purchased Separately
$1544.95
Jade Special Package Deal
*$1225.00*

## CABLES

MINI-DISK CABLE KIT: To connect two 5 1/4" drives to disk controller board. Contains assembled and tested 5' long signal cable with 34 pin edge connectors. Also includes cables and connector for D.C. power supply.
WCA-3431K ............... $34.95

8" DISK CABLE KIT: To connect two 8" disk drives to edge-type controller (e.g., Versafloppy, Double-D). Contains assembled and tested signal cable with connectors plus cable and connectors for both A.C. and D.C. power.
WCA-5031K ............... $38.45

8" DISK CABLE KIT: Same as WCA-5031K except controller end of signal cable uses "Header" type connector, e.g., for Tarbell Controller.
WCA-5032K ............... $38.95

SIGNAL CABLE ONLY: For one 5 1/4" drive to edge type controller connector (e.g., TRS-80 to Vista Disk Drive).
WCA-3421A ............... $24.95
Same as Above, except for two 5 1/4" drives. WCA-3431A .......... $29.95

## SPECIAL

5¼" Diskettes
$29.95/Box of TEN
SPECIFY SOFT, 10, or 16 SECTOR

8" SINGLE SIDE, SINGLE DENSITY
$34.95/Box of TEN

## DISK DRIVES

MPI B51 5 1/4 " .......... $295.00
Single or double density, up to 40 tracks, track to track access time only 5ms.

MPI B52 5 1/4" .......... $450.00
Double head version of MPI B51

Shugart SA400 5 1/4" ..... $325.00
Single Density, 35 Track

Siemens FDD100-8 8" ..... $495.00
Shugart 801R replacement, Single or double density, runs cooler and quieter.

Siemens FDD200-8 8" ..... $575.00
Double head version of FDD100-8 capable of double density, double sided storage.

Shugart 801R 8 " ......... $875.00
Hard or soft sectored, 400K BYTE drive.

## FLOPPY DISK INTERFACE

JADE FLOPPY DISK (Tarbell board)

| | |
|---|---|
| Kit | $195.00 |
| Assembled & Tested | $250.00 |

## S.D. Computer Products VERSA-FLOPPY

| | |
|---|---|
| Kit | $159.95 |
| Assembled & Tested | $239.00 |

## Vista V80 Mini Disk SYSTEM FOR TRS—80 $395.00

Includes disk drive, power supply, regulator board, and compact case. The V-80 offers 23% more storage capacity. Simply take it out of the box, plug in the cable, and it's ready to run. Requires 16K, Level II, expansion interface.

**Interface Cable .. $24.95**

## Vector Plugboards

8800V
Universal/Microcomputer/Processor Plugboard Use With S-100 Bus. Complete With Heat Sink & Hardware.
5.3" x 10" x 1/16"
$19.95

8801-1
Same as 880V Except Plain, Less Power Buses & Heat Sink.
$15.95

P Pattern Plugboards For I.C.'s. Epoxy Glass 1/18" 44 Pin. Connector Space .156
3662 6.5" x 4.5" .... $7.65
3662-2 9.6" x 4.5" $11.45.

Hi-Density Dual-In-Line Plugboard For Wire Wrap With Power & Grd. Bus Epoxy Glass 1/16" 44 Pin Con Spaced .156
3682 9.6" x 4.5" .. $10.97
3682-2 6.5" x 4.5 ... $9.81

Gen Purpose D.I.P. Boards With Bus Pattern For Solder Or Wire Wrap. Epoxy Glass 1/16" 44 Pin Con. Space .156
3677 9.6" x 4.5" .. $10.90
3677-2 6.5" x 4.5 ... $9.74

3690-12
Card Extender
Card Extender Has 100 Contacts 50 Per Side ON .125 centers. Attached Connector Is Compatible With S-100 Bus Systems
$25.83
3690
6.5" 22/4 Pin .158 Centers Extenders.
$13.17

## NOVATION CAT

ACOUSTIC MODEM
Features Include: 300 Baud Answer/Originate, Bell 103, Comes Assembled and Tested

Midsummer Special $189.00

## JADE Computer Products

4901 W. ROSECRANS AVE., HAWTHORNE, CALIFORNIA, 90250

ORDER TOLL FREE

**800-421-5809** CONTINENTAL U.S.
**800-262-1710** INSIDE CALIFORNIA

**WRITE FOR OUR FREE CATALOG**

Cash, checks, money orders, and credit cards accepted. Minimum order: $10.00. California residents add 6 sales tax Minimum shipping and handling charge: $2.50. Discounts available at OEM quantities.

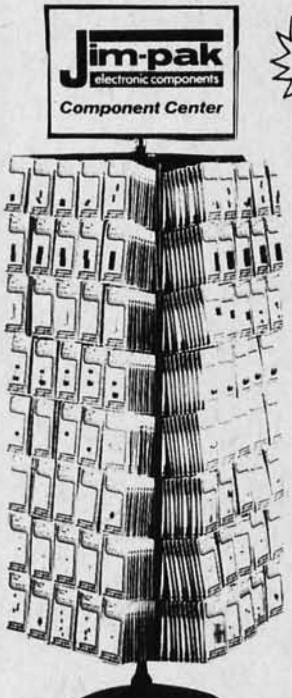
Circle 195 on inquiry card.

INTEGRATED CIRCUITS MICROPROCESSOR LED'S SOCKETS CAPACITORS DIODES TRANSISTORS RESISTORS POTENTIOMETERS

SWITCHES CRYSTALS CONNECTORS HEAT SINKS TEST CLIPS BOOKS KITS DIP JUMPERS CABLE ASSEMBLIES SWITCHES CRYSTALS CONNECTORS HEAT SINKS

Jim-pak® electronic components

# One-Stop Component and Kit Center

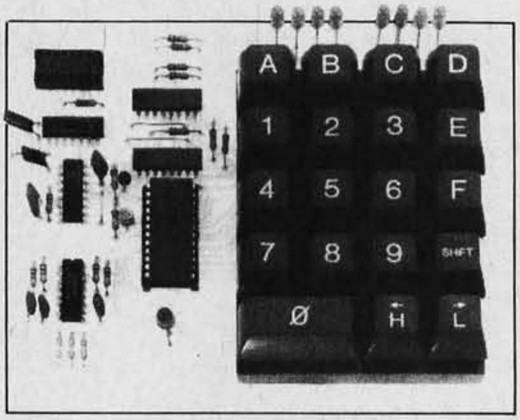



new

# JE600 Hexadecimal Encoder Kit

## FULL 8 BIT LATCHED OUTPUT – 19 KEY BOARD

The JE600 Encoder Keyboard provides two separate hexadecimal digits produced from sequential key entries to allow direct programming for 8 bit microprocessor or 8 bit memory circuits. Three (3) additional keys are provided for user operations with one having a bistable output available. The outputs are latched and monitored with 9 LED readouts. Also included is a key entry strobe.




FEATURES:

- Full 8 bit latched output for microprocessor use
- 3 User Define keys with one being bistable operation
- Debounce circuit provided for all 19 keys
- 9 LED readouts to verify entries
- Easy interfacing with standard 16 pin IC connector
- Only +5VDC required for operations

## NOW!!! OVER 300 AUTHORIZED DISTRIBUTORS . . . HERE'S JUST A SAMPLING:

| | |
|---|---|
| **ALABAMA** | |
| *Mobile* | *Lafayette Radio Electronics* |
| **ALASKA** | |
| *Anchorage* | *TV Mart* |
| **ARIZONA** | |
| *Yuma* | *Yuma Electronics* |
| **CALIFORNIA** | |
| *Anaheim* | *Heathkit Electronic Center* |
| *Antioch* | *Radio Mart* |
| *Bellflower* | *Earl's Hobby Shop* |
| *Berkeley* | *Al Lasher Electronics* |
| *Fullerton* | *Orvac Electronics Inc.* |
| *Mission Viejo* | *Tower Electronics Corp.* |
| *Monterey* | *Zackit* |
| *Oceanside* | *Electronic Center* |
| *Palo Alto* | *Zack Electronics* |
| *Pasadena* | *Dow Radio Inc.* |
| *Sacramento* | *Heathkit Electronic Center* |
| *Sacramento* | *Zackit* |
| *San Carlos* | *J & H Outlet Store* |
| *San Diego* | *Radio Shack A.S.C.–Mira Mesa* |
| *San Fernando* | *San Fernando Electronics* |
| *San Francisco* | *Zack Electronics* |
| *San Francisco* | *Zenith Distributing Corp.* |
| *San Jose* | *United Radio & TV Supply* |
| *San Luis Obispo* | *Mid-State Electronic Supply* |
| *Santa Ana* | *Quality Electronics* |
| *Santa Cruz* | *Santa Cruz Electronics* |
| *Santa Maria* | *Caps Electronics* |
| *Santa Monica* | *Mission Control* |
| *Suisun City* | *Byte Shop* |
| *Sunnyvale* | *Sunnyvale Electronics* |
| *Vallejo* | *Zackit* |
| *Walnut Creek* | *MicroSun Computer Center* |
| **COLORADO** | |
| *Aurora* | *Com Co Electronics* |
| *Durango* | *The Computer Shop* |
| **CONNECTICUT** | |
| *Avon* | *Heathkit Electronic Center* |
| *Bridgeport* | *Bridgeport Computer* |
| **DELAWARE** | |
| *New Castle* | *Delaware Amateur Supply* |
| **FLORIDA** | |
| *Ft. Lauderdale* | *Computers For You* |
| *Hialeah* | *Heathkit Electronic Center* |
| **GEORGIA** | |
| *Atlanta* | *Atlanta Computer Mart* |
| *Columbus* | *Radio Wholesale* |
| *Stone Mountain* | *Coleman's Electronics* |
| **HAWAII** | |
| *Honolulu* | *Integrated Circuit Supply* |
| **IDAHO** | |
| *Idaho Falls* | *Audiotronics* |
| **ILLINOIS** | |
| *Evanston* | *Tri-State Electronic Corp.* |
| *Groveland* | *Moyer Electronics* |
| *Mount Prospect* | *Tri-State Electronic Corp.* |
| **ILLINOIS (Continued)** | |
| *Oak Park* | *Spectronics* |
| *Rockford* | *Computer Store of Rockford* |
| *Schaumburg* | *Data Domain* |
| **INDIANA** | |
| *East Chicago* | *Acro Electronics Corp.* |
| *Evansville* | *Hutch & Son Inc.* |
| **IOWA** | |
| *Indianola* | *Electronix Limited* |
| **KANSAS** | |
| *Salina* | *Electronics, Inc.* |
| *Wichita* | *Amateur Radio Equipment Company* |
| **KENTUCKY** | |
| *Lexington* | *Radio-Electronic Equipment Co.* |
| *Louisville* | *Peerless Electronic Equipment Co.* |
| **LOUISIANA** | |
| *Baton Rouge* | *Pelican Electronics* |
| *Houma* | *Pelican Electronics* |
| *Metarie* | *Pelican Electronics* |
| **MARYLAND** | |
| *Annapolis* | *J & M Electronics* |
| *Baltimore* | *Harco of Baltimore* |
| *Churchville* | *Churchville Electronics* |
| *Damascus* | *Damascus CB* |
| *La Vale* | *J & M Electronics* |
| *Rockville* | *Computer Workshop* |
| *Rockville* | *Heathkit Electronic Center* |
| *Towson* | *Baynesville Electronics Inc.* |
| *Towson* | *Heathkit Electronic Center* |
| **MASSACHUSETTS** | |
| *Peabody* | *Heathkit Electronic Center* |
| *Pittsfield* | *Pittsfield Radio Co. Inc.* |
| *Waltham* | *Computer Mart Inc.* |
| *Wellesley* | *Heathkit Electronic Center* |
| *Worcester* | *RM Electronics Inc.* |
| **MICHIGAN** | |
| *Ann Arbor* | *Eric Electronics* |
| *Ann Arbor* | *Wedemeyer Electronic Supply* |
| *Canton* | *Electronic Connection* |
| *Clawson* | *Radio Supply & Engineering* |
| *Detroit* | *Heathkit Electronic Center* |
| *East Detroit* | *Heathkit Electronic Center* |
| *East Lansing* | *Eric Electronics* |
| *Flint* | *Hobby Electronic Center* |
| *Garden City* | *Computer Center* |
| *Grand Rapids* | *Radio Parts Inc.* |
| *Lansing* | *Fulton Radio Supply Co.* |
| *Lansing* | *Wedemeyer Electronic Supply* |
| *Livonia* | *Eric Electronics* |
| *Livonia* | *Radio Supply & Engineering* |
| *Midland* | *Computronix Corp.* |
| *Niles* | *Niles Radio Supply* |
| *Oak Park* | *Eric Electronics* |
| *Sterling Heights* | *Eric Electronics* |
| **MINNESOTA** | |
| *Duluth* | *Northwest Radio of Duluth* |
| *Hopkins* | *Heathkit Electronic Supply* |
| *St. Paul* | *Heathkit Electronic Supply* |
| **MISSOURI** | |
| *El Dorado Springs* | *Beckman Electronics* |
| **MONTANA** | |
| *Billings* | *Conley Radio Supply* |
| *Bozeman* | *Electronic Service & Dist.* |
| **NEBRASKA** | |
| *Lincoln* | *Altair Computer Center* |
| *Lincoln* | *Scott Electronic Supply Inc.* |
| *Omaha* | *Heathkit Electronic Center* |
| *Omaha* | *Omaha Computer Store* |
| **NEVADA** | |
| *Las Vegas* | *Century 23* |
| **NEW JERSEY** | |
| *Bricktown* | *Radio Shack Associate Store* |
| *Cherry Hill* | *The Computer Emporium* |
| *Fair Lawn* | *Heathkit Electronic Center* |
| *Ocean* | *Heathkit Electronic Center* |
| *Pennsauken* | *Lafayette Radio* |
| *Pompton Lake* | *Computer Center of N.J.* |
| *Ramsey* | *Typetronic Computer Store* |
| **NEW MEXICO** | |
| *Las Cruces* | *Mannie's Electronic Supply* |
| **NEW YORK** | |
| *Endwell* | *Computer Tree Inc.* |
| *Jericho* | *Heathkit Electronic Center* |
| *Kingston* | *Greylock Electronics* |
| *New York* | *Computer Mart of New York* |
| *North White Plains* | *Heathkit Electronic Center* |
| *Rochester* | *Heathkit Electronic Center* |
| *Troy* | *Trojan Electronics* |
| *Utica* | *Am-Com Electronics* |
| **NORTH CAROLINA** | |
| *Boone* | *Alpha Digital Systems* |
| *Durham* | *Futureworld* |
| *Greensboro* | *Byte Shop* |
| *Raleigh* | *Byte Shop of Raleigh* |
| **NORTH DAKOTA** | |
| *Fargo* | *The Computer Company* |
| **OHIO** | |
| *Bucyrus* | *Mead Electronics* |
| *Cincinnati* | *Heathkit Electronic Center* |
| *Columbus* | *Heathkit Electronic Center* |
| *Reynoldsburg* | *Universal Amateur Radio* |
| *Youngstown* | *Ross Radio Company* |
| **OKLAHOMA** | |
| *Oklahoma City* | *Bitt, Bytes & Micros* |
| *Oklahoma City* | *Heathkit Electronic Center* |
| **OREGON** | |
| *Albany* | *Oregon Ham Sales* |
| *Coos Bay* | *Herrick Electronics* |
| *Portland* | *Portland Radio Supply* |
| *Salem* | *Computer Pathways* |
| **PENNSYLVANIA** | |
| *Drums* | *Mr. D's Digital Electronics* |
| *Hershey* | *Microcomputer Systems Inc.* |
| *Philadelphia* | *Heathkit Electronic Center* |
| *Pittsburgh* | *Heathkit Electronic Center* |
| *York* | *G.Y.C. Company* |
| **RHODE ISLAND** | |
| *Cranston* | *Jabbour Electronics City* |
| *Pawtucket* | *Jabbour Electronics City* |
| *Warwick* | *Heathkit Electronic Center* |
| **TENNESSEE** | |
| *Chattanooga* | *William's Data Comp Div.* |
| *Clarksville* | *Masstronics* |
| *Cookeville* | *Wagnon's Stereo Center* |
| *Knoxville* | *Eastern Micro* |
| *Nashville* | *Electra Distributing Co.* |
| **TEXAS** | |
| *Amarillo* | *Computer Encounters Inc.* |
| *Dallas* | *CompuShop* |
| *Dallas* | *Heathkit Electronic Center* |
| *Houston* | *CompuShop* |
| *Houston* | *Interactive Computers* |
| *San Antonio* | *Appliance & Equipment Co.* |
| *San Antonio* | *Sherman Electronics Supply Inc.* |
| **UTAH** | |
| *Midvale* | *Heathkit Electronic Center* |
| *Provo* | *Alpine Electronic Supply Co.* |
| **VIRGINIA** | |
| *Alexandria* | *Computer Hardware Store* |
| *Alexandria* | *Heathkit Electronic Center* |
| *Blacksburg* | *Scotty's Radio & TV Inc.* |
| *Charlottesville* | *Graves Electronics* |
| *Falls Church* | *Crossroad Electronics* |
| *Hampton* | *Lafayette Radio* |
| *Norfolk* | *Avec Electronics Corp.* |
| *Richmond* | *Avec Electronics Corp.* |
| *Roanoke* | *The Computer Place* |
| *Virginia Beach* | *Heathkit Electronic Center* |
| **WASHINGTON** | |
| *Longview* | *Progress Electronics* |
| *Moses Lake* | *Ron's Electronics* |
| *Pasco* | *Riverview Electronics* |
| *Seattle* | *Amateur Radio Supply* |
| *Seattle* | *C-Com* |
| *Seattle* | *Empire Electronics* |
| *Spokane* | *Personal Computers* |
| *Tacoma* | *C & G Electronics* |
| **WEST VIRGINIA** | |
| *Morgantown* | *Electro Distributing Co.* |
| *Wheeling* | *Lafayette Radio* |
| **WISCONSIN** | |
| *West Allis* | *Olson Electronics* |
| **CANADA** | |
| *Alberta (Calgary)* | *The Computer Store* |
| *Ontario (Toronto)* | *House of Computers* |
| *Ontario (Willowdale)* | *Home Computer Centre* |
| *Quebec (Montreal)* | *Wang's Microcenter* |
| **ENGLAND** | |
| *Berkshire* | *NewBear Computing Store* |
| **GUAM** | *Marianas Electronics* |
| **PANAMA** | |
| *Panama City* | *Sonitel, S.A.* |
| **SINGAPORE** | *Inter-trade (PTE) Ltd.* |

## ASK YOUR ELECTRONICS STORE TO STOCK JIM-PAK® TODAY!!

**JIM-PAK® • 1021 HOWARD AVENUE, SAN CARLOS, CALIFORNIA 94070 • (415) 592-8097**

SWITCHES CRYSTALS CONNECTORS HEAT SINKS TEST CLIPS BOOKS KITS DIP JUMPERS CABLE ASSEMBLIES SWITCHES CRYSTALS CONNECTORS HEAT SINKS

INTEGRATED CIRCUITS MICROPROCESSOR LED'S SOCKETS CAPACITORS DIODES TRANSISTORS RESISTORS POTENTIOMETERS

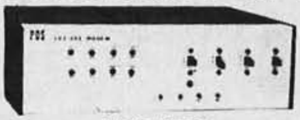


Circle 203 on inquiry card.

Circle 296 on inquiry card.

## *COMMERCIAL GRADE PERIPHERALS FOR THE MICROCOMPUTER*

**PRINTER TERMINALS**



**MODEMS**

**TAPE DRIVES**

**•ASCII SELECTRIC PRINTER/TYPEWRITER:** Why settle for less than letter-quality printout from your computer? Refurbished IBM Model 725 can be used as off-line typewriter or on-line printer. Complete with solenoids, power supply, case and ASCII interface card (TTL to CPU parallel port.) Interface includes programmable ASCII translation table on EPROM with up to 8 tables for use with various type spheres. Feedback signals on completion of each print cycle insures fastest printing speed (15 cps.)
Price: programmed w/3 translation tables (one type sphere): ......... $695.00

**•SELECTRIC I/O TERMINALS** (by GTE/Information Systems). Both ASCII & IBM code versions with microcomputer interface software & hardware (RS-232 connector.) Cassette drive models permit up to 2400 baud data transfer rate as well as off-line data storage, use as memory typewriter, & use as data entry device for office personnel familiar with Selectric typewriters but not computers. Wide-carriage, interchangeable type spheres; optional built-in modem. All units cleaned, adjusted & warranted.
- Model 5541 (IBM Correspondence code) ................ $695.00
- Model 5550 (corres. code, built-in cassette drive) .......... $1195.00
- Model 5560 (ASCII code, built-in cassette drive) .......... $1295.00

**•DIABLO MODEL 1550 "DAISY-WHEEL" TERMINAL:** Refurbished letter-quality terminal with Hytype I printer (30 cps), full ASCII keyboard, RS-232 serial prot, 110, 150, 300 baud, 768 char. print buffer, software controlled graphics mode (1/60 in. spacing), built-in stand. SHIP WT. 200 lb. Price, refurbished: ............ $1895.00

**•DIABLO HYTYPE I Model 1200 PRINTER MECHANISM:** used, complete and tested. Requires power supply, case & mCPU interface. 15 day return privilege — no other warranties. LIMITED QUANTITY! .......... $750.00
- –6' Ribbon cable & connector for printer Main Logic PCB ........ $10.00
- –14-pin Winchester connector & 18" power supply cable ......... $5.00
- –"As-is" spare printer PCB's for parts (Logic, Heat Sink, Control): ea. $20.00
- –New Pin-feed Platen (14"): $50 if bought w/printer; separately ... $100.00

**•POS 103/202 *"MIX or MATCH"* MODEM:** BELL 103 and/or BELL 202 FREQUENCIES: Unique POS control design permits use in one housing of both Bell-compatible 103 (0 – 300 baud) and 202 (0 – 1200 baud) modem modules originally made by VADIC Corp. for a telephone company subsidiary. FEATURES: RS-232 serial interface, auto-answer, auto-dial, LED display, telephone line interface via acoustic coupler, manual DAA, or auto-answer DAA (sold separately.) FULLY ADJUSTED; no special tools required. 3,000 mile range over standard dial-up telephone lines.
- –POS 103 MODEM (with Auto Answer, Auto Dial) ........... $179.95
- –POS 202 MODEM (Half-Duplex with Reverse Channel)........ $249.95
- –POS 202 MODEM (Half-Duplex w/Rev. Ch., Auto-Answer) ..... $279.95
- –POS 103/202 MODEM (Auto-Answer, Auto-Dial)........... $399.95

**•POS-100 NRZ1 TAPE DRIVE CONTROLLER/FORMATTER:** Designed as interface between S-100 bus mCPU and 9-track, 800 BPI, NRZ1 tape drive. Allows microcomputerist to read and write IBM-compatible ½" mag tapes. Software provided for 8080 or Z-80 systems. Requires modification for drives of various mfrs.
Price: (Includes S-100 card, controller card, 10' cable, software listing) . $750.00

**•NRZ1 TAPE DRIVE by WILLARD LABS.** 9-track, 800 BPI, NRZ1 format, 12"/sec., 1200 ft. reels (10 megabyte capacity) Fully tested and warranted $599.00

**•CONVERT 15" IBM OFFICE SELECTRIC TO I/O TYPEWRITER:** Kit includes assembled solenoids, switches, wire harness, magnet driver PCB plus instructions for installation and mCPU interface ..................$200.00

**•DIGITAL CASSETTE DRIVE** (from GTE/IS Terminal): 1800 baud, 6"/sec; AC motor; fwd/rewnd circuitry plus tape head, no read/write electronics $25.

**•FORMS TRACTORS,** Moore Variable width "Form A-Liner" for print terminals:
- a) Model 565P for 15" Carriage IBM Selectrics (new):............. $50.00
- b) Model K81 for QUME or DIABLO Hytype I or II printers (new): .... $90.00

**•POWER SUPPLIES** for Disk Drive, mCPU, tested under load shown:
- –No. 519 (w/fan & AC cord): +5V reg., ±12V reg., +24V, @4A (10 lb.). .$39.95
- –LAMBDA No. LMEE5 w/OV protect: +5V reg. @ 25A (35 lb.)...... $69.95

**NO RISK! 15 DAY APPROVAL ON ALL MAIL-ORDERS**

**Full documentation included PLUS interface instructions where indicated. All equipment is shipped insured FOB Palo Alto within 14 days after check clears or COD order is received. Prices may change without notice.**

**PACIFIC OFFICE SYSTEMS, INC.**
2600 El Camino Real, Suite 502
Palo Alto, Calif. 94306
(415) 321-3866

Call or write for details, quantity prices, catalog. 15 day return privilege PLUS 90 day no charge replacement of defective parts. All orders shipped from stock. No back orders, no substitutions. M/C & VISA accepted.

---

# ProComp/New England

*120 Boylston Street/Fourth Floor*
*Boston, MA/02116 Phone 617·482·4450*
*HOURS: M-F 10-5, S 10-4*

## Hardware

*Seattle 16K Static Ram*
*Measurement Systems 48 64K Dynamic Ram*
*Micromation Doubler Disk Controllers*
*Computalker CT-1*
*Cromemco Single Card Computer*
*...AND MORE*

## Software

*Graham·Dorian*
*Structured Systems Group*
*National Software Exchange*
*Selector III*
*...AND MORE*

## Systems

*Compucorp 625 MkII & 665*
*Imsai VDP 42 & VDP 80*
*Complete Double Density Disk Systems*

## Supplies

*Diskettes*
*Storage Binders*
*Printer Ribbons*

□ *SPECIAL BARGAINS ON USED EQUIPMENT!!!*

□ *Special prices on these and other in-stock items during our Grand Opening Sale now through September 30th.*

□ *Write, call or stop in for our catalog and Grand Opening Sale Price List.*

□ *VISA and MASTERCHARGE*

Circle 69 on inquiry card.

# Electrolabs
## POB 6721, Stanford, Ca. 94305

In California: 415-321-5601
Elsewhere: 800-227-8266
TLX: 345567

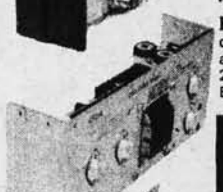
# NEW CATALOGUE

Crystals
Integrated circuits
Keyboards
Lasers
LSI-11
Media
RAMs
S-100 Components
Z-80 Components


Low Profile Solder-Tail Sockets 0.75/Pin (1000's) 1Ct./Pin,

## FLOPPY SYSTEMS

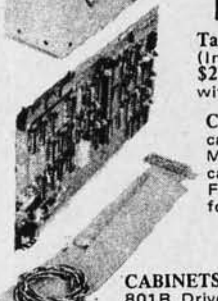
**8" Siemens FDD120-8 Drive** All Siemen's options included in this drive which can be configured hard or soft and single or double density. (Others give only stripped unit) **$399.00**



**"Power One" Model CP206 Floppy Power Unit.** For two drives going full-out, and possably more on less severe service. 2.8A@24V, 2.5A@5V, 0.5A@–5V. Beautiful quality. **$99.00**

**DISKETTES** (Standard)

| | | |
|---|---|---|
| 8" | Boxed 10 | $39.00 |
| 5¼" | Boxed 10 | $34.95 |



**Tarbell ("It Works") Interface** (Includes cable set for 2 drives) **$265.00** BUT ONLY **$219.00** with purchase of two drives.

Cable Kits 10' with 50 cond. cable and connectors and also Molex connectors and power cable: For one drive: **$27.50** For two drives: **$33.95**, and for three drives: **$38.95**

**CABINETS** for FDD120 and 801R Drives, or CP206 power supply. Matte finish in mar resistant black epoxy paint. Stacking type design. **$29.99**

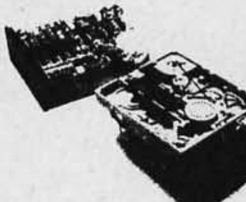
# ANNOUNCEMENT! SUPERDISK

**ELECTROLABS is proud to announce appointment as DISTRIBUTOR by CII-Honeywell Bull.**

| | |
|---|---|
| **PRICE BREAKTHROUGH on *SUPERDISK* 10 MBY!** | **$3495.00** |
| **General purpose controller (requires 2 parallel I/O ports)** | **1500.00** |
| **S-100 Controller (DMA) ........................** | **995.00** |
| **"RL–01/RK–05" surrogate .....................** | **1900.00** |
| **(transparent to RT, RS, RX)** | |

**SOFTWARE:** (CP/M Compatible)
**SUPERDOS 1**
**(Z-80) ..... $695.00**
**MICRODOS 1**
**(TRS-80) ... $199.00**
**Power supply (switching)**
**.......... $395.00**
**Enclosure (desktop)**
**.......... $ 99.00**

Removeable Media Cartridge Drive



**5¼" MINI-FLOPPY DRIVE $299.00**

**– single or double density – quick access time – high reliability & durability**

**Mini-floppy CABLE KIT: for TRS-80 or your Tarbell controller. $24.95**

## Daisy Wheel Printers
## Qume Sprint 3\45

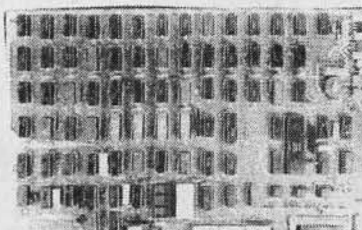
New!

Print wheels $8.95  Ribbons $5.95

**OEM Style mechanism $1399.00**

Used Sylvania 12" Video Monitors. Composite video 15mhz, 115vac, 50/60hz New Tube. As shown $109 OEM style without case: $99, Anti-glare tube option add $12. Specify p4 or p39

## ESAT 200B

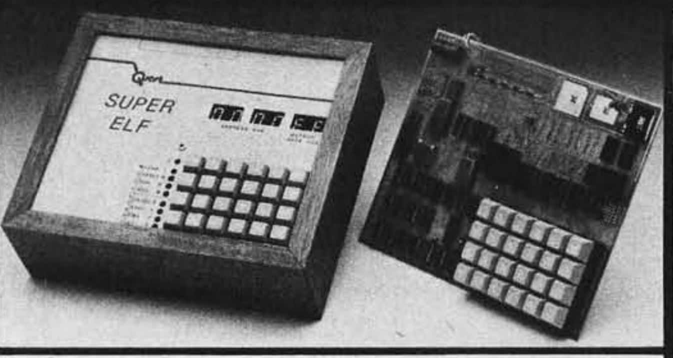


**BI-LINGUAL 80x24 COMMUNICATING TERMINAL**


**Scrolling, full cursor, bell, 8x8 metrix, 110-19,200 baud, Dual Font Applications. Arabic & Hebrew, Multilingual Data Entry, Forms Drawing, Music Instruction, Specialized Graphics (e.g. Games, Chemical Plants, Switchyards) $349.00** We carry keyboards, cases, power supplies, etc., enough to make an entire system.



Circle 115 on inquiry card.




## RCA Cosmac Super Elf Computer $106.95

Compare features before you decide to buy any other computer. There is no other computer on the market today that has all the desirable benefits of the **Super Elf** for so little money. The Super Elf is a small single board computer that does many **big** things. It is an excellent computer for training and for learning programming with its machine language and yet it is easily **expanded** with **additional memory, Tiny Basic, ASCII Keyboards, video character generation, etc.**

The **Super Elf** includes a **ROM** monitor for program loading, editing and execution with **SINGLE STEP for program debugging** which is not included in others at the same price. With **SINGLE STEP** you can see the microprocessor chip operating with the **unique Quest** address and data bus displays **before, during** and **after** executing instructions. Also, CPU mode and instruction cycle are decoded and displayed on 8 LED indicators.

An RCA 1861 **video graphics** chip allows you to connect to your own TV with an inexpensive video modulator to do graphics and games. There is a **speaker system** included for writing your own music or using many music programs already written. The speaker amplifier may also be used to drive relays for control purposes.

A 24 key **HEX keyboard** includes 16 HEX keys plus **load, reset, run, wait, input, memory protect, monitor select** and **single step**. Large, on board displays provide output and optional **high and low address**. There is a 44 pin standard connector for PC cards and a 50 pin connector for the **Quest Super Expansion Board**. Power supply and sockets for **all** IC's are included in the price plus a detailed 127 pg. instruction manual which now includes over 40 pgs. of software info. including a series of lessons to help get you started and a music program and graphics target game.

Remember, other computers only offer Super Elf features at additional cost or not at all. **Compare before you buy. Super Elf Kit $106.95, High address option $8.95, Low address option $9.95. Custom Cabinet** with drilled and labelled plexiglass front panel **$24.95.** Expansion Cabinet with room for 4 S-100 boards **$41.00. NiCad Battery Memory Saver Kit $6.95.** All kits and options also come completely assembled and tested.

**Questdata**, a 12 page monthly software publication for 1802 computer users is available by subscription for $12.00 per year.

### Attention Elf Owners

**New products in hardware and software coming soon.**

Tiny Basic cassette $10.00, on ROM $38.00, original Elf kit board $14.95.

## Super Expansion Board with Cassette Interface $89.95

This is truly an astounding value! This board has been designed to allow **you** to decide how you want it optioned. **The Super Expansion Board comes with 4K of low power RAM** fully addressable anywhere in 64K with built-in memory protect and a **cassette interface.** Provisions have been made for all other options on the same board and it fits neatly into the hardwood cabinet alongside the **Super Elf.** The board includes slots for up to 6K of **EPROM** (2708, 2758, 2716 or TI 2716) and is **fully socketed.** EPROM can be used for the monitor and Tiny Basic or other purposes.

**A IK Super ROM Monitor $19.95** is available as an on board option in 2708 EPROM which has been preprogrammed with a program loader/editor and error checking multi file cassette read/write software, (relocatible cassette file) another exclusive from Quest. It includes register save and readout, block move capability and video graphics driver with blinking cursor. Break points can be used with the register save feature to isolate program bugs quickly, then follow with single step. The Super Monitor is written with subroutines allowing users to take advantage of monitor functions simply by calling them up. Improvements and revisions are easily done with the monitor. If you have the **Super Expansion Board** and **Super Monitor** the monitor is up and running at the push of a button.

Other on board options include **Parallel Input and Output Ports** with **full handshake.** They allow easy connection of an ASCII keyboard to the input port. **RS 232 and 20 ma Current Loop** for teletype or other device are on board and if you need more memory there are two **S-100** slots for static RAM or video boards. A Godbout 8K RAM board is available for $135.00. Also a 1K Super Monitor version 2 with video driver for full capability display with Tiny Basic and a video interface board. **Parallel I/O Ports $9.85, RS 232 $4.50, TTY 20 ma I/F $1.95, S-100 $4.50. A 50 pin connector set** with ribbon cable is available at $12.50 for easy connection between the **Super Elf** and the **Super Expansion Board.**

The **Power Supply Kit** for the Super Expansion Board is a 5 amp supply with multiple positive and negative voltages $29.95. Add $4.00 for shipping. Prepunched frame $7.50. Case $10.00. Add $1.50 for shipping.

P.O. Box 4430X Santa Clara, CA 95054
For will call only: (408) 988-1640
2322 Walsh Ave.

**Quest ELECTRONICS**

**Same day shipment.** First line parts only. Factory tested. Guaranteed money back. Quality IC's and other components at factory prices.

## INTEGRATED CIRCUITS

| 7400TTL | |
|---|---|
| 7400N | .17 |
| 7402N | .17 |
| 7404N | .19 |
| 7409N | .23 |
| 7410N | .17 |
| 7414N | .63 |
| 7420N | .17 |
| 7422N | 1.39 |
| 7430N | .20 |
| 7442N | .50 |
| 7445N | .69 |
| 7447N | .60 |
| 7448N | .69 |
| 7450N | .17 |
| 7474N | .29 |
| 7475N | .49 |
| 7485N | .88 |
| 7489N | 2.00 |
| 7490N | .43 |
| 7492N | .43 |
| 7493N | .43 |
| 7495N | .69 |
| 74100N | .90 |
| 74107N | .29 |
| 74121N | .34 |
| 74123N | .59 |
| 74125N | .39 |
| 74145N | .69 |
| 74150N | .95 |
| 74151N | .69 |
| 74154N | 1.00 |
| 74157N | .69 |
| 74161N | .87 |
| 74162N | .87 |
| 74163N | .87 |
| 74174N | .96 |
| 74175N | .90 |
| 74190N | 1.15 |
| 74192N | .87 |
| 74193N | .85 |
| 74221N | 1.55 |
| 74298N | 1.65 |
| 74365N | .95 |
| 74366N | .95 |
| 74367N | .95 |

| 74LS00 TTL | |
|---|---|
| 74LS00N | .25 |
| 74LS02N | .25 |
| 74LS04N | .25 |
| 74LS05N | .25 |
| 74LS08N | .25 |
| 74LS10N | .25 |
| 74LS13N | .40 |
| 74LS14N | .90 |
| 74LS20N | .25 |
| 74LS22N | .25 |
| 74LS28N | .41 |
| 74LS30N | .25 |
| 74LS33N | .39 |
| 74LS38N | .30 |
| 74LS74N | 1.15 |
| 74LS75N | .47 |
| 74LS90N | .51 |
| 74LS93N | .51 |
| 74LS95N | 1.39 |
| 74LS107N | .35 |
| 74LS112N | .35 |
| 74LS113N | .35 |
| 74LS132N | .72 |
| 74LS136N | .35 |
| 74LS151N | .67 |
| 74LS155N | .67 |
| 74LS157N | .67 |
| 74LS162N | .91 |
| 74LS163N | .91 |
| 74LS174N | .95 |
| 74LS190N | 1.06 |
| 74LS221N | 1.95 |
| 74LS258N | .67 |
| 74LS367N | 1.35 |

| LINEAR | |
|---|---|
| CA3045 | .90 |
| CA3046 | .67 |
| CA3081 | 1.80 |
| CA3082 | 1.90 |
| CA3089 | 2.95 |
| LM301AN/AH | .35 |
| LM305H | .87 |
| LM307N | .35 |
| LM308N | .89 |
| LM309H | 1.15 |
| LM309K | 1.50 |
| LM311H/N | .90 |
| LM317T/K | 2.92 |
| LM318 | 1.35 |
| LM320K-5 | 1.60 |
| LM323K-5 | 6.95 |
| LM320K-12 | 1.60 |
| LM320K-15 | 1.60 |
| LM320T-5 | 1.60 |
| LM320T-8 | 1.60 |
| LM320T-12 | 1.50 |
| LM320T-15 | 1.60 |
| LM324N | 1.15 |
| LM339N | 1.55 |
| LM340K-5 | 1.35 |
| LM340K-8 | 1.35 |
| LM340K-12 | 1.35 |
| LM340K-15 | 1.35 |
| LM340K-24 | 1.35 |
| LM340T-5 | 1.25 |
| LM340T-8 | 1.25 |
| LM340T-12 | 1.25 |
| LM340T-15 | 1.25 |
| LM340T-18 | 1.25 |
| LM340T-24 | 1.25 |
| LM343H | 4.50 |
| LM350 | 7.50 |
| LM370 | 1.15 |
| LM377 | 3.00 |

| | |
|---|---|
| LM379M | 5.00 |
| LM380N | 1.00 |
| LM381 | 1.60 |
| LM382 | 1.60 |
| LM703H | .40 |
| LM709H | .28 |
| LM723H/N | .50 |
| LM733N | .67 |
| LM741CH | .35 |
| LM741N | .25 |
| LM747H/N | .62 |
| LM748N | .35 |
| LM1303N | .82 |
| LM1304 | 1.10 |
| LM1305 | 1.27 |
| LM1307 | 2.00 |
| LM1310 | 2.75 |
| LM1458 | .47 |
| LM1800 | 1.75 |
| LM1812 | 7.50 |
| LM1889 | 3.00 |
| LM2111 | 1.75 |
| LM2902 | 1.50 |
| LM3900N | .60 |
| LM3905 | 1.75 |
| LM3909N | .61 |
| MC1458V | .50 |
| NE550N | .65 |
| NE555V | .43 |
| NE556A | .79 |
| NE565A | 1.00 |
| NE566V | 1.50 |
| NE567V | 1.20 |
| NE570B | 5.00 |
| NE571B | 5.00 |
| 78L05 | .60 |
| 78L08 | .60 |
| 79L05 | .70 |
| 78M05 | .85 |
| 75108 | 1.75 |
| 75491CN | .50 |
| 75492CN | .55 |
| 75494CN | .89 |

| A to D CONVERTER | |
|---|---|
| 8038B | 4.50 |
| 8700CJ | 13.95 |
| 8701CN | 15.95 |
| 8750CJ | 13.95 |
| LD130 | 9.95 |
| 9400CJV/F | 7.40 |
| ICL7103 | 9.50 |
| ICL7107 | 14.25 |
| 8702 | 17.95 |

| CMOS | |
|---|---|
| CD34001 Fair. | .50 |
| CD4000 | .16 |
| CD4001 | .28 |
| CD4002 | .28 |
| CD4006 | 1.10 |
| CD4007 | .28 |
| CD4008 | .28 |
| CD4009 | .39 |
| CD4010 | .39 |
| CD4011 | .28 |
| CD4012 | .28 |
| CD4013 | .36 |
| CD4014 | .86 |
| CD4015 | .86 |
| CD4016 | .36 |
| CD4017 | .94 |
| CD4018 | .94 |
| CD4019 | .28 |
| CD4020 | 1.02 |
| CD4021 | 1.02 |
| CD4022 | .86 |
| CD4023 | .28 |
| CD4024 | .75 |
| CD4025 | .28 |
| CD4026 | 1.51 |
| CD4027 | .36 |
| CD4028 | .79 |
| CD4029 | 1.02 |
| CD4030 | .28 |
| CD4035 | 1.02 |
| CD4040 | 1.02 |
| CD4042 | .71 |
| CD4043 | .63 |
| CD4044 | .63 |
| CD4046 | 1.67 |
| CD4049 | .36 |
| CD4050 | .36 |
| CD4051 | 1.13 |
| CD4060 | 1.42 |
| CD4066 | .71 |
| CD4068 | .40 |
| CD4069 | .40 |
| CD4070 | .40 |
| CD4071 | .28 |
| CD4072 | .28 |
| CD4073 | .28 |
| CD4075 | .28 |
| CD4076 | 1.75 |
| CD4078 | .40 |
| CD4081 | .28 |
| CD4082 | .28 |
| CD4116 | .47 |
| CD4490 | 5.50 |
| CD4507 | 1.00 |
| CD4508 | 4.25 |
| CD4510 | 1.02 |
| CD4511 | .94 |
| CD4515 | 2.52 |
| CD4516 | 1.10 |
| CD4518 | 1.02 |
| CD4520 | 1.02 |
| CD4527 | 1.51 |
| CD4528 | .79 |
| CD4553 | 3.50 |
| CD4566 | 2.25 |
| CD4583 | 4.50 |

| | |
|---|---|
| CD4585 | 1.10 |
| CD40192 | 3.00 |
| 74C00 | .28 |
| 74C04 | .33 |
| 74C10 | .28 |
| 74C14 | 2.10 |
| 74C20 | .28 |
| 74C30 | .28 |
| 74C48 | 1.95 |
| 74C74 | .75 |
| 74C76 | 1.40 |
| 74C90 | 1.15 |
| 74C93 | 1.40 |
| 74C154 | 3.00 |
| 74C160 | 1.44 |
| 74C175 | 1.35 |
| 74C192 | 1.65 |
| 74C221 | 2.00 |
| 74C905 | 3.00 |
| 74C906 | .75 |
| 74C914 | 1.95 |
| 74C922 | 5.50 |
| 74C923 | 5.50 |
| 74C925 | 6.95 |
| 74C926 | 6.95 |
| 74C927 | 6.95 |

| INTERFACE | |
|---|---|
| 8095 | .95 |
| 8096 | .95 |
| 8097 | .95 |
| 8098 | .95 |
| 8T09 | 1.25 |
| 8T10 | 4.50 |
| 8T13 | 3.00 |
| 8T20 | 5.50 |
| 8T23 | 3.10 |
| 8T24 | 3.50 |
| 8T25 | 3.20 |
| 8T26 | 1.69 |
| 8T28 | 2.75 |
| 8T97 | 1.69 |
| 8T98 | 1.69 |

| MOS/MEMORY RAM | |
|---|---|
| 2101-1 | 3.95 |
| 2102-1 | .95 |
| 2102AL-4 | 1.60 |
| 21L02-1 | 1.18 |
| 21F02 | 1.25 |
| 2104A-4 | 4.95 |
| 2107B-4 | 3.75 |
| 2111-1 | 3.75 |
| 2112-2 | 3.95 |
| 2114L-3 | 6.75 |
| 4116 | 10.95 |
| 2513B | 6.30 |
| MM5262 | .40 |
| MM5280 | 3.00 |
| MM5320 | 9.95 |
| MM5330 | 5.94 |
| PD411D-3 | 4.00 |
| PD411D-4 | 5.00 |
| P5101L | 13.95 |
| 4200A | 9.95 |
| 82S25 | 2.90 |
| 91L02A | 1.50 |
| HD0165-5 | 6.95 |
| MM57100 | 4.50 |
| GIAY38500-1 | 9.95 |
| MCM6571A | 9.95 |
| 9368 | 3.50 |
| 4100 | 10.00 |
| 416 | 16.00 |

| CLOCKS | |
|---|---|
| MM5309 | 3.00 |
| MM5311 | 3.60 |
| MM5312 | 4.80 |
| MM5313 | 3.60 |
| MM5314 | 3.90 |
| MM5315 | 4.00 |
| MM5316 | 5.00 |
| MM5318 | 3.60 |
| MM5369 | 2.10 |
| MM5841 | 14.45 |
| MM5865 | 7.95 |
| CT7001 | 5.80 |
| CT7010 | 8.95 |
| CT7015 | 8.95 |
| MM5375AA/N | 3.90 |
| MM5375AB/N | 4.90 |
| 7205 | 16.50 |
| 7207 | 7.50 |
| 7208 | 15.95 |
| 7209 | 4.95 |
| DS0026CN | 3.75 |
| DS0056CN | 3.75 |
| MM53104 | 2.50 |

| MICROPROCESSOR | |
|---|---|
| 6800 | 17.50 |
| 6802 | 18.75 |
| 8080A with data | 8.95 |
| 8085 | 27.00 |
| Z80A | 19.75 |
| 8212 | 2.90 |
| 8214 | 8.00 |
| 8216 | 2.90 |
| 8224 | 2.90 |
| 8228 | 5.35 |
| 8251 | 8.50 |
| 8253 | 10.00 |
| 8255 | 9.25 |
| 8257 | 19.50 |
| 8259 | 19.50 |
| 1802CP plas. | 13.95 |
| 1802DP plas. | 17.95 |
| 1861P | 11.50 |
| CDP1802CD | 19.95 |
| CDP1802D | 25.00 |
| CDP1861 | 12.95 |

| | |
|---|---|
| 6820 | 9.95 |
| 6850 | 12.95 |
| 6502 | 12.50 |
| 6504 | 16.50 |
| 6522 | 13.60 |

**IC SOCKETS**
**Solder Tin Low Profile**

| PIN | 1 UP | PIN | 1UP |
|---|---|---|---|
| 8 | .15 | 22 | .30 |
| 14 | .14 | 24 | .35 |
| 16 | .16 | 28 | .42 |
| 18 | .27 | 36 | .58 |
| 20 | .29 | 40 | .57 |

3 level wire wrap gold.
14 pin .25
2 level 14 pin ww .20

**WIRE WRAP LEVEL 3**

| PIN | | PIN | |
|---|---|---|---|
| 14 | .25 | 24 | .86 |
| 16 | .33 | 28 | 1.00 |
| 18 | .57 | 40 | 1.23 |

| UART/FIFO | |
|---|---|
| AY5-1013 | 5.50 |
| AY5-1014 | 7.50 |
| 3341 | 6.95 |

| PROM | |
|---|---|
| 1702A | 3.95 |
| N82S23 | 2.95 |
| N82S123 | 3.50 |
| N82S126 | 3.75 |
| N82S129 | 3.75 |
| N82S131 | 3.75 |
| N82S136 | 8.75 |
| N82S137 | 8.75 |
| 2708 | 10.50 |
| DM8577 | 2.90 |
| 8223 | 2.90 |
| 2716T1 | 29.50 |
| 2716 Intel | 48.00 |

| CRYSTALS | |
|---|---|
| 1 MHz | 4.50 |
| 2 MHz | 4.50 |
| 4 MHz | 4.25 |
| 5 MHz | 4.25 |
| 10 MHz | 4.25 |
| 18 MHz | 3.90 |
| 20 MHz | 3.90 |
| 32 MHz | 3.90 |
| 32768 MHz | 4.00 |
| 1.8432 MHz | 4.50 |
| 3.5795 MHz | 1.20 |
| 2.0100 MHz | 1.95 |
| 2.097152 MHz | 4.50 |
| 2.4576 MHz | 4.50 |
| 3.2768 MHz | 4.50 |
| 5.0688 MHz | 4.50 |
| 5.185 MHz | 4.50 |
| 5.7143 MHz | 4.50 |
| 6.5536 MHz | 4.50 |
| 14.31818 MHz | 4.25 |
| 18.432 MHz | 4.50 |
| 22.1184 MHz | 4.50 |

| CONNECTORS | |
|---|---|
| 44 pin edge | 2.75 |
| 100 pin edge | 4.50 |
| 100 pin edge WW | 4.75 |

| KEYBOARD ENCODERS | |
|---|---|
| AY5-2376 | $12.50 |
| AY5-3600 | 17.95 |
| 74C922 | 5.50 |
| 74C923 | 5.50 |
| HD0165-5 | 6.95 |

| IC Test Clips | 1 | 10 |
|---|---|---|
| Red | .55 | .47 |
| Black | .55 | .47 |

**Keyer** 8043 14.50
comp. w/spec./sock.

| TRANSISTORS | |
|---|---|
| 2N1893 | .40 |
| 2N2222A | .18 |
| 2N2369 | .30 |
| 2N2904A | .20 |
| 2N2907A | .25 |
| 2N3053 | .40 |
| 2N3638 | .25 |
| 2N3643 | .25 |
| 2N3904 | .18 |
| 2N3906 | .18 |
| 2N3055 | .69 |
| 2N4400 | .25 |
| 2N4401 | .75 |
| 2N4402 | .20 |
| TIP31 | .60 |
| TIP33A | 1.00 |

**D Connectors RS232**
25 Pin Subminiatures

| | |
|---|---|
| DB25P | 2.95 |
| DB25S | 3.95 |
| Cover | 1.50 |
| RS232 Complete Set | 6.50 |
| DE9P | 1.50 |
| DE9S | 1.95 |
| DA15P | 2.10 |
| DA15S | 3.10 |

**CLOCK MODULES** Complete alarm clocks ready to hook up with transformer and switches. Very compact with .50" and .84" digits.

| | |
|---|---|
| **MA1002A, C or E .50"** | **8.95** |
| **102P3 Transformer** | **2.25** |
| **MA1010A, C or E .84"** | **11.95** |
| **102P2 Transformer** | **2.25** |
| **Special transformer and six switches when purchased w/module** | **2.95** |
| **MA1003 car module .3" green fluor. display** | **15.95** |

**RESISTORS** ¼ watt 5%
10 per type .03 1000 per type .012
25 per type .025 350 piece pack
100 per type .015 5 per type 6.75

| KEYBOARDS | |
|---|---|
| 56 key ASCII keyboard kit | $67.50 |
| Fully assembled | 77.50 |
| 53 key ASCII keyboard kit | 60.00 |
| Fully assembled 70.00 Enclosure | 14.95 |

| LEDS | |
|---|---|
| Red T018 | .15 |
| Green, Yellow T018 | .20 |
| Jumbo Red | .20 |
| Green, Orange, Yellow Jumbo | .25 |
| **Cliplite LED Mounting Clips** | 8/$1.25 |

(specify red, amber, green, yellow, clear)

**CONTINENTAL SPECIALTIES in stock**
Complete line of breadboard test equip.
**MAX-100 8 digit Freq. Ctr. $128.95**

**OK WIRE WRAP TOOLS in stock**
**Portable Multimeter $18.00**

**DIGITAL THERMOMETER $48.50**
Batt. oper. General purpose or medical 32°–230°F. Disposable probe cover ±.2° accuracy. Comp. Assy. in compact case. Switches from F° to C°.

| COMPUTER BOARD KITS | |
|---|---|
| 8K RAM Board Kit | $135.00 |
| 4K EPROM Kit | 114.95 |
| I/O Board Kit | 44.50 |
| Extender Board w/connector | 12.50 |
| 16K EPROM board kit w/o PROMS | 74.50 |
| **North Star Floppy Disk Kit** | $665.00 |
| Additional Drive Kit | 415.00 |

| SPECIAL PRODUCTS | |
|---|---|
| MM5865 Stopwatch Timer | 9.00 |
| PC board | 7.50 |
| **Switches** Mom. Pushbutton | .27 |
| 3 pos. slide | .25 |
| **Encoder** HD0165-5 | 6.95 |
| **3 Digit Universal Counter Board Kit** Operates 5–18 Volt DC to 5 MHz typ. .125" LED display | **10.50** |
| **Voice actuated switch** | .50 |
| **Paratronics 100A Logic Analyzer Kit** | $235.00 |
| Model 10 Trigger Expander Kit | $229.00 |
| Model 150 Bus Grabber Kit | $369.00 |
| **Sinclair 3½ Digit Multimeter** | **$59.95** |
| **Clock Calendar Kit** | **$23.95** |

| TRANSFORMERS | |
|---|---|
| 6V 300 ma | 3.25 |
| 12 Volt 300 ma transformer | 1.25 |
| 12.6V CT 600 ma | 3.75 |
| 12V 250 ma wall plug | 2.95 |
| 12V CT 250 ma wall plug | 3.50 |
| 24V CT 400 ma | 3.95 |
| 10V 1.2 amp wall plug | 4.85 |
| 12V 6 amp | 12.95 |

| DISPLAY LEDS | | | |
|---|---|---|---|
| MAN1 | CA | .270 | 2.90 |
| MAN3 | CC | .125 | .39 |
| MAN72/74 | CA/CA | .300 | 1.00 |
| DL704 | CC | .300 | 1.25 |
| DL707/DL707R | CA | .300 | 1.00 |
| DL727/728 | CA/CC | .500 | 1.90 |
| DL747/750 | CA/CC | .600 | 1.95 |
| DL750 | CC | .600 | 1.95 |
| FND359 | CC | .357 | .70 |
| FND500/507 | CC/CA | .500 | 1.35 |
| FND503/510 | CC/CA | .500 | .90 |
| FND800/807 | CC/CA | .800 | 2.20 |
| 3 digit Bubble | | | .60 |
| 4 digit bubble | | | .80 |
| DG8 Fluorescent | | | 1.75 |
| DG10 Fluorescent | | | 1.75 |
| 5 digit 14 pin display | | | 1.00 |
| NSN69 9 digit display | | | .60 |
| 7520 Clairex photocells | | | .39 |
| TIL311 Hex | | | 9.50 |

| COMPUTER GRADE CAPS | | | |
|---|---|---|---|
| 1600 | mfd | 200V | 4.50 |
| 2000 | mfd | 45V | 2.50 |
| 3200 | | 50V | 2.50 |
| 5500 | | 25V | 2.50 |
| 5800 | | 40V | 3.00 |
| 6100 | | 40V | 3.00 |
| 6100 | | 50V | 3.00 |
| 7700 | | 50V | 2.50 |
| 8000 | | 55V | 2.50 |
| 10000 | | 20V | 2.50 |
| 12000 | | 30V | 3.50 |
| 35000 | | 25V | 5.00 |
| 55000 | | 25V | 5.00 |
| 82000 | | 15V | 6.00 |
| 60000 | | 40V | 6.00 |
| 76000 | | 40V | 6.00 |
| 120000 | | 10V | 6.00 |

### Auto Clock Kit $17.95

DC clock with 4-.50" displays. Uses National MA-1012 module with alarm option. Includes light dimmer, crystal timebase PC boards. Fully regulated, comp. instructs. Add $3.95 for beautiful dark gray case. Best value anywhere.

### RCA Cosmac VIP Kit $229.00

Video computer with games and graphics. Fully assem. and test. **$249.00**
All VIP options avail. week deliv.

### Not a Cheap Clock Kit $14.95

Includes everything except case. 2-PC boards. 6-.50" LED Displays. 5314 clock chip, transformer, all components and full instructions. Orange displays also avail. **Same kit w/.80" displays.** Red only. **$21.95** Case **$11.75**

### 60 Hz Crystal Time Base Kit $4.40

Converts digital clocks from AC line frequency to crystal time base. Outstanding accuracy. Kit includes: PC board, IC, crystal, resistors, capacitors and trimmer.

### Digital Temp. Meter Kit $39.95

Indoor and outdoor. Switches back and forth. Beautiful. 50" LED readouts. Nothing like it available. Needs no additional parts for complete, full operation. Will measure −100° to +200°F, tenths of a degree, air or liquid. Beautiful woodgrain case w/bezel **$11.75**

### NiCad Battery Fixer/Charger Kit

Opens shorted cells that won't hold a charge and then charges them up, all in one kit w/full parts and instructions. **$7.25**

### PROM Eraser

Will erase 25 PROMs in 15 minutes. Ultraviolet, assembled **$34.50**

### Rockwell AIM 65 Computer

6502 based single board with full ASCII keyboard and 20 column thermal printer. 20 char. alphanumeric display, ROM monitor, fully expandable. $375.00. 4K version $450.00. 4K Assembler $85.00, 8K Basic Interpreter $100.00. Power supply assem. in case $60.00. AIM 65 in thin briefcase with power supply $485.00.

### Multi-volt Computer Power Supply

8v 5 amp, ±18v .5 amp, 5v 1.5 amp, −5v .5 amp, 12v .5 amp, −12 option. ±5v, ±12v are regulated. Kit $29.95. Kit with punched frame $37.45. Woodgrain case $10.00.

### Video Modulator Kit $8.95

Convert your TV set into a high quality monitor without affecting normal usage. Complete kit with full instructions.

### 2.5 MHz Frequency Counter Kit

Complete kit less case **$37.50**

### 30 MHz Frequency Counter Kit

Complete kit less case **$47.75**
Prescaler kit to 350 MHz **$19.95**

### 79 IC Update Master Manual $35.00

Complete IC data selector, 2500 pg. master reference guide. Over 50,000 cross references. Free update service through 1979. Domestic postage $3.50. Foreign $5.00. 1978 IC Master closeout $19.50.

### Stopwatch Kit $26.95

Full six digit battery operated. 2–5 volts. 3.2768 MHz crystal accuracy. Times to 59 min., 59 sec., 99 1/100 sec. Times std., split and Taylor. 7205 chip, all components minus case. Full instructions.

### Hickok 3½ Digit LCD Multimeter

Batt/AC oper. 0.1mv–1000v. 5 ranges. 0.5% accur. Resistance 6 low power ranges 0.1 ohm–20M ohm. DC curr. .01 to 100ma. Hand held, ½" LCD displays, auto zero, polarity, overrange. **$69.95.**

### S-100 Computer Boards

| | |
|---|---|
| 8K Static RAM Kit Godbout | $135.00 |
| 16K Static RAM Kit | 265.00 |
| 24K Static RAM Kit | 423.00 |
| 32K Dynamic RAM Kit | 310.00 |
| 64K Dynamic RAM Kit | 470.00 |
| 8K/16K Eprom Kit (less PROMS) | **$89.00** |
| Video Interface Kit | **$139.00** |
| Motherboard **$39.** Extender Board | **$8.99** |

**TERMS: $5.00 min. order U.S. Funds. Calif residents add 6% tax.**
**BankAmericard and Master Charge accepted.**
**Shipping charges will be added on charge cards.**

**FREE: Send for your copy of our NEW 1979 QUEST CATALOG. Include 28¢ stamp.**

Circle 126 on inquiry card.

# The DATA-TRANS 1000

## A completely refurbished **IBM** Selectric Terminal with built-in **ASCII** Interface.

# $1395

**Features:**

- 300 Baud
- 14.9 characters per second printout
- Reliable heavy duty Selectric mechanism
- RS-232C Interface
- Documentation included
- 60 day warranty - parts and labor
- High quality Selectric printing Off-line use as typewriter
- Optional tractor feed available
- 15 inch carriage width

**HOW TO ORDER DATA-TRANS 1000**

1. We accept Visa, Master Charge. Make cashiers checks or personal check payable to:
**DATA-TRANS**
2. All orders are shipped F.O.B. San Jose, CA
3. Deliveries are immediate

For orders and information

**DATA-TRANS**
2154 O'Toole St.
Unit E
San Jose, CA 95131
Phone: (408) 263-9246

**MICRO-PROCESSORS: FROM CHIPS TO SYSTEMS**

This book cover all aspects of microprocessors, from the basic concepts to advanced interfacing techniques, in a progressive presentation. It is independent from any manufacturer, and presents uniform **standard** principles and design techniques, including the interconnect of a **standard** system, as well as specific components. It introduces the MPU, how it works internally, the system components (ROM, RAM, UART, PIO, others), the system interconnect, applications, programming, and the problems and techniques of system development. By R. Zaks. SYBEX. Ref. C201. $9.95

**MICRO-PROCESSOR INTERFACING TECHNIQUES**

Microprocessor interfacing is no longer an art. It is a set of techniques, and in some cases just a set of components. This comprehensive book introduces the basic interfacing concepts and techniques, then presents in detail the implementation details, from hardware to software. It covers all the essential peripherals, from keyboard to floppy disk, as well as the standard buses (S100 to IEEE 488) and introduces the basic troubleshooting techniques. (2nd Expanded Edition). By Austin Lesea and R. Zaks. Ref. C207 SYBEX. $11.95

**PROGRAMMING THE 6502 PROGRAMMING THE Z80 PROGRAMMING THE 8080***

It covers all essential aspects of programming, as well as the advantages and disadvantages of the 6502 and should bring the reader to the point where he can start writing complete applications programs. For the reader who wishes more, a companion volume is available: The 6502 Applications Book. By R. Zaks. 6502: Ref. C202; Z80: Ref. C280; 8080: Ref. C208. SYBEX. Each $10.95

**44 BUS MOTHER BOARD**

Has provisions for ten 44 pin (.156) connectors, spaced 3/4 of an inch apart. Pin 20 is connected to X, and 22 is connected to Z for power and ground. All the other pins are connected in parallel. This board also has provisions for bypass capacitors. Board cost $15.00 Part No. 102. Connectors $3.00 each Part No. 44WP.

**AN INTRODUCTION TO PERSONAL AND BUSINESS COMPUTING**

No computer background is required. The book is designed to educate the reader in all the aspects of a system, from the selection of the microcomputer to the required peripherals. By Rodnay Zaks. Ref. C200, SYBEX $6.95

**TVT COOKBOOK**

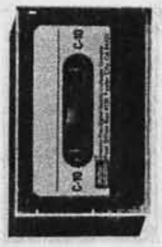
Bk 1064 — by Don Lancaster. Describes the use of a standard television receiver as a microprocessor CRT terminal. Explains and describes character generation, cursor control and interface information in typical, easy-to-understand Lancascaster style. $9.95

**COMPUTER PROGRAMMING HANDBOOK**

A complete guide to computer programming & data processing. Includes many worked-out examples. By Peter Staak, TAB $9.95

**DIGITAL CASSETTE**

5 min. each side. Box of 10 $9.95. Part No. C-5.



**To Order:** Mention part no. description, and price. In USA shipping paid by us for orders accompanied by check or money order. We accept C.O.D. orders in the U.S. only, or a VISA or Master Charge no., expiration date, signature, phone no., shipping charges will be added. CA residents add 6.5% for tax. Outside USA add 10% for air mail postage and handling. Payment must be in U.S. dollars. Dealer inquiries invited. 24 hour order line (408) 226-4064.

Send for FREE Catalog . . . a big self-addressed envelope with 41¢ postage gets it fastest!

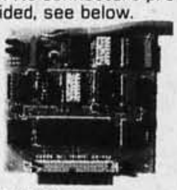
**ELECTRONIC SYSTEMS** Dept. B, P. O. Box 21638, San Jose, CA USA 95151

Circle 125 on inquiry card.

Circle 125 on inquiry card.

## TRS-80 E.S. SERIAL I/O

• Can input into basic • Can use LLIST and LPRINT to output, or output continuously • RS-232 compatible • Can be used with or without the expansion bus • On board switch selectable baud rates of 110, 150, 300, 600, 1200, 2400, parity or no parity odd or even, 5 to 8 data bits, and 1 or 2 stop bits. D.T.R. line • Requires +5, -12 VDC • Board only $19.95 Part No. 8010, with parts $59.95 Part No. 8010A, assembled $79.95 Part No. 8010 C. No connectors provided, see below.

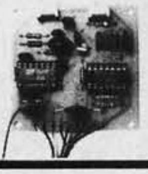


EIA/RS-232 connector Part No. DB25P $6.00, with 9', 8 conductor cable $10.95 Part No. DB25P9.

3' ribbon cable with attached connectors to fit TRS-80 and our serial board $19.95 Part No. 3CAB40.

## MODEM

• Type 103 • Full or half duplex • Works up to 300 baud • Originate or Answer • No coils, only low cost components • TTL input and output-serial • Connect 8 Ω speaker and crystal mic. directly to board • Uses XR FSK demodulator • Requires +5 volts • Board only $7.60 Part No. 109, with parts $27.50 Part No. 109A

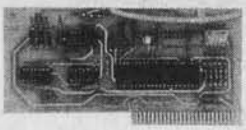


## DISKETTES



Box of 10, 5" $29.95, 8" $39.95.
Plastic box, holds 10 diskettes, 5" - $4.50, 8" - $6.50.

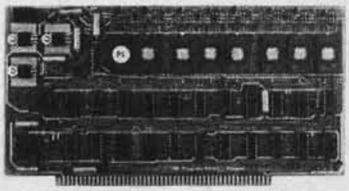
## APPLE II E.S. SERIAL I/O INTERFACE



Baud rate is continuously adjustable from 0 to 30,000 • Plugs into any peripheral connector • Low current drain. RS-232 input and output • On board switch selectable 5 to 8 data bits, 1 or 2 stop bits, and parity or no parity either odd or even • Jumper selectable address • SOFTWARE • Input and Output routine from monitor or BASIC to teletype or other serial printer • Program for using an Apple II for a video or an intelligent terminal. Also can output in correspondence code to interface with some selectrics. • Also watches DTR • Board only $15.00 Part No. 2, with parts $42.00 Part No. 2A, assembled $62.00 Part No. 2C

## 8K EPROM PIICEON

Saves programs on PROM permanently (until erased via UV light) up to 8K bytes. Programs may be directly run from the program saver such as fixed routines or assemblers. • S-100 bus compatible • Room for 8K bytes of EPROM non-volatile memory (2708's). • Onboard PROM programming • Address relocation of each 4K of memory to any 4K boundary within 64K • Power on jump and reset jump option for "turnkey" systems and computers without a front panel • Program saver software available • Solder mask both sides • Full silkscreen for easy assembly. Program saver software in 1 2708 EPROM $25. Bare board $35 including custom coil, board with parts but no EPROMS $139, with 4 EPROMS $179, with 8 EPROMS $219.

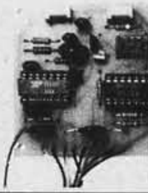


## T.V. TYPEWRITER

• Stand alone TVT • 32 char/line, 16 lines, modifications for 64 char/line included • Parallel ASCII (TTL) input • Video output • 1K on board memory • Output for computer controlled curser • Auto scroll • Non-destructive curser • Curser inputs: up, down, left, right, home, EOL, EOS • Scroll up, down • Requires +5 volts at 1.5 amps, and -12 volts at 30 mA • All 7400, TTL chips • Char. gen. 2513 • Upper case only • Board only $39.00 Part No. 106, with parts $145.00 Part No. 106A

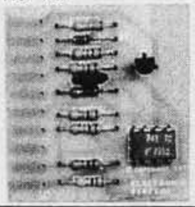
## TAPE INTERFACE

• Play and record Kansas City Standard tapes • Converts a low cost tape recorder to a digital recorder • Works up to 1200 baud • Digital in and out are TTL-serial • Output of board connects to mic. in of recorder • Earphone of recorder connects to input on board • No coils • Requires +5 volts, low power drain • Board only $7.60 Part No. 111, with parts $27.50 Part No. 111A



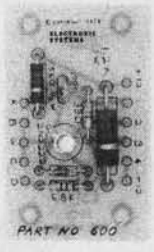
## RS-232/ TTL INTERFACE

• Converts TTL to RS-232, and converts RS-232 to TTL • Two separate circuits • Requires -12 and +12 volts • All connections go to a 10 pin gold plated edge connector • Board only $4.50 Part No. 232, with parts $7.00 Part No. 232A 10 Pin edge connector $3.00 Part No. 10P



## RS-232/TTY INTERFACE

This board has two active circuits, one converts RS-232 to 20mA, and the other converts 20mA to RS-232. Requires +12 and -12 volts. Board only $4.50 Part No. 600, with parts $7.00 Part No. 600A.





## WAMECO PRODUCTS WITH ELECTRONIC SYSTEMS PARTS

**FDC-1** FLOPPY CONTROLLER BOARD will drive shugart, pertek, remex 5" & 8" drives up to 8 drives, on board PROM with power boot up, will operate with CPM (not included). PCBD ............ $42.95

**FPB-1** Front Panel. (Finally) IMSAI size hex displays. Byte or instruction single step. PCBD ............ $42.95

**MEM-1A** 8Kx8 fully buffered, S-100, uses 2102 type RAMS. PCBD ............ $24.95, $168 Kit

**QMB-12** MOTHER BOARD, 13 slot, terminated, S-100 board only ....... $34.95, $89.95 Kit

**CPU-1** 8080A Processor board S-100 with 8 level vector interrupt PCBD .. $25.95, $89.95 Kit

**RTC-1** Realtime clock board. Two independent interrupts. Software programmable. PCBD ............ $25.95, $60.95 Kit

**EPM-1** 1702A 4K EPROM card PCBD ............ $25.95, $49.95 with parts less EPROMS

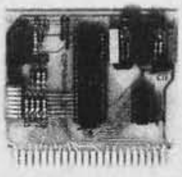
**EPM-2** 2708/2716 16K/32K EPROM card PCBD ........... $24.95, $49.95 with parts less EPROMS

**QMB-9** MOTHER BOARD. Short Version of QMB-12. 9 Slots PCBD ....... $30.95, $67.95 Kit

**MEM-2** 16Kx8 Fully Buffered 2114 Board PCBD ............ $25.95, $269.95 Kit

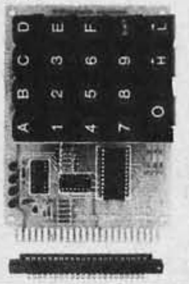
## UART & BAUD RATE GENERATOR

• Converts serial to parallel and parallel to serial • Low cost on board baud rate generator • Baud rates: 110, 150, 300, 600, 1200, and 2400 • Low power drain +5 volts and -12 volts required • TTL compatible • All characters contain a start bit, 5 to 8 data bits, 1 or 2 stop bits, and either odd or even parity. • All connections go to a 44 pin gold plated edge connector • Board only $12.00 Part No. 101, with parts $35.00 Part No. 101A, 44 pin edge connector $4.00 Part No. 44P



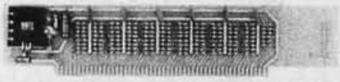
## HEX ENCODED KEYBOARD E.S.

This HEX keyboard has 19 keys, 16 encoded with 3 user definable. The encoded TTL outputs, 8-4-2-1 and STROBE are debounced and available in true and complement form. Four onboard LEDs indicate the HEX code generated for each key depression. The board requires a single +5 volt supply. Board only $15.00 Part No. HEX-3, with parts $49.95 Part No. HEX-3A. 44 pin edge connector $4.00 Part No. 44P.



## S-100 BUS ACTIVE TERMINATOR

Board only $14.95 Part No. 900, with parts $24.95 Part No. 900A



## DC POWER SUPPLY

• Board supplies a regulated +5 volts at 3 amps., +12, -12, and -5 volts at 1 amp. • Power required is 8 volts AC at 3 amps., and 24 volts AC C.T. at 1.5 amps. • Board only $12.50 Part No. 6085, with parts excluding transformers $42.50 Part No. 6085A

**To Order:** Mention part no. description, and price. In USA shipping paid by us for orders accompanied by check or money order. We accept C.O.D. orders in the U.S. only, or a VISA or Master Charge no., expiration date, signature, phone no., shipping charges will be added. CA residents add 6.5% for tax. Outside USA add 10% for air mail postage and handling. Payment must be in U.S. dollars. Dealer inquiries invited. 24 hour order line (408) 226-4064.

Send for FREE Catalog . . . a big self-addressed envelope with 41¢ postage gets it fastest!

# ELECTRONIC SYSTEMS Dept. B, P. O. Box 21638, San Jose, CA USA 95151

Circle 312 on inquiry card.

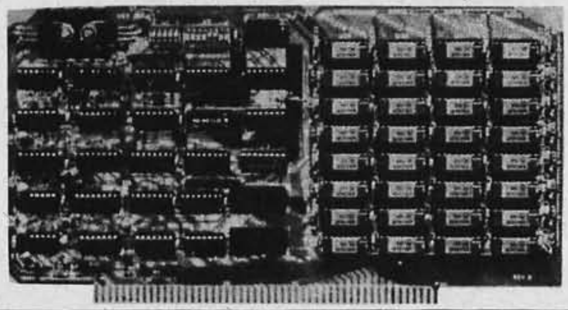
# Visit our new retail location!

Visit our new retail location! Visit our new retail location!

Visit our new retail location! Visit our new retail location!

## SD EXPANDORAM

*The Ultimate S-100 Memory*



The EXPANDORAM is available in versions from 16K up to 64K, so for a minimum investment you can have a memory system that will grow with your needs. This is a dynamic memory with the invisable on-board refresh, and IT WORKS!

- Interfaces with Altair, IMSAI, SOL-8, Cromenco, SBC-100, and others.
- Bank Selectable
- Phantom
- Power 8VDC, ± 16VDC, 5 Watts
- Lowest Cost Per Bit
- Uses Popular 4116 RAMS
- PC Board is doubled solder masked and has silk-screen parts layout.

- Extensive documentation clearly written
- Complete Kit includes all Sockets for 64K
- Memory access time: 375ns, Cycle time: 500ns.
- No wait states required.
- 16K boundries and Protection via Dip Switches
- Designed to work with Z-80, 8080, 8085 CPU's.

**EXPANDO 64 KIT (4116)**

| | |
|---|---|
| 16K | $249 |
| 32K | $324 |
| 48K | $399 |
| 64K | $474 |

## DISC DRIVES

Sugart SA400 5¼"
with attractive metal case
$ 325

*Sugart 801*
with attractive metal case
**$495.00**

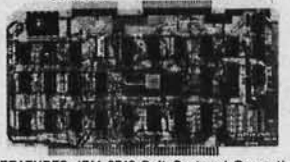
*Siemens FDD 200-8 8" double-sided double density*

***$650.00***

## DISC CONTROLLER

**SD "VERSAFLOPPY" Kit**

The Versatile Floppy Disk Controller — **Only $159^00**



FEATURES: IBM 3740 Soft Sectored Compatible. S-100 BUS Compatible for Z-80 or 8080. Controls up to 4 Drives (single or double sided). Directly controls the following drives:
1. Shugart SA400/450 Mini Floppy
2. Shugart SA800/850 Standard Floppy.
3. PERSCI 70 and 277.
4. MFE 700/750.
5. CDC 9404/9406.
6. GSI/Siemans FDD120-8.

34 Pin Connector for Mini Floppy. 50 Pin Connector for Standard Floppy. Operates with modified CP/M operating system and C-Basic Comptler. The new "Versafloppy" from S.D. Computer Products provides complete control for many of the available Floppy Disk Drives, Both Mini and Full Size. FD1771B-1 Single Density Controller Chip. Listings for Control Software are included in price.

CPM for SD Versafloppy $100^00

## *SAVE $100^00*

## DM2700S DISK & CABINET with POWER SUPPLY

DM2700S includes Siemans or Shugart Disk Drive with the following features:

- Single or Double Density
- Hard or Soft Sector
- Write Protect
- Hard Sector Detection
- 500 KB/S Transfer
- 800 KB unformated
- Bit density 6536 BP1
- Sugart 800 Series Compatable

Cabinet includes:
- 110V to 125V 60 Hz power supply
- Data Cable
- Fan
- Accepts per SCI, Shugart, Siemans 8" Drives

SHOP & COMPARE!

**DM2700S** Disk Drive & Cabinet **REG. $750 SALE PRICED $650^00**

**DM2700** Cabinet, less Drive ~~$249^00~~ **$225^00**

**SPECIAL:** SD Versafloppy Kit, CPM, and DM2700S **$888**

CONTINENTAL SPECIALTIES CORPORATION

CSC

## Logic Probes and Digital Pulsers

SALE

SALE

**LOGIC PROBES**

CSC logic probes are the ultimate tool for breadboard design and testing. These hand-held units provide an instant overview of circuit conditions. Simple to use; just clip power leads to circuit's power supply, set logic family switch to TTL/DTL or CMOS/HTL. Touch probe to test node. Trace logic levels and pulses through digital circuits. Even stretch and latch for easy pulse detection. Instant recognition of high, low or invalid levels, open circuits and nodes. Simple, dual-level detector LEDs tell it quickly, correctly. HI (Logic "1"); LO (Logic "0"). Also incorporates blinking pulse detector. e.g., HI and LO LEDs blink on or off, tracking "1" or "0" states at square wave frequencies up to 1.5 MHz. Pulse LED blinks on for ⅓ second during pulse transition. Choice of three models to meet individual requirements; budget, project and speed of logic circuits.

**MODEL LP-1**

Hand-held logic probe provides instant reading of logic levels for TTL, DTL, HTL or CMOS. **Input Impedance:** 100,000 ohms. **Minimum Detectable Pulse:** 50 ns. **Maximum Input Signal (Frequency):** 10 MHz. **Pulse Detector (LED):** High speed train or single event. **Pulse Memory:** Pulse or level transition detected and stored.

**CSC Model LP-1 Logic Probe**—Net Each ............ ~~$44.95~~ **$42.70**

**MODEL LP-2**

Economy version of Model LP-1. Safer than a voltmeter. More accurate than a scope. **Input Impedance:** 300,000 ohms. **Minimum Detectable Pulse:** 300 ns. **Maximum Input Signal (Frequency):** 1.5 MHz. **Pulse Detector (LED):** High speed train or single event. **Pulse Memory:** None.

**CSC Model LP-2 Logic Probe**—Net Each ............ ~~$24.95~~ **$23.70**

**MODEL LP-3**

High speed logic probe. Captures pulses as short as 10 ns. **Input Impedance:** 500,000 ohms. **Minimum Detectable Pulse:** 10 ns. **Maximum Input Signal (Frequency):** 50 MHz. **Pulse Detector (LED):** High speed train or single event. **Pulse Memory:** Pulse or level transition detected and stored.

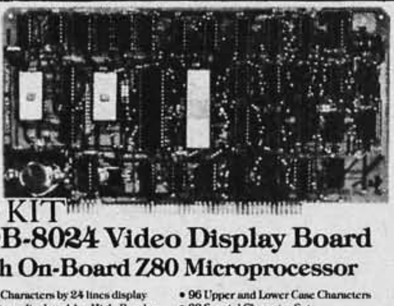
**CSC Model LP-3 Logic Probe**—Net Each ............ ~~$69.95~~ **$66.45**

**DIGITAL PULSER**

The ultimate in speed and ease of operation. Simply connect clip leads to positive and negative power, then touch DP-1's probe to a circuit node; automatic polarity sensor detects circuit's high or low condition. Depress the pushbutton and trigger an opposite polarity pulse into the circuit. Fast troubleshooting includes injecting signals at key points in TTL, DTL, CMOS or other popular circuits. Test with single pulse or 100 pulses per second via built-in dual control push-button; button selects single shot or continuous modes. LED indicator monitors operating modes by flashing once for single pulse or continuously for a pulse train. Completely automatic, pencil-size lab/field pulse generator for any family of digital circuits. **Output:** Tri-state. **Polarity:** Pulse-sensing auto-polarity. **Sync and Source:** 100 mA. **Pulse Train:** 100 pps. **LED Indicator:** Flashes for single pulse; stays lit for pulse train.

**CSC Model DP-1 Digital Pulser**—Net Each ............ ~~$74.95~~ **$71.20**

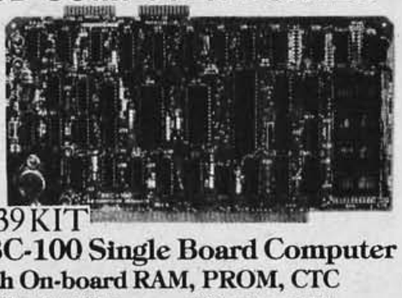
## SD COMPUTER BOARDS



$319 KIT

### VDB-8024 Video Display Board With On-Board Z80 Microprocessor

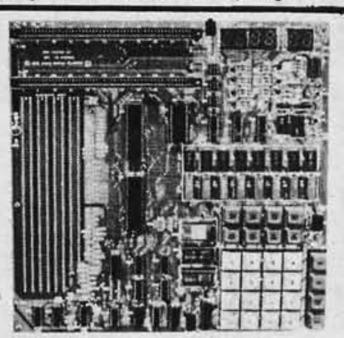
- Full 80 Characters by 24 lines display
- Characters displayed by High Resolution 7×10 Matrix
- Keyboard Power and Interface
- Composite Video Output
- Separate TTL Level Synchronization and Video Outputs
- 2K Bytes Independent On Board Memory
- On-Board Z80 Microprocessor
- Glich Free Display
- 96 Upper and Lower Case Characters
- 32 Special Character Set
- 128 Additional user Programmable Characters
- Full Cursor Control
- Forward and Reverse Scrolling Capability
- Operates as an Independent Terminal
- Variable Speed Display Rate
- Blinking, Underlining, Field Reverse, Field Protect and Combinations



$239 KIT

### SBC-100 Single Board Computer with On-board RAM, PROM, CTC

- Four Channel Counter/Timer (Z80-CTC)
- Software Programmable Baud Rate Generator
- S-100 Bus Compatible
- No Front Panel Required for Operation
- Optional Vectored Interrupts
- Z80Central Processing Unit
- 1024 Bytes of Random Access Memory
- 8K Bytes of Available PROM
- Serial Input/Output Port with both Synchronous and Asynchronous Operation
- Parallel Input and Output Ports



$249

### Z80 Starter Kit A Complete Microcomputer on a Board

- Z80 Central Processing Unit with 158 Instructions
- On Board Keyboard and Display
- Kansas City Standard Cassette Interface
- PROM Programmer Built on-board
- Expansion provision for two S-100 Connectors
- Wire Wrap area for custom circuitry
- Single 5 Volt Operation when not programming
- 1K Bytes of RAM (Expandable to 2K Bytes)
- 1K Bytes of RAM (Expandable to 2K Bytes)
- 4 Channel Hardware Counter/Timer (Z80-CTC)
- Two Bi-directional 8-bit I/O Ports (Z80-PIO)
- Switch Selectable PROM or Monitor Restart
- 2K Byte ZBUG Monitor in ROM
- Memory Examine and Change
- Port Examine and Change
- Z80 CPU Register Examine and Change
- Up to 5 Programmable Breakpoints
- Single Step through RAM or PROM
- Audio Cassette Load and Dump
- Vectored Interrupts provided by Z80-CTC and Z80-PIO
- Ideal for Experimentaion and Evaluating the Z80 CPU

VISA

## PRIORITY ONE ELECTRONICS ©

16723B Roscoe Blvd. Sepulveda, CA 91343

master charge

Terms: Visa, MC, BAC, Check, Money Order, C.O.D. U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Prepaid U.S. orders less than $75.00 include 5% shipping and handling, minimum $2.50. Excess refunded. Just in case . . . please include your phone no.

Prices subject to change without notice.

We will do our best to maintain prices thru August 1979.

**phone orders welcome (213) 894-8171, (800) 423-5633**

**OEM and Institutional inquiries invited.**

**ORDER TOLL FREE 1-800-423-5633 ORDER TOLL FREE 1-800-423-5633**

# Visit our new retail location!

Circle 312 on inquiry card.

Visit our new retail location! — Visit our new retail location! — Visit our new retail location!

**A Portable, Dual Trace Oscilloscope with Big Performance and a Low, Low Price**

***NEW MS-230 Dual Trace Miniscope with 30 MHz Bandwidth!***

**$559^00**

**PORTABLE BATTERY OPERATED • MADE IN THE U.S.A.!**
**With Rechargeable Batteries & Charger Unit**

Non-Linear Systems, Inc.
Progress Since 1952

NLS

ANOTHER MINISCOPE!
NEW

## FEATURES

- Dual Trace—2-channel; separate, chopped or alternate modes.
- 30-megahertz bandwidth. • External and internal trigger.
- Time Base—0.05 microseconds to 0.2 Sec./div—21 settings. • Battery of line operation.
- Line synchronization mode. • Power consumption less than 50W.
- Vertical Gain—0.01 to 50 volts/div—12 settings. Size: 2.9" H x 6.4" W x 8.5" D.
- Weighs only 3.5 lbs with batteries.
- TEST MOST DIGITAL LOGIC CIRCUITS INCLUDING MICROPROCESSORS

From the originators of the Digital Voltmeter, the people who have broken sales and performance records for Oscilloscopes, Non-Linear Systems, comes the MS-230 miniscope.

Non Linear Systems took their engineering and modular construction skills and made a dream a reality, a Dual-Trace 30 MHz miniscope, small enough to fit in most briefcases with room to spare at an affordable price.

## VERTICAL

**Mode:** *CH1, CH2, CH1 & CH2 (Chopped) & CH2 (Alt.) (The following specifications apply to each channel.)*
**Bandwidth:** DC to 30 MHz, ± 3 db @ 3 division deflection. Typical 4 division deflection is obtainable up to 20 MHz.
**Coupling:** AC, DC or ground, switch selectable. Low frequency 3 db point on AC is 3 Hz.
**Rise Time:** Approximately 10 nSec @ 3 division deflection.
**Vertical Input:** 10 mV/div to 50V/div in 12 calibrated ranges. Accuracy is 3% of full scale with vernier in full clockwise position. Vernier provides continuously variable deflection factors between fixed ranges, uncalibrated.
**Input Impedance:** 1 megohm in parallel with 50 pF.
**Maximum Input Voltage:** 250V (DC and Peak AC).

## HORIZONTAL

**Mode:** Internal Time Base or External Horizontal, switch selectable. In the XY mode, vertical input is through CH1 and horizontal input through CH2.
**Time Base:** 0.5u Sec/div to 0.2 Sec/div in 21 calibrated ranges. Accuracy is 3% of full scale with vernier in full clockwise position. Vernier provides continuously variable settings between fixed ranges, uncalibrated.
**Amplifier**
**Bandwidth:** DC to 1 MHz (± 3 db)
**Coupling:** AC, DC or ground, switch selectable. Low frequency point on AC is 3 Hz.
**Deflection Factor:** 10 mV/div to 50V/div in 12 calibrated ranges. The ranges can be calibrated with the CH2 gain control.
**Input Impedance:** 1 megohm in parallel with 50 pF.
**Maximum Input Voltage:** 250V (DC and Peak AC).

## TRIGGER

**Modes:**
Automatic: trigger is disabled, time base free runs.
Internal: In the dual trace modes, the internal trigger source is CH1. External and Line (line not functional when MS-230 operates on batteries.) Input impedance is 1 megohm on External Trigger.
External: Controls function as for internal triggering (1 Megohm input impedence).
Line: Trigger is derived from line frequency when using the battery charger.
**Slope:** + or -, switch selectable.
**Coupling:** AC
**Sensitivity:** Less than 1 div for internal trigger and less than 1 volt for external trigger.
**Level:** Trigger level control permits continuous adjustment of trigger point in all modes except Auto.

## CALIBRATOR:
A square-wave signal of 1 volt p-p is provided. Voltage accuracy is ±5%, frequency is approximately 1 KHz.

## DISPLAY

**Graticule:** 4 x 5 div, each division is 0.25 inch, Viewing area 1.1" H x 1.35" W.

## CRT:
Bluish-white phosphor, medium persistence. CRT uses low power filament for low battery drain.

## POWER SOURCES

**Internal:** Three sealed, rechargeable lead-acid cells. Operating time using fully charged cells is approximately 45 minutes. Charging circuitry is integral and functions when the MS-230 is connected to power line through plug-in transformer (supplied with each instrument). Battery charge time with instrument non-operating is 16 hours.
**External:** Operates continuously from 115 vac source 50 - 400 Hz when connected via plug-in transformer. (230 vac is available) Power consumption from AC line is less than 50 watts.

## ENVIRONMENT

**Operating Temperature:** 0° to 40°C
**Shock and Vibration:** Will withstand normal shock and vibration encountered in commercial shipping and handling.

## PHYSICAL MEASUREMENTS

**Size:** 2.9" H x 6.4" W x 8.5" D. (73.7 mm x 162.6mm x215.9 mm)
**Weight:** 3.5 lbs. (1.59 kg) with batteries.

## FURNISHED ACCESSORIES:
Tilt stand, battery charger, 2 input cables, and 3 miniature banana plugs.

## WARRANTY:
One year parts and labor.
Made in the USA!

**MS-230 with Rechargeable Batteries and Charger**
**$559^00**

### PROBES

Deluxe 10 to 1 probe with 10 megohm input. 100 MHz probe with 4 interchangeable tips: Spring-loaded retractable cover tip, Insulating tip, BNC tip, IC tip, also included cap adjustment tool and zippered vinyl case.

41-141 .................................. **$27.00**

**DELUXE COMBINATION PROBE**

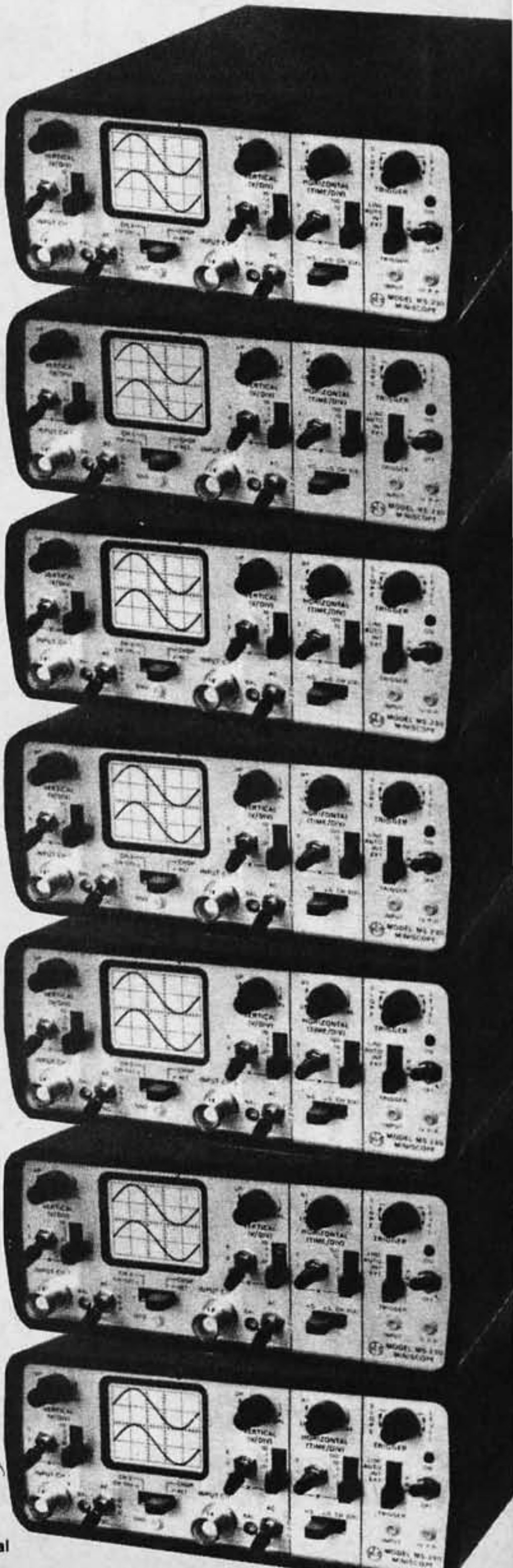
Same as above except the probe has a switch to select; 10 to 1, 1 to 1 or a ground reference position.
**41-37R Red probe body**
**41-37G Grey probe body** .................. **$38.50**

**LEATHER CARRYING CASE**

*The leather case has 2 separate compartments. One to hold the scope, the other to hold the charger, probe, shoulder strap, etc. The case can be worn on the belt, or over the neck.*
*The snaps used on the case are "one way", thus accidental striking of the case against an object will not undo the snaps or let it be pulled off your belt.*

41-180 .................................. $45.00

**75.00 OFF**
on any accessories purchased with MS-230 Miniscope. Just send or mention this COUPON and Byte Magazine.



***PRIORITY ONE ELECTRONICS*** ©
16723B Roscoe Blvd. Sepulveda, CA 91343

Terms: Visa, MC, BAC, Check, Money Order, C.O.D. U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Prepaid U.S. orders less than $75.00 include 5% shipping and handling, minimum $2.50. Excess refunded. Just in case . . . please include your phone no.
Prices subject to change without notice.
We will do our best to maintain prices thru August 1979.

**OEM and Institutional inquiries invited.**

**phone orders welcome (213) 894-8171, (800) 423-5633**

**ORDER TOLL FREE 1-800-423-5633 ORDER TOLL FREE 1-800-423-5633**

Circle 312 on inquiry card.

# HICKOK LX303 $74.95

## HICKOK LX303

**$74.95***

.5%, 3½ digit 19 Range DVM. ½" LCD displays runs 200 hrs on 1 battery. 10 Meg Ohm Input. 1 yr. guarantee, made in U.S.A., test leads included.

**Available Accessories**

- RC-3 115V AC Adapter ........... $7.50
- CC-3 Deluxe Padded Vinyl Carrying Case ............... $7.50
- VP-10 X10 DCV Probe Adapter/ Protector 10Kv .............. $14.95
- VP-40 40Kv DC Probe ........... $35.00
- CS-1 10 Amp Current Shunt ...... $14.95

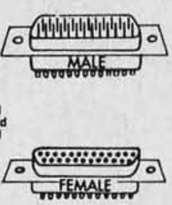
***FREE**

*Just for Asking.*
*FREE BATTERY with your meter.*

## RS232 & "D" TYPE CONNECTORS

P = Plug-Male S = Socket-Female C = Cover-Hood

| PART NO. | DESCRIPTION | PRICE 1-4 | 5-9 | 10-24 |
|---|---|---|---|---|
| DE-9P | 9 Pin Male | 1.50 | 1.30 | 1.10 |
| DE-9S | 9 Pin Female | 1.95 | 1.75 | 1.45 |
| DE-9C | 9 Pin Cover | 1.50 | 1.30 | 1.10 |
| DA15P | 15 Pin Male | 2.00 | 1.80 | 1.55 |
| DA15S | 15 Pin Female | 2.90 | 2.70 | 2.45 |
| DA15C | 15 Pin Cover | 1.80 | 1.60 | 1.30 |
| DB-25P | 25 Pin Male | 2.50 | 2.20 | 2.05 |
| DB-25S | 25 Pin Female | 3.50 | 3.10 | 2.95 |
| DB51212-1 | 1 pc. Grey Hood | 1.65 | 1.35 | 1.20 |
| DB1226-1A | 2 pc. Black Hood | 1.80 | 1.50 | 1.35 |
| DB110963-3 | 2 pc. Grey Hood | 1.70 | 1.40 | 1.25 |
| DC37P | 37 Pin Male | 3.95 | 3.75 | 3.50 |
| DC37S | 37 Pin Female | 5.50 | 5.25 | 4.90 |
| DC37C | 37 Pin Cover | 2.00 | 1.80 | 1.60 |
| DD50P | 50 Pin Male | 5.00 | 4.75 | 4.60 |
| DD50S | 50 Pin Female | 6.50 | 6.00 | 5.75 |
| DD50C | 50 Pin Cover | 2.50 | 2.30 | 2.20 |
| D20418-S | Hardware Set (2 pair) | 1.00 | .80 | .70 |




**Connector for CENTRONICS 700 SERIES:**
Amphenol 57-30360 for back of Centronics 700 Series printers
1-4—$9.00 5-up—$7.50

## 3 LEVEL GOLD WIRE WRAP SOCKETS

Sockets purchased in multiples of 50 per type may be combined for best price.

| | 1-9 | 10-24 | 25-99 | 100-249 | 250-999 |
|---|---|---|---|---|---|
| 8 pin | .40 | .36 | .34 | .31 | .27 |
| 14 pin | .39 | .38 | .36 | .32 | .31 |
| 16 pin | .50 | .42 | .40 | .36 | .34 |
| 18 pin | .70 | .60 | .55 | .50 | .45 |
| 20 pin | .90 | .80 | .75 | .65 | .62 |
| 22 pin | .95 | .85 | .80 | .70 | .65 |
| 24 pin | .95 | .85 | .80 | .70 | .65 |
| 28 pin | 1.25 | 1.15 | 1.00 | .95 | .90 |
| 40 pin | 1.65 | 1.45 | 1.35 | 1.20 | 1.10 |

All sockets are GOLD 3 level closed entry. 2 level Tail, Low Profile, Tin Sockets and Dip Plugs available. CALL FOR QUOTATION.

## S-100 BUS EDGE CONNECTORS

SALE SALE

**S100-WWG** 50/100 Cont. .125 ctrs. 3 LEVEL WIRE WRAP .025" sq. posts on .250 spaced rows. GOLD PLATED.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $4.00 | $3.75 | $3.50 |

**S100-STG** 50/100 Cont. .125 ctrs. DIP SOLDER TAIL on .250 spaced rows for VECTOR, IMSAI, CROMENCO motherboards GOLD plated.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $3.50 | $3.25 | $3.00 |

**S100ALT** 50/100 Cont. .125 ctrs. DIP SOLDER TAIL on .140 spaced rows for ALTAIR motherboards. GOLD plated.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $4.50 | $4.25 | $4.00 |

**S100SE** 50/100 Cont. .125 ctrs. PIERCED SOLDER EYELET tails. GOLD

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $5.00 | $4.50 | $4.00 |

### Other Popular Edge Connectors

**D2244-5SE** 22/44 Cont. .156 ctrs. PIERCED SOLDER EYELET tails. GOLD plated.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $3.00 | $2.60 | $2.20 |

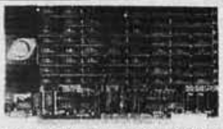
**D2244-5WW** 22/44 Cont. .156 ctrs. WIRE WRAP tails. GOLD.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $3.95 | $3.70 | $3.40 |

CG-1 IMSAI Style Card Guides 5/$1.00

See our July Ad for many other connectors.

## THE MICROBYTE M32KSS 32K STATIC MEMORY BOARD

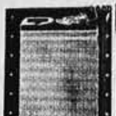


- Fully S100 Bus Compatible, IMSAI, SOL, ALTAIR, ALPHA MICRO
- Uses National's Low Power 5257 4K x 1 Static Rams
- 2 MHz or 4 MHz operation
- Gold contacts for higher reliability
- On board single 5 amp regulator
- Thermally designed heat sink (board operating temperature 0°-70°C)
- Commercially designed power bus. 7 ground bus bars. 0.1 uf decoupling capacitors.
- Fully tri-state buffered
- Inputs fully low power Shottky Schmitt Trigger buffered on all address and data lines.
- Phantom is jumper selectable to pin 67
- Each 4K bank addressable to any 4K slot with in a 64K boundry.
- 4K hardware or software selectable
- One on board 8-bit output port enables or disables the 32K in 4K blocks
- Selectable port address
- 4K banks can be selected or disabled on power on clear or reset
- Will operate with or without front panel
- Compatible with ALPHA MICRO, with extended memory management for selection beyond 64K
- No DMA restriction
- Low power consumption 2.3 — 2.5 amps
- Fully warranted for 120 days from date of shipment.

**32K 2 MHz $619 32K 4MHz $649**

## Vector 8803 MOTHER BOARD FOR S100 BUS MICRO-COMPUTERS

- Kit includes 12 tantalum capacitors for +5, +12, −12 buses and insulated mounting spacers.
- Wiring side shown. Component side bare epoxy glass with white markings for component locations.
- G10 epoxy glass board with 2 ounce copper, solder plated and .038 diameter holes for leads.
- Solder mask with solder windows on etched circuits to avoid accidental short circuits.
- Mounts 11 receptacles with 100 contacts (2 rows) on .125 centers with .250 row spacing. Vector part number R681-2, or mounts 10 receptacles plus interconnections to smaller mother board for expansion.
- Includes etched circuits and instructions for option of active, pull-up, or floating terminations.
- Large buses: +5V and GND (10 AMPS), ±12V or 16V (7 AMPS). Current ratings are per MIL-STD-275 with 10°C rise.
- Fits in Vector-pak enclosures.
- Fits in IMSAI 8080 microcomputer as expander board.



**Price: $29.50**

## Vector Plugboards

**8800V**

Universal Microcomputer/processor plugboard, use with S-100 bus. Complete with heat sink & hardware. 5.3" x 10" x 1/16".

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $19.95 | $17.95 | $15.96 |

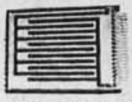
**8801-1**

Same as 8800V except plain; less power buses & heat sink.

| 1-4 | 5-9 | 10-24 |
|---|---|---|
| $15.22 | $13.79 | $12.18 |

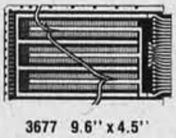
**3682** 9.6" x 4.5" $10.97
**3682-2** 6.5" x 4.5" $9.81

Hi-Density Dual-In-Line Plugboard for Wire Wrap with Power & Grd. Bus Epoxy Glass 1/16" 44 pin con. spaced .156



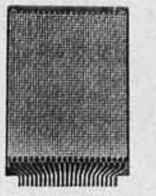
**3677** 9.6" x 4.5" **$10.90**
**3677-2** 6.5" x 4.5" **$9.74**

Gen. Purpose D.I.P. Boards with Bus Pattern for Solder or Wire Wrap. Epoxy Glass 1/16" 44 pin con. spaced .156



**3662** 6.5" x 4.5" **$7.65**
**3662-2** 9.6" x 4.5" **$11.45**

P pattern plugboards for IC's Epoxy Glass 1/16" 44 pin con. spaced .156



3690-12

**CARD EXTENDER**

Card Extender has 100 contacts 50 per side on .125 centers-Attached connector-is compatible with S-100 Bus Systems. $25.83
**3690** 6.5" 22/44 pin .156 ctrs. Extenders .... $13.17

## 1/16 Vector BOARD

**.042 dia holes on 0.1 spacing for IC's**

| Phenolic PART NO. | SIZE | PRICE 1-9 | 10-19 |
|---|---|---|---|
| 64P44XXXP | 4.5x6.5" | $1.56 | $1.40 |
| 169P44XXXP | 4.5x17" | $3.69 | $3.32 |
| **Epoxy Glass** | | | |
| 64P44 | 4.5x6.5" | $1.79 | $1.61 |
| 84P44 | 4.5x8.5" | $2.21 | $1.99 |
| 169P44 | 4.5x17" | $4.52 | $4.07 |
| 169P84 | 8.5x17" | $8.03 | $7.23 |

## TRS-80/APPLE MEMORY EXPANSION KITS

**4116's RAMS**
*from Leading Manufacturers*
(16Kx1 200ns)

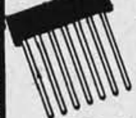
**8 for $75.00**

Add $3.00 for programming Jumpers for TRS-80 Keyboard

## IC SOCKET SALE

| | |
|---|---|
| 14 pin Low Profile | 10/$2.10 **100/$14.00** |
| 16 pin Low Profile | 10/$2.20 **100/$16.00** |
| 24 pin Low Profile | 3/$1.00 **40/$10.00** |
| 40 pin Solder Tail | 3/$1.00 **40/$10.00** |
| 24 pin Dip Plug with cover | 3/$1.00 **40/$10.00** |

## WOW 14 & 16 PIN GOLD 3 LEVEL WIRE WRAP SOCKES



14 - G3 100 for $33.00
16 - G3 100 for $33.00
50 of each for $35.00

## PANAVISE


PANAVISE TILTS, TURNS, AND ROTATES TO ANY POSITION. IT HOLDS YOUR WORK EXACTLY WHERE YOU WANT IT.

## MEMORY MEMORY

| | |
|---|---|
| 2102LIPC Low Power 450ns in lots of 25 | $1.10 |
| 2102AL-2 Low Power 250ns in lots of 25 | $1.25 |
| 2114-4L 1Kx4 450ns Low Power | 8/$45.00 |
| 2114-3L 1Kx4 300 ns Low Power | 8/$50.00 |
| 5257-3L 4Kx1 300ns Low Power | 8/$50.00 |
| 2708 8K 450ns EPROM | $9.00 |
| 2716 16K 5 Volt Only EPROM | $45.00 |

## LEDU

**IM-10A List $89.00**
**SPECIAL**
**$56.95 with tube**

Perfectly balanced fluorescent lighting with precision magnifier lens. Tough thermoplastic shade. Easy lens removal. New wire clip design permits easy installation and removal of fluorescent tube. Comes with plastic shield to protect tube from soiling and damage.

Colors: Gray, Black, and Chocolate Brown. Comes with one 22 watt T-9 Circline fluorescent tube. 3 diopter lens.

## ORDER TOLL FREE

**1-800-423-5633**

except CA., AK., HI., Call (213) 894-8171

## Vector WRAP POST

for .042 dia. holes (all boards on this page)
T44/C pkg. 100 .. $ 2.34
T44/M pkg. 1000 ..... $14.35
A-13 hand installing tool ....... $ 2.94

## PRIORITY ONE ELECTRONICS ©

16723B Roscoe Blvd. Sepulveda, CA 91343

Terms: Visa, MC, BAC, Check, Money Order, C.O.D. U.S. Funds Only. CA residents add 6% sales tax. Minimum order $10.00. Prepaid U.S. orders less than $75.00 include 5% shipping and handling, minimum $2.50. Excess refunded. Just in case . . . please include your phone no.
Prices subject to change without notice.
We will do our best to maintain prices thru August 1979.

**phone orders welcome (213) 894-8171, (800) 423-5633**

OEM and Institutional inquiries invited.

## B+K PRECISION

**TEST EQUIPMENT**
*CALL FOR SPECIAL PRICES*

HICKOK LX303 $74.95 HICKOK LX303 $74.95 HICKOK LX303 $74.95

**ORDER TOLL FREE 1-800-423-5633 ORDER TOLL FREE 1-800-423-5633**

# Unclassified Ads

### NEW UNCLASSIFIED POLICY

*Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed notice to that effect. To be considered for publication, an advertisement must be clearly noncommercial, typed double spaced on plain white paper, contain 75 words or less, and include complete name and address information.*

*These notices are free of charge and will be printed one time only on a space available basis. Notices can be accepted from individuals or bona fide computer users clubs only. We can engage in no correspondence on these and your confirmation of placement is appearance in an issue of BYTE.*

*Please note that it may take three or four months for an ad to appear in the magazine.* ■

FOR SALE: SwTPC MP-A2 processor board (latest), with SWTBUG monitor, $105. Four 4 K programmable memory boards with premium 350 ns low power chips, $65 each. AC-30 cassette interface, $50. All assembled, burned in and running. Prices are 1/3 less than kit. Robert Levine, 32 King St, New York NY 10014, (212) 691-2897 evenings.

FOR SALE: Digital Group 8 K static programmable memory board. In original package, unassembled, complete with all new chips, etc. Best offer over $80 takes. Will guarantee. Bob Waber, 6564 E Michigan Av, Apt 52, Saline MI 48176, (313) 429-7174.

FOR SALE: AMS memory system, 32 K by 16 bits plus parity. Includes rack mount cabinet, power supply, two fans, battery backup, 25 slot dual back plane, no documentation. Originally used with Advanced Memory Systems 2100 processor. Has 66 K bytes of Intersil/AMS 6002 dynamic memory (1024 by 1). $500 plus shipping. Dan S Parker, 1007 3rd St, Davis CA 95616, (916) 758-2341.

FOR SALE: IMSAI 8080 mainframe with 11 edge connectors. Dual serial input/output ports. 8 K static programmable memory with hardware memory protect. 8080 processor board. Documentation and software. ASR33 Teletype with paper tape reader and punch. $1600 or best offer. Rich Levinson, 124 Nashoba Rd, Concord MA 01742, (617) 369-8471.

FOR SALE: Digital Group Z-80 system. Includes dress cabinet, 10 K static memory, TVC-64, I/O-F, 12 A power supply, complete documentation and software including Assembler, MaxiBASIC and diagnostic/demonstration tape. Add a keyboard and monitor and you're ready to run. $850 or will consider reasonable offer. W Colsher, 4328 Nutmeg Ln Apt 111, Lisle IL 60532, (312) 964-1168 anytime.

FOR SALE: An IBM type 1004 used Selectric terminal. It can be converted into a microcomputer Selectric printer with high quality printout. All documents and the reprints of the articles from various journals will be included for free. The first cashier check for $245 will get all. Please send check along with your telephone number to H T Chen, Physics Dept UGA, Athens GA 30602.

FOR SALE: 6502 microcomputer system built on OSI number 400 series boards. Includes 16 K static programmable memory (four number 420c memory boards), TIM monitor (1 K read only memory), 20 mA or RS-232 serial input/output, 8 slot mother board, MOS Technology manuals, wire wrap kludge board, cabinet, oversized power supply and software on paper tape. Software includes BASIC, Editor/Assembler and games. Asking $350 for all this, you ship. Kenneth A Scharf, 34 Royal Crest Dr, Marlboro MA 01752, (617) 481-5534.

FOR SALE: PET Printer adapter with built-in power supply and connectors, $90 complete. GE computer grade caps. 2900 mF, 350 V, $6.50, 850 mF, 450 V, $3.50. All new but surplus. J Gatliff, POB 627, Eau Gallie FL 32935.

FOR SALE: MMD-1 with 256 bytes programmable memory, all documentation plus BUGBOOKS three and five. Excellent tutorial system. $250 plus UPS. Frank DeBolt, 114 Eastpines Rd, Savannah GA 31410, (912) 897-1384.

MUST SELL: New, unused equipment for 8080 or Z-80 based systems with S-100 bus. Purchased before deciding to buy a PET. North Star Micro-Disk System: controller, drive in cabinet with power supply, DOS and BASIC on diskette, documentation, cost $800. North Star Hardware Floating Point Board: greatly speeds up math calculations in BASIC cost $400. Will sacrifice. Karta S Khalsa, 32 The Hollow, Amherst MA 01002, (413) 256-0391.

FOR SALE: Texas Instruments 30 cps Silent 700 printer/keyboard terminal in good condition $380. Teletype BRPE 110 cps PT punch $160. Caelus 2.5 M bytes top loading disk drive $700. Also NOVA extender board. NOVA wire wrap board, mounting slides, AMP and TRW connectors, etc. Nemeth, 560 Upr Mountain Av, Montclair NJ 07043.

FOR SALE: Heathkit H9 video terminal with full documentation, RS-232 input/output scrolling, 110 to 9600 bps, $400 postpaid. Excellent condition. Charles E Zalenski, 9 River Ter, Johnson City NY 13790, (607) 797-5777 days.

FOR SALE: September 1975 thru December 1978 BYTE. Good condition except some response cards used. No missing covers or pages. Sell one or all for best total price (over $99) by end of month this ad printed. High bidders notified, send SASE if response desired otherwise. Jim Matthews, 2028 Merrily Dr, Montgomery AL 36111.

FOR SALE: 64 K plus memory — Interfaced for S100 buss. General Electric 16 K by 40 core memory complete with all cables, power supplies and total documentation. Only $350 plus shipping from Kansas City MO. Jon Smirl, 1927 Orrington Av Apt 8209, Evanston IL 60201, (312) 492-0794. After June 15th the address will be: 5817 Hutson Rd, Kansas City MO 64151, (816) 741-5688.

FOR SALE: Diablo Hytype II (1345WP) word processing printer (with metal wheel), with cover and friction feed platen. Never used. Interfaceable to SOL, 6800 or 8080s. $1599 without power supply. Roger Gersonde, 3011 N Sherman Blvd, Milwaukee WI 53210, (414) 332-9202 day, (414) 445-7880 nights.

FOR SALE: Two 16 K 250 ns TDL static programmable memory boards, one 16 K 250 ns Seattle static programmable memory board, TDL Z-80 processor, SMB board and software (cassette and paper tape), separately or together. All working perfectly; just changing to different system configuration. Barry Gordon, 31 E 31st St, Baltimore MD 21218.

WANTED: Radio Shack TRS-80. Any quantity, any condition. Immediate cash available. Some used units available. Write with description, condition and phone number for immediate quote. DEC PDP-8/E and M modules, RK05, ASR33, RK8E, etc. buy, sell, trade, repair, custom interface. Jim Simpson, POB 632, W Caldwell NJ 07006, evenings (201) 226-9185 or 342-3110.

WANTED: Information on the TC-71 sold by NCE from anybody who has one or has worked on one. Also, have one Radio Shack keyboard for sale, reasonable. Burl E Anderson, 71 Edwards Av, Galesburg IL 61401, (309) 342-5660.

FOR SALE: Altair S-100 bus single drive, single density 8 inch PERTEC floppy disk system with Altair Extended BASIC Version 4.1, read only memory card with Bootstrap read only memory, floppy disk controller boards, cables and complete documentation. Excellent working condition. Reliable. Selling to reconfigure system for hard disk drive. $2800 new. Make offer. Mike Harris, 3750 S Maple Grove Rd, Boise ID 83705, (208) 362-5154.

FOR SALE: Two 4 K by 16 Heath memory boards, $125 each. One H10 paper tape punch with five rolls and three boxes of fanfold tape, $125. Two parallel interface boards, one assembled $130, one unassembled at $85. Digital cassette recorder, $175. James E Tarvid, 2735 N Frederick, Milwaukee WI 53211, (414) 964-8633.

FOR SALE: 16 bit minicomputer, Interdata 5/16 complete on one 10 by 10 inches board with 24 K bytes programmable memory, microprogrammable with monitor in read only memory, Micro-I/O (input/output) buss interfaces with microcomputer style peripherals, ASCII terminal port and Interdata multiplexor buss. Large Interdata software library including BOSS, BASIC, FORTRAN, FFT's, processor and memory tests, etc. Brand new with full documentation, asking $2750. Also, Interdata universal logic interface for I/O, status and control ports with wire wrap area, $300. David Rosenboom, POB 543 Sta Z, Toronto, Ontario CANADA M5N 2Z6, (416) 593-4179.

FOR SALE: Beehive SuperBee II video display terminal. 8008 microprocessor controlled. Scroll mode, page transmit or line transmit. 24 line by 80 character screen, but can hold 200 plus lines in own memory to scroll/page backward and forward. Editing features: line insert/delete, character insert/delete. Function keys. Tabs settable anywhere, may be set by computer. Formatted screen (fill in blanks) can be specified. Truly the Rolls Royce of terminals. $900 or best offer. Michael J Eager, 481 Century Dr, Campbell CA 95008. You must send SASE.

FOR SALE: Heath H8 with 48 K, two each SIO/cassette, interface; $1975. Heath WH17 dual floppy disk system; $925. Heath H9 video terminal; $550. Heath cassette plus recorder; $45. All factory tested and running, some under factory warranty. Reason selling: I have two computer systems. Buy package for $3300 or separately. All offers considered. Ray King, 915 El Rancho, Pocatello ID 83201, (208) 237-0979.

FOR SALE: 32 K static programmable memory factory assembled and tested. Four Industrial Micro 8 K S-100 boards, cost $884 new, asking $650 (ran out of slots). Teletype ASR33 teletypewriter with paper tape reader/punch, stand, $595 plus shipping. Mark Lyon, 6320 Red Prairie Rd, Sheridan OR 97378.

WILL TRADE: Have written programs for Bally HLC with audio cassette interface such as: Checkbook Balancer, Number Sort, Math Quiz, Tic-Tac-Toe, Slot Machine, Hourglass Graphics. I am interested in acquiring other Bally BASIC programs on audio cassette. Chuck Zellers, 2921 Roselawn Dr, Grand Island NE 68801.

FOR SALE: Data Products portable terminal, 10 cps, hard copy, built-in modem and coupler, ASCII/Teletype, RS-232 interface. The first check for $550 will receive this device which is excellent for timesharing or as a microcomputer terminal. Carl Echols, 112 Creekside Ln, Noblesville IN 46060, (317) 849-5247.

FOR SALE OR TRADE: REMEX high speed paper tape reader with stop on character, $150; Burroughs digital cassette drive, $50; 5 V at 70 A power supply, $50. All work fine. Trade any or all for X,Y plotter/recorder. Jim McCord, 330 Vereda Leyenda, Goleta CA 93017, (805) 968-6681.

FOR SALE: DEC MPS microcomputer. Includes 16 K programmable memory plus 4 K bytes eraseable read only memory. Also has vectored interrupt board with parallel input/output. Price $395 plus shipping. Curtis P Hoffman, 169 Millham St, Marlboro MA 01762, (617) 481-7827.

# Reader Service

*To get further information on the products advertised in BYTE, fill out the reader service card with your name and address. Then circle the appropriate numbers for the advertisers you select from the list. Add a 15-cent stamp to the card, then drop it in the mail. Not only do you gain information, but our advertisers are encouraged to use the marketplace provided by BYTE. This helps us bring you a bigger BYTE.*



**Correspond directly with company.*

## BOMB— BYTE's Ongoing Monitor Box

| Article No. | ARTICLE | Page |
|---|---|---|
| 1 | Allen: An Overview of LISP | 10 |
| 2 | Laubsch, Fischer, and Bocker: LISP Based Systems for Education | 18 |
| 3 | Prini and Rudalics: Lambdino Storage Management System | 26 |
| 4 | Kornfeld: Pattern-Directed Invocation Languages | 34 |
| 5 | Ciarcia: Anyone Know the Real Time? | 50 |
| 6 | Albus: A Model of the Brain for Robot Control, Part 3 | 66 |
| 7 | Yeager: Exploring TRS-80 Graphics | 82 |
| 8 | Powers: The Nature of Robots, Part 3 | 94 |
| 9 | Taft: The Design of an M6800 LISP Interpreter | 132 |
| 10 | Pratt: A Mathematician's View of LISP | 162 |
| 11 | Halsema: A Preview of the Motorola 68000 | 170 |
| 12 | Stoutemyer: LISP Based Symbolic Math Systems | 176 |
| 13 | Weyhrauch and Graves: LISP Applications in Boolean Logic | 206 |
| 14 | Gass: An Overview of Long Division | 220 |

### May BOMB Maps a Winner

The May 1979 first place BOMB award of $100 went to William D Johnston for taking a direct route to the top with "Computer Generated Maps," page 10. The second place prize of $50 went to Steve Ciarcia for "Communicate on a Light Beam," page 32. Placing third was "Representing Three Dimensional Objects In Your Computer," page 14 by Richard Blum, with Bob Haas' "Single Chip Video Controller," page 52 taking fourth place.


## Graham-Dorian's Integrated, On-Line Programs Are Fast, Efficient, And Easy To Use.

Rely on Graham-Dorian, a full-line computer software manufacturer, for sophisticated programs — the most detailed on the market today. They're ready to go to work immediately or to be tailored for even more specific needs.

On-line capabilities enable you to make a single entry and update all affected files. An inquiry into a file at any time provides up-to-date information — no batching or sorting of input data.

The programs are easy to use. Messages on the video display guide you each step of the way. Programs make use of indexed sequential and chained files for fast and convenient retrieval of data with efficient use of disk space.

Order in standard eight-inch disk either double or single density, or various mini-floppy formats. Each program contains a free CBASIC-2, plus user's manual and hard copy source listing.

- **Accounts Receivable** — Records invoices, prepares statements and trial balance reports, etc. Automatically reports aging of accounts in periods of 30, 60, and 90 days, with each item listed separately.
- **Accounts Payable** — Vendor lookup and change, entering vendor invoices, writing checks (many options), cash flow analysis, accounts payable check register, and vendor list. Ideal for analyzing expenditures by vendor and by due date.
- **General Ledger** — Includes lookup and change, making journal entries, trial balance, transaction register, chart of accounts, financial statements, and monthly closing.
- **Job Costing** — Provides work order lookup, enters labor transactions, material set-up, progress report of hours, labor distribution report, weekly labor reset, actual versus estimated cost per job.
- **Inventory** — Can be connected with cash register for point of sale inventory control. Number of on-line items limited only by disk space available.
- **Cash Register** — Creates daily sales reports containing information on gift certificates, paidouts, overrings, refunds, and how much in each category a salesperson sold.
- **Payroll** — Handles 100% of all necessary payroll functions including state income tax tables for your state. Ideally suited for both large and small companies.
- **Apartment** — Said one user, "Obviously, this was developed by apartment owners." The package fills virtually all the needs of apartment owners and managers. Ideal for projects with 75 units or more.
- **CBASIC-2** — The most comprehensive and powerful commercially oriented BASIC available today. Enhancements over CBASIC-1: integer variables, multiple line functions, CHAINing with COMMON variables, additional predefined functions, etc. Comes free with any package, or purchase separately.

**Compatible with many computers:** Northstar, IMSAI, Altos, Cromenco, Industrial Micro Systems, Radio Shack TRS-80, SD Systems, Digital Microsystems, Dynabyte DB8/2, Micropolis MOD II, Vector MZ, and other 8080, 8085, and Z-80-based systems.

**See your GDSS dealer or send for information packet and sample runs.**

GDE

**Graham-Dorian Software Systems**
A Division of Graham-Dorian Enterprises
211 N. Broadway / Wichita, KS 67202 / (316) 265-8633

Circle 152 on inquiry card.

# OHIO SCIENTIFIC DOES IT AGAIN

Ohio Scientific has taken its standard Challenger III computer and married it to the new Shugart 29 Megabyte Winchester Drive. The result is the C3-C. This new microcomputer now fills the vacuum that existed for computer users who need more mass storage capability than floppies can offer – yet until now, could not justify the additional cost of a larger capacity hard disk computer such as our C3-B 74 Megabyte disk system.

**Winchester Technology**

Winchester hard disk drives offer small business and professional computer users the logical solution to mass storage problems that are beyond the capability of floppy disks. In addition, Winchester disks feature a track seek-time that is much better than floppies and because they spin at eight times the rate of floppies, Winchesters have a shorter latency. Both of these points reflect one remarkable speed advantage Winchester disks have over floppies.

**Coupled to the Challenger III Computer**

Ohio Scientific's award winning Challenger III computer is a classic. It is the only computer series that utilizes the three most popular microprocessors – 6502A, 68B00 and Z-80. This tremendous processor versatility enables one to utilize a seemingly endless selection of quality programs available from Ohio Scientific's software library as well as from many independent suppliers.

**And Advanced Software**

For instance, there are single user, multi-user and network operating systems. A complete turnkey small business package, OS-AMCAP provides accounts receivable, accounts payable, disbursements, cash receipts, general ledger, etc. OS-CP/M offers a complete FORTRAN and COBOL package. And there is WP-2, a complete word processing system. For information management, OS-DMS, features an advanced file handling system and program library that simplifies information storage and recall and routinely performs tasks which usually require special programming on other systems.

**Yields the Microcomputer of the Future**

With an eye toward the future, the C3-C, like all other Challenger III's was designed with provisions for future generation 16 bit microprocessors via plug-in options.

There are ten open slots for lots of I/O and multi-user operation. Truly, the Ohio Scientific C3-C is a computer with a future.

## The new C3-C computer with 29 Megabyte Winchester Hard Disk.

**$9340 with 48K static RAM and OS-65U operating system!**


**OHIO SCIENTIFIC**

1333 S. CHILLICOTHE RD., AURORA, OHIO 44202 (216) 562-3101

OEM pricing available