

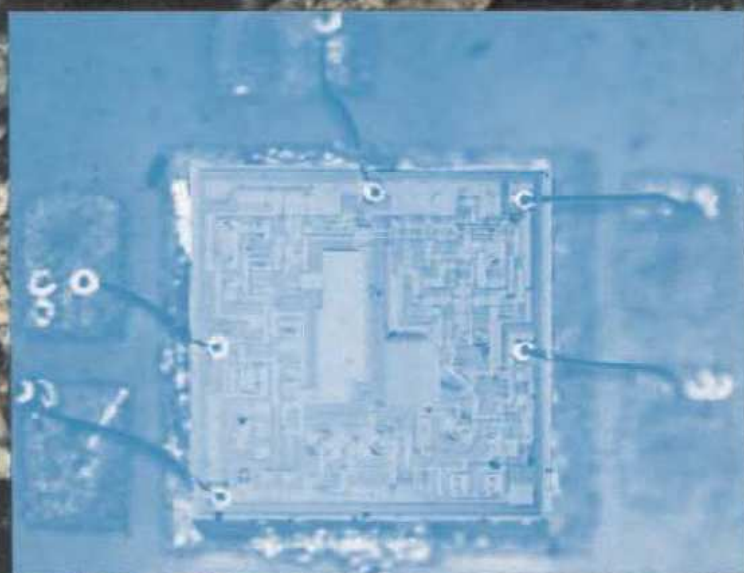
'68'

2.00

MICRO JOURNAL

VOLUME I ISSUE 4 • Devoted to the 6800 User • June 1979
"Small Computers Doing Big Things."

SERVING THE 6800 USERS WORLDWIDE





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.



SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216

(512) 344-0241

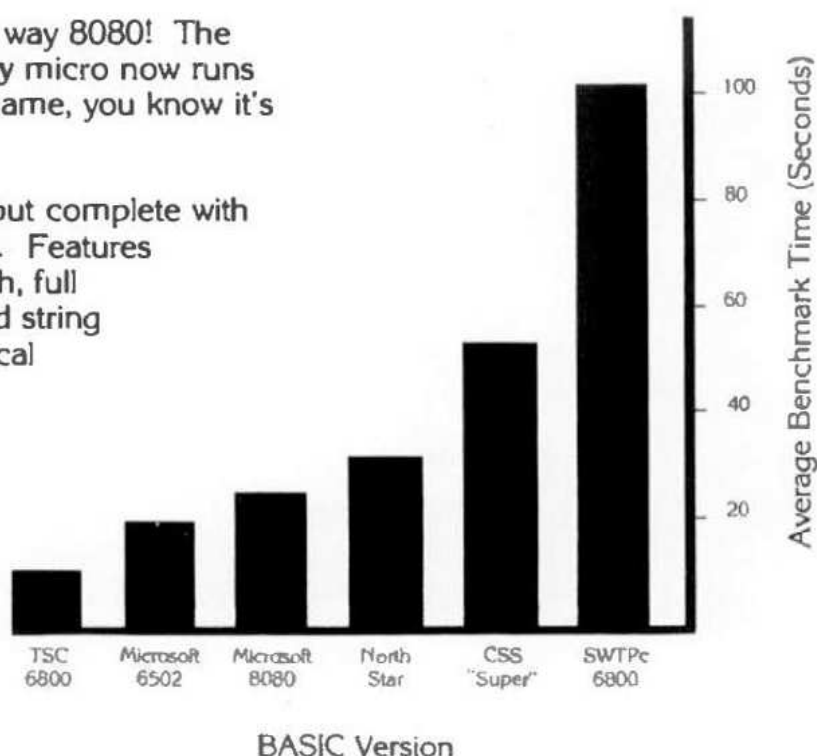
TSC BASIC for 6800

The fastest floating point BASIC for any micro.

Move over 6502! Out of the way 8080! The fastest floating point BASIC for any micro now runs on the 6800. And with the TSC name, you know it's top quality.

TSC BASIC is not only fast, but complete with over 50 commands and functions. Features include six digit floating point math, full transcendental functions, unlimited string length, if/then/else construct, logical operators, and two-dimensional arrays (including string arrays).

Available now on KCS cassette for \$39.95. Requires 9K minimum, no source listing included. Soon to come is a version for the FLEX™ disk operating system.



**Technical Systems
Consultants, Inc.**

Box 2574 W. Lafayette, IN 47906

Specialists in Software & Hardware for Industry & the Hobbyist

Graph based on benchmarks listed in October 1977 issue of Klobaud™ magazine.

'68'

MICRO JOURNAL

Portions of the text of '68' Micro Journal set using the following:
6800/2, DMAF1 and CT-82
Southwest Technical Products Corp.
219 W. Rhapsody
San Antonio, TX 78216

Editor, Word Processor and Sort/Merge
Technical Systems Consultants, Inc.
Box 2574
W. Lafayette, IN 47906 *MINIFLEX & FLEX REG.™
Technical Systems Consultants, Inc.

Selectric I/O
World Wide Electronics, Inc.
130 Northwestern Blvd.
Nashua, NH 03060

Publisher/Editor
Don Williams Sr.

Assistant Editor — Software
Mickey E. Ferguson

Assistant Editor — Hardware
Dennis Womack

Associate Editor — Southwest
Dr. Jack Bryant

Associate Editor — At Large
Dr. Chuck Adams

Associate Editor — Midwest
Howard Berenbon

Subscriptions and Office Manager
Joyce Williams

Typography and Color Separations
Williams Company, Inc.
Chattanooga, TN 37421

CONTENTS

Bryant	6	CRUNCHERS CORNER
	11	LETTERS
Womack	12	LOCATE (UPDATE)
Eagle	12	PROGRAMMING QUICKIES (BASIC)
Ferguson	17	BASIC RENUMBERING (SWTPC)
Dembinski	29	TIME PROMPTS (FLEX)™
Boyd	32	REVIEW OF 8800 MONITORS
	37	WEST COAST COMPUTER FAIRE
Pennington	39	MAGIC SQUARES (BASIC)
	40	EDITORS REMARKS
Schuman	41	BASIC CASSETTE FORMATS

Send All Correspondence To:

'68' Micro Journal
3018 Hamill Rd.
PO Box 849
Hixson, Tennessee 37343
— Phone —
Office: 615-870-1993
Plant: 615-892-7544

Copyright ©

'68' Micro Journal is published 12 times a year by '68' Micro Journal, 6131 Airways Blvd., Chattanooga, TN 37421. Second Class postage paid at Chattanooga, TN. Postmaster: Send Form 3579 to '68' Micro Journal, PO Box 849, Hixson, TN 37343.

Subscription Rates U.S.A.:

(Special Charter Rate)

1 year \$10.50
2 years \$18.50
3 years \$26.50

Lifetime \$125.00 (one-time payment) Twice rate shown above for Air Mail/First Class anywhere in the U.S.A.

—ITEMS SUBMITTED FOR PUBLICATION—

(Letters to the Editor for Publication) All 'letters to the Editor' should be substantiated by facts. Opinions should be indicated as such. All letters must be signed. We are interested in receiving letters that will benefit or alert our readers. Praise as well as gripes is always good subject matter. Your name may be withheld upon request. If you have had a good experience with a 6800 vendor please put it in a letter. If the experience was bad put that in a letter also. Remember, if you tell us who they are then it is only fair that your name 'not' be withheld. This means that all letters published, of a critical nature, cannot have a name withheld. We will attempt to publish 'verbatim' letters that are composed using 'good taste.' We reserve the right to define (for '68' Micro) what constitutes 'good taste.'

(Articles and items submitted for publication) Please, always include your full name, address, and telephone number. Date and number all sheets. TYPE them if you can, poorly handwritten copy is sometimes the difference between go, no-go. All items should be on 8X11 inch, white paper. Most all art work will be reproduced photographically, this includes all listings, diagrams and other non-text material. All typewritten copy should be done with a NEW RIBBON. All hand drawn art should be black on white paper. Please no hand written code items over 50 bytes. Neatly typed copy will be directly reproduced. Column width should be 3¼ inches.

(Advertising) Any Classified: Maximum 20 words. All single letters and/or numbers will be considered one (1) word. No Commercial or Business Type Classified advertising. Classified ads will be published in our standard format. Classified ads \$7.50 one time run, paid in advance.

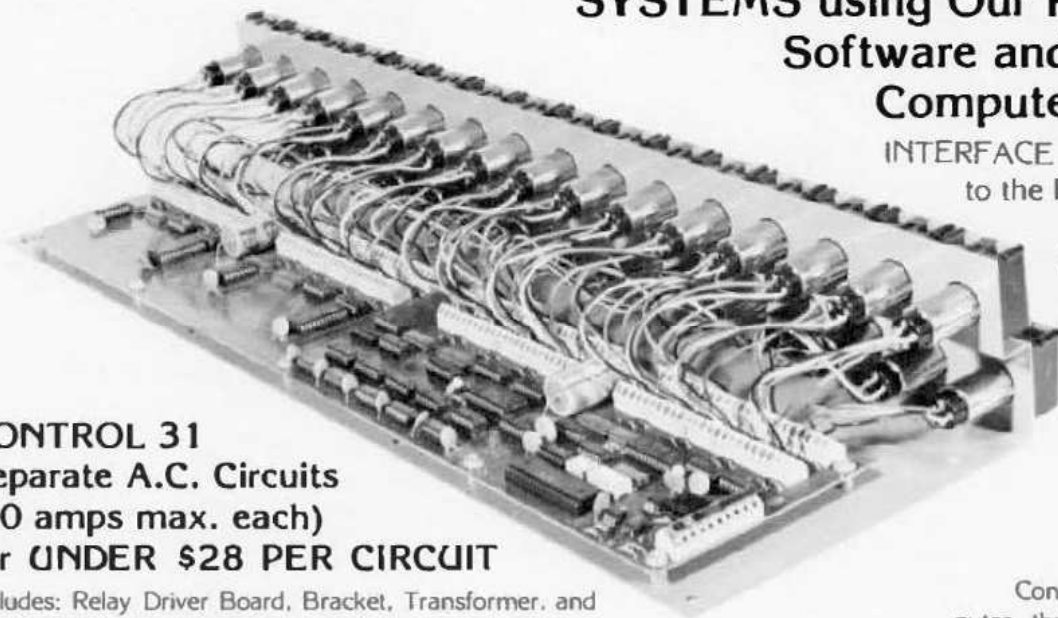
Commercial and/or Business advertisers please write or phone for current rate sheet and publication lag time.



GHOST POWER

BOARDS for ALL COMPUTERS, or COMPLETE TURNKEY SYSTEMS using Our Real Time Software and GHOST Computer System

INTERFACE your Computer
to the Real World with
**GIMIX Relay
Driver Boards.**



**CONTROL 31
Separate A.C. Circuits
(20 amps max. each)
for UNDER \$28 PER CIRCUIT**

Includes: Relay Driver Board, Bracket, Transformer, and
31 GE RR.8 Relays. Assembled and Tested with Documenta-
tion and Software Driver Routines. ***848⁰⁵**

Relay Driver Boards Only

***448⁸⁶**

**INTERFACE the Real World to your Computer with GIMIX
OPTO-BOARDS and REMOTE KEYPAD SYSTEMS:**

OPTO-BOARDS Connect One 8 Bit Parallel I/O Port to as many as
34 Outside World Switch Closures (e.g. doorbells, eyes) with: 1500 Volt Isola-
tion, built-in contact debounce, on board scanning and buffer.

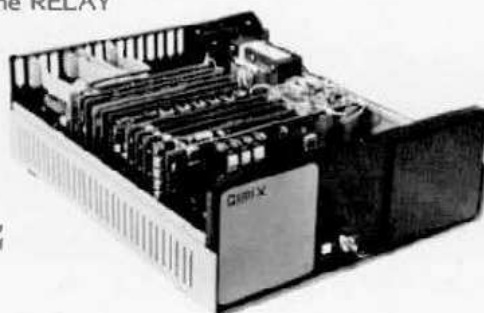
REMOTE KEYPAD SYSTEMS

Provide Remote Data Entry for Computer
Systems or Directly Control One RELAY
DRIVER BOARD or STEREO
SYSTEM CONTROLLER.

**6800-SS 50 BUS
16K SYSTEMS**
from ***1294²⁹**

Includes: Mainframe cabinet, mother board, power
supply, fan, CPU, 16K static RAM, and choice of
I/O card.

Other packages available.



Connects to any Com-
puter through a 20 ma.
current loop (up to 4 Boards 128
Relays per port.) Reports the on/off
status of each relay on demand.

**16K Static RAM Boards
for the SS-50 Bus**

MORE FEATURES:

- Gold bus connectors
- 4 separate 4K Blocks
- Individual Addressing
Write Protect, and
Enable/Disable for each Block
- Tested at 2MHz

**AT A LOWER PRICE
ASSEMBLED**

***298¹³**

As above
with
Sockets
and
Software
control
features.

**ASSEMBLED
*368¹⁶**

THAT'S VALUE!

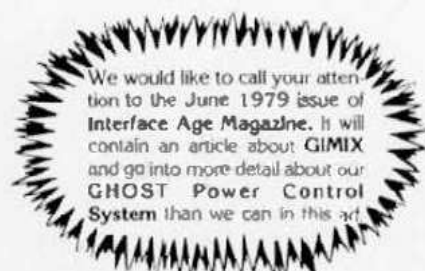


GIMIX INC.

The Company that delivers. Quality Electronic products since 1975.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

© 1979 GIMIX INC.
GIMIX® and GHOST® are Registered Trademarks of GIMIX INC.

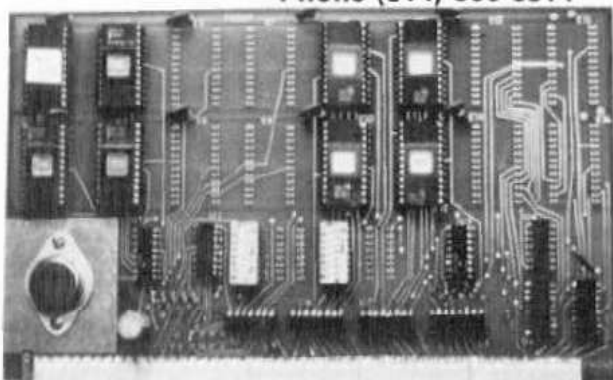


See GHOST Ad page 43

DIGITAL SERVICE & DESIGN

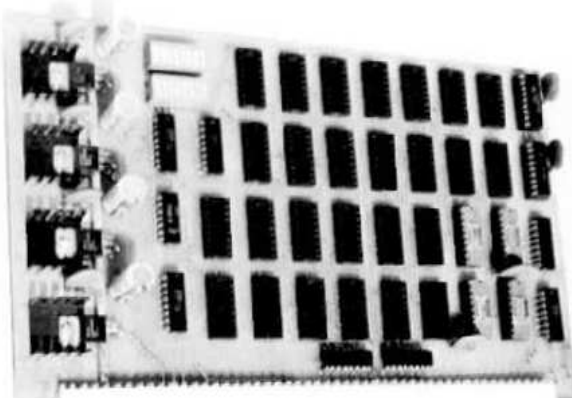


P.O. BOX 741
NEWARK, OHIO 43055
Phone (614) 366 6314



DSD P/R-32K\$27.00

32K or 16K EPROM & RAM memory card.
2716 2K x 8 or 2758 1K x 8 5V only EPROMS.
TMS 4016 2K x 8 or MK 4118 1K x 8 5V RAMS.
Up to 4 independent addressed 8K blocks. Dip
switch or jumper selected. Size 9" x 5 1/2"



DSD 2114-16K\$27.00

Full Static 16K Ram memory card designed to
use the 2114 or TMS 4045 1024 x 4 Static Ram.
The card has two independent addressed 8K
memory blocks. Card size 9" x 5 1/2". Power
requirements 7-8V unreg. @ 3.5A.

DSD U P 8255M\$14.00

Universal parallel interface card with wire wrap
area using INTEL'S 8255 parallel peripheral
interface chip. 24 programmable I/O lines.
(Three 8 bit Ports or Two 8 bit Ports with hand-
shaking) Card size 5 1/2" x 5" Standard SS-50 30
pin I/O BUS. 5V only.

Cards are bare with data and edge connector.
Ohio residents add 4 1/2% sales tax.

6847 Color Graphic card in design

EPROM PROGRAMMER Model EP-2A-79



SOFTWARE AVAILABLE FOR F-8, 8080, 6800,
8085, Z-80, 6502, KIM-1, 1802, 2650.
EPROM type is selected by a personality module
which plugs into the front of the programmer.
Power requirements are 115 VAC, 50/60 HZ at 35
watts. It is supplied with a 36 inch ribbon cable
for connecting to microcomputers. Requires 1 1/2
I/O ports. Priced at \$155 with one set of
software. Personality modules are shown below.

Part No.	Program	Price
PN-0	TMS 2708	\$15.00
PN-1	2704, 2706	15.00
PN-2	2732	30.00
PN-3	TMS 2716	15.00
PN-4	TMS 2532	30.00
PN-5	TMS 2516, 2716, 2758	15.00

Optimal Technology, Inc.
Blue Wood 127, Earlysville, VA 22936
Phone (804) 973-5482

THE MICRO WORKS

EPROM PROGRAMMER
SPEAKS FOR ITSELF!

Extended height
for easy
operation

Zero-insertion-
force socket!

Handy on/off
switch controls
on board voltages

Sexy LED!

Neat

Clever circuit
generates on
board voltage

Programs EPROMs
to full mfg.
specs

Fits in 1 I/O
slot in
SWTPC 6800

Assembled,
burned in and
tested

B-08 EPROM Programmer..... \$ 89.95
regulated + 12 v. \$104.95

U2708 Utility in EPROM \$ 29.95
All software routines necessary to test, burn, verify and copy 2708s using the
B-08 Programmer

DEPT. K, P.O. BOX 1110, DEL MAR, CA 92014 (714) 756-2687

for FLEX® users:

LOOKUP a data manager

- * SIMPLE, EASY TO USE
- * CREATE, FIND, ADD, LIST, AND DELETE DATA RECORDS
- * FREE FORM DATA DEFINITION
- * FREE FORM DATA INQUIRY
- * DATA FILES MAY BE EDITED OR USED FROM BASIC
- * RUNS IN MINIMUM SYSTEM
- * INCLUDES MINIDISK, FULL INSTRUCTION BOOKLET, AND WARRANTY
- * COMPLETE WITH FREE INDEX TO ALL 6800 ARTICLES FOR INQUIRY FROM DISK
- * FLEX VERSION 2.0 SUPPORTED

LOOKUP MDF - \$ 49.95

MYCROFTWARE SYSTEMS
P. O. BOX 1138
ST. CHARLES MO. 63301

© FLEX is a trademark of Technical System Consultants, Inc.

THE MICRO WORKS



Give your 6800 computer the gift of sight! The Micro Works Digisector® opens up a whole new world for your computer. Your micro can now be a part of the action, taking pictures like this one to amuse your friends, watching your home while you're away, helping your household robot avoid bumping into walls, providing fast to slow scan conversion for you home ... the applications abound.

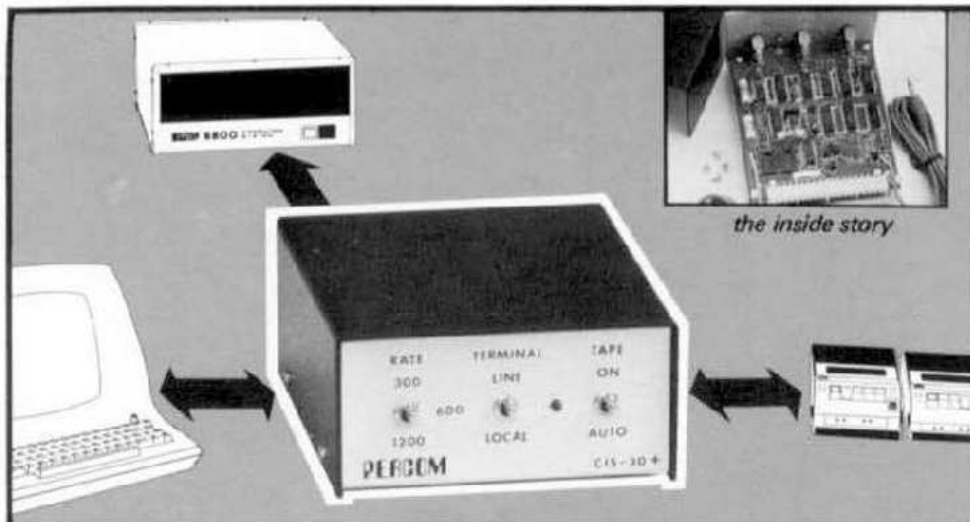
The Micro Works Digisector is a completely unique device; its resolution and speed are unmatched in industry and the price is unbeatable anywhere. The Digisector and a cheap TV camera are all you'll need to see eye to eye with your 6800. Since operation is straightforward, you don't have to be a software wizard to utilize the Digisector's extensive capabilities. The Micro Works Digisector board provides the following exclusive features:

- High Resolution — a 256 x 256 picture element scan
- Precision — 64 levels of grey scale
- Speed — Conversion times as low as 3 microseconds per pixel
- Versatility — Accepts either interlaced (NTSC) or non-interlaced (industrial) video input
- Compactness — Utilizes 1 I/O slot in your SWTPC 6800 or equivalent
- Economy — The Digisector is a professional tool priced for the hobbyist

The Digisector (DS-68), like all Micro Works products, comes fully assembled, tested and burned in. Only the highest quality components are used, and the boards are double sided with plated through holes, solder mask and silkscreen. All software is fully source listed and commented. The Micro Works is proud to add the DS-68 to its line of quality computer accessories for the hobbyist. Price 189.95

Write or call for information on other quality 6800 products, including computer portrait systems.

DEPT. 1,
P.O. BOX 1110, OEL MAR, CA 92014 (714) 755-2867



- Record and playback at 120, 60 or 30 self-clocking bytes per second (extended Kansas City Standard)
- 1200, 600 or 300 baud data terminal interface
- Dual cassette operation
- Compatible with SWTPC cassette software
- Optional kit permits program control of cassettes
- Optional adaptor permits interfacing with any computer

Upgrade your SWTPC 6800 system to 1200 baud with PerCom's CIS-30+ dual-cassette/terminal interface

The CIS-30+ ... four times as fast as SWTPC's AC-30 with the same dual-cassette capability ... **plus** a 1200-baud data terminal interface ... in a SWTPC color-compatible package that's only 1/10 the size of the AC-30.

Dependable? The simplicity of Harold Mauch PerCom Data designs says more than any well-chosen words. Simply put, for only \$79.95* you get the fastest, most dependable dual function interface you can buy for your SWTPC 6800.

To order or request literature call Percom toll-free 1-800-527-1592.

PerCom 'peripherals for personal computing'

PERCOM

PERCOM DATA COMPANY, INC.
DEPT. 68
211 N. KIRBY • GARLAND, TX 75042
(214) 272-3421

*Kit price. Assembled and tested:
\$99.95 + shipping. Tex. res. add
5% tax. BAC & MC available.

CRUNCHERS CORNER

Conducted by
Jack Bryant

Questions and comments submitted to this column can be on any subject relevant to "number crunching," and should be addressed to:

Jack Bryant
Department of Mathematics
Texas A&M University
College Station, Texas 77843

We ask that all correspondents supply their names and addresses.

MORE INTEGER ARITHMETIC

In earlier columns, we introduced 16 bit two's complement integer arithmetic. A "package" of fundamental subroutines, including division with remainder and I/O conversions, was coded in M6800 assembly language. Also included was a very simple program to illustrate and test the subroutines. In this month's column, we have three closely related goals: to clean up a loose end in the test program, to point out that last month's program evaluates expressions in a very nonstandard fashion, and to show how last month's package can be developed into an indexed integer arithmetic package. Let's start with the last.

We begin by pointing out the main disadvantage to accumulator arithmetic. Suppose we want to use subroutine ADDI to add two numbers, say in page zero memory locations with assembly language names OPR1 and OPR2, and we wish the result to end up in OPR1. Since this is the main program, we can assume that the index register is available, and we code:

```
LDX OPR1
STX I
LOX OPR
STX M
JSR A I
LOX I
STX OPR1
```

This segment requires 15 bytes. We could better have written

```
LDA A OPR1+1
ADD A OPR2+1
STA A OPR1+1
LOA A OPR1
ADC A OPR2
STA A OPR1
```

which requires only 12 bytes and is even faster than ADDI. The problem in using ADDI is moving the parameters (the contents of OPR1 and OPR2) to the software accumulators and back. The simplest but not always the best way out of this problem is to have no software accumulators and to maintain a *stack* of operands. (Another method is to make the arithmetic functions honest subroutines with parameters being passed using the stack pointer and indexed addressing. Much more on this later.) For the present, let's consider only the use of the index register to maintain the operand stack.

ABSOLUTE ADDRESS	DATA
n	INDEX + MS BYTE
n+1	LS BYTE
n+2	MS BYTE
n+3	LS BYTE

OPR1
OPR2

In this situation, the following subroutine ADDIX will replace (OPR1) with (OPR2)+(OPR1), and it can be called efficiently.

```

ADDIX LDA A 1,X
      ADD A 3,X
      STA A 1,X
      LDA A 0,X
      ADC A 2,X
      STA A 0,X
      RTS

```

The user does not have to be concerned with the absolute address of the operands, nor with tedious in-line coding. However, a new orientation must be taken: namely, the operands are now managed using the index register, and binary operands now refer to the most recent two operands. To realize this, another keyword is required which operates somewhat like [ENTER+] does on some hand held calculators (most notably, those of Hewlett-Packard).

Before going on, we should note that ADDIX takes roughly twice as long as ADDI; however, for more complex operations the difference between indexed and absolute addressing is relatively much less, and indexed addressing often requires less memory. As a general rule, expect to give away about twenty percent in time and save ten percent in memory when using indexed addressing.

AN INDEXED OPERAND STACK

The M6800 has a *real* stack, used for subroutine linkage and for the management of interrupts. What we mean by an *indexed stack* is an area of RAM which is managed like a real stack, but using the index register. Two variables are needed to manage the stack for our purposes:

TOPSTK	The top of the stack
CURSTK	The present pointer to the last two operands in the stack.

The size of the stack is limited only by how much memory is devoted to it. Operands enter the stack from the bottom, and are eliminated following an arithmetic operation. Binary operations eliminate the lowest operand on the stack and increment CURSTK to point to the present lowest. Unary operations replace the lowest by the result of the operation. (This is quite different from the algebraic approach taken before.) Before either type of operation, the eliminated operand should be saved so that some command can be used to retrieve it.

What seems to be evolving here is similar to the evaluation of an expression written in *Polish* notation (so called because it was developed by the Polish logician J. Lukasiewicz). In ordinary Polish notation, the operators precede the operands. This is unnatural for hand entry of an expression. In reverse Polish, the operands precede the operators. In a Hewlett-Packard calculator, instead of entering an algebraic expression such as $1+2=$, one enters 1 [ENTER +] 2 + . The [ENTER +] key is used as a delimiter between operands 1 and 2. What makes this method so effective is the stack is now prepared for another operand and for another operator.

Any expression in algebraic notation can be translated into reverse Polish and contariwise. The interested reader may wish to try a few: in these examples, we let the operands be denoted by single

alphabetic characters (A,B,C,...) and consider only the four basic binary operators. (Others will be added later.)

Examples:

Algebraic	Reverse Polish
1. $A*(B+C)$	ABC+*
2. $(A+B)*C+D$	AB+C*D+
3. $A+Y*(B+Y*(C+Y*(D+Y*E)))$	AYBYCYDYE*****+

Exercises:

4. $A+B*C$
5. $A/B-C*D$
6. $A+(B+C*D/(E-F))$

Example 3 is the evaluation of a fourth degree polynomial after it has been nested (i.e., using Horner's method). Note how much simpler the Polish expression is.

A Polish expression is much more easily evaluated than an ordinary one. The list of operands and operators is scanned left-to right; when an operator (binary or otherwise) is encountered, the arguments for the operator are the immediately preceding operand stack elements. A binary operator requires two. The operator is evaluated, and all operands replaced by the evaluation. (This reduces by 1 the total number of operands for a binary operator.)

Before we consider how this can be coded, let's look at something strange about last month's approach. It is also related to the evaluation of expressions.

Suppose, in last month's program, one entered the string:

12345/5-1000N*2A [RETURN] .

Essentially instantly, 2938 is printed or displayed. It is instructive to examine

the contents of I and M as this evaluation progresses:

I	M	OPR
12345	5	/
2469	1000	-
1469		N
-1469	2	*
-2938		A
2938 + displayed		

Although the intent of last month's program was to allow notation in the test which was somewhat like ordinary algebraic notation, we neither allowed parenthesis nor recognized the different hierarchy which the various operations enjoy in usual notation. We will seriously address this problem in later columns.

IMPLEMENTING AN OPERAND STACK

The rules for managing the operand stack are implied by the rules for the evaluation of Polish expressions. In this implementation, we begin with (CURSTK) = (TOPSTK), with a certain number (say 10) bytes being reserved for the stack work area below actual operands.

The stack starts at (TOPSTK) and is increased in size when operands are placed in it. For example, suppose we reserve 20 bytes for the stack (allowing 5 operands to be stacked before any binary operators are applied).

Suppose the stack begins at location \$200. Then the actual configuration of the stack initially and after the entry of numbers 8, -1, 10 and 256 is as follows:

Location (hex)	Initial	Stack 8	Stack -1	Stack 10	Stack 256
200	T,C→00	T,C→00	T → 00	T → 00	T → 00
201	00	00	00	00	00
202	00	00	C → 00	00	00
203	00	08	08	08	08
204	Work	Work	FF	C → FF	FF
205	Area	Area	FF	FF	FF
206			Work	00	C → 00
207			Area	0A	0A
208				W k	01
209				Area	00
20A					Work
20B					Area
20C					
20D					
20E					
20F					
210					
211					
212					
213					

Consider now the sequence +, +, -, ABS of operands:

Location (hex)	After +	After +	After -	After ABS
200	T → 00	T → 00	T,C→00	T,C→00
201	00	00	00	00
202	00	C → 00	FE	01
203	08	08	FF	01
204	C → FF	01	Work	Work
205	FF	09	Area	Area
206	01	Work		
207	0A	Area		
208	Work			
209	Area			

If a binary operator is encountered with (TOPSTK) = (CURSTK), perform the operation as usual, except, following the operation, move the result to (TOPSTK)+2 and do not decrement (CURSTK) by two. This feature allows the user to believe there is an infinite stack, initially all zeros. When operands are being entered, check for the stack being full (as it became in this example), and take care to not overflow the space provided.

In next month's column, we implement these ideas in an enhanced version of the integer arithmetic package which now expects reverse Polish entry order. This program also contains other enhancements of the integer arithmetic package. One of them is this month's program, discussed next.

LOOSE ENDS IN BANG

The purpose of most of the logic in BANG was to decode a one byte operator and branch to the appropriate routine. A typical segment was coded:

```

    CMP B #$2B +
    BNE SUBQ
    JSR ADDJ GO ADD
    BRA BANGN
SUBQ CMP B #$2D -

```

This takes 8 or 9 bytes per keyword (operator). It is an easily understood method, but hardly elegant. A fancy way to branch according to a keyword (from a one byte table) involves the simple table lookup subroutine LOOKUP: (see Listing 1)

One can see that this table lookup method requires subroutine LOOKUP (which, except for the user error routine jump, is relocatable and reentrant) to be present only once for all such table search operations. Only 7 bytes plus three bytes per table entry are needed. Since we are processing 7 operators in BANG, we require $20+28=48$ bytes and get routine LOOKUP free. This is less than the 60 bytes required by the simpler coding in the first version of BANG.

Since LOOKUP is reentrant, it can (indirectly) call itself. This feature could be used when some keywords have two byte length. An example of this situation is the program by which the flow charts seen in this column are prepared. Most of the keywords are two bytes in length. For example:

```

DE  a decision box
DI  a disk box
DO  a document box

```

Keywords are processed as follows: with

the first letter (in this example, 'D'):

```

    LOX  #FIRSTT LOAD FIRST LTR TABLE
    JMP  LOOKUP  DECODE
FIRSTT ...
    FCB  'D      KEYWORD "D"
    FDB  DFIRST  ADDRESS OF D ROUTINE
    ...
DFIRST JSR  INEE  GET NEXT BYTE
    TAB                      SAVE IN ACCB
    LOX  #DSEC   LOAD SEC LTR TABLE
    JMP  LOOKUP  GO DO IT
DSEC   FCB  'E
    FDB  DECISN
    FCB  'I
    FDB  DISK
    FCB  'D
    FDB  DOCUMT
    FCB  D

```

The routine DECISN also uses LOOKUP since it requires further input (to show which arms on the decision box are desired).

Answers to Exercises:

Algebraic	Reverse Polish
4. $A+B*C$	ABC**
5. $A/B-C*D$	AB/CD*-
6. $A*(B+C*D/(E-F))$	ABCD*EF-/++

```

*
*      SUBROUTINE LOOKUP
*
*  ENTRY: X POINTS TO THE KEYWORD-BRANCH
*          ADDRESS TABLE. ACCB CONTAINS
*          THE ONE BYTE KEY.
*
*  EXIT:  IF FOUND, A JUMP TO THE ADDRESS
*          FOLLOWING THE MATCH IN THE TABLE
*          IS EXECUTED. IF NOT FOUND, A JUMP
*          TO ERROR ROUTINE USRERR WITH ACCA
*          CONTAINING D1 IS EXECUTED.
*
LOOKUP LDA A 0,X      LOAD KEYWORD
      BEQ  TABEND    TEST IF END

```

```

CBA      SEE IF MATCH
BEQ      FOUND
INX
INX      SKIP PAST ADR
INX
BRA      LOOKUP
FOUND    LDX      1,X      MATCH: LOAD ADDRESS
          JMP      0,X      AND BRANCH
TABEND    INC      A      SHOW ERROR CODE 1
          JMP      USRERR   FOR RUNNING OUT TBL

```

★
★ EXAMPLE TABLE FOR BANG
★

```

BANGTA  FCB      '+'      ADDITION
          FDB      ADDI
          FCB      '-'      SUBTRACTION
          FDB      SUBI
          FCB      '*'      MULTIPLICATION
          FDB      MULI
          FCB      '/'      DIVISION
          FDB      DIVI
          FCB      'R      REMAINDER
          FDB      REMI
          FCB      'A      ABS VALUE
          FDB      ABSI
          FCB      'N      NEGATION
          FDB      NEGI
          FCB      0      ZERO TERMINATOR

```

★
★ EXAMPLE USAGE
★

★ ASSUME ACCB CONTAINS THE OPERATOR
★

```

LDX      #BANGTA LOAD TABLE START
JSR      LOOKUP  ALL ROUTINES ARE
...      SUBROUTINES.

```

Listing 1. An example of a table lookup program which can shorten last month's program BANG.

LETTERS

JOHN R. GYE
8001 Fairview Ave. S.E.
Lynn, WA 98583
(206) 491-7411

April 21, 1979

'68' Micro Journal
3018 Hamill Road
Mixon, TN 37343

Attention: Joyce Williams

Dear Sirs:

Just received a (complimentary?) issue of '68' Micro Journal Volume 1, Issue 2, just in the nick of time. I was just about to switch to 6502-based Ohio Scientific as I've given up on trying to upgrade the Heath ET-3400 Trainer -- their (finally!) announced expansion kit still won't get me any closer to mini-floppy configuration which is my ultimate goal.

Now, if you think I'm going to deface this magazine by cutting out your subscription coupon, you're wrong! I hope you can accept this letter in lieu of your form.

Please enter my charter two-year subscription at the \$18.50 rate. Charge it to my VISA account Number [REDACTED]

If it is possible, send me the first issue Volume 1 issue 1, February to go with my existing March/April issue and rush issue 3, May as soon as it is off the presses.

By the way, my compliments to Mickey Ferguson. His article on the CT-80 Terminal has told me more than other Kilobaud or SWTPC themselves.

Cordially,

John R. Gye

Ref Bull Electronics, Inc.
John R. Gye, Owner
P.O. Box 1514, P.O. Box 1514, 7412
Hampden, Washington 98047

SWTPC Southwest Technical Products Corporation
219 W. Ruppel
San Antonio, Texas 78218

May 3, 1979

Dear Editor,

We would like to point out to all users of our system that the story printed in the latest issue of 6800 BITS is very misleading and should be clarified.

Copy of paragraph is enclosed.

This story seems to imply that our 8 inch floppy disk systems are not usable with our computer systems and that even if you manage to get the clock speed up to 1.6 MHz it still will not work because our 32 K memories will not cycle this fast.

What is not mentioned is that the disk system referred to is not the Southwest Technical Products DMF-1 eight inch disk system, but the Snake Signal Broadcasting eight inch disk system. Our DMF-1 will work just fine with all existing 6800 computer systems. It uses a DMA controller that will properly interface the floppy disk to systems running at 850 to 1200 KHz as most do. The Snake Signal disk system uses a less expensive controller design that is similar to the technique commonly used with mini-drives in which the processor handles the information coming in from the disk directly. This is fine with mini-drives, but an eight inch drive inputs data too fast for a processor operating at 1.0 MHz or so to keep up. The solution used in the Snake Signal drive is to operate the processor at 1.6 MHz minimum. The fact that this makes it incompatible with most 6800 CPUs out there in the world is never mentioned in any literature, or ad that I have seen.

Southwest Technical Products has always tried to make new equipment as compatible as possible with previous designs and to make upgrades possible where practical. We would like to set the record straight in the above story as to exactly where the problems and incompatibilities originate.

Sincerely,

Daniel Meyer
Daniel Meyer - Pres.

A STAFF REVIEW The 68 Micro Journal Lab

The JPC Products Model TC-3 cassette interface is rather unique. It uses only five integrated circuits (including the voltage regulator) on a single glass-epoxy circuit board. It plugs directly into the I/O bus on an SS-50 computer. And it provides a VERY RELIABLE means of saving programs and data to tape at bit rates up to 4800 baud. The TC-3 even includes

provision for automatic motor control for 1-8 drives. Although three drives is really a practical maximum. The TC-3 cassette interface sells for \$49.95 from JPC Products, P.O. Box 5615, Albuquerque, New Mexico 87185. JPC Products also offers a cassette operating system for use with the TC-3 for \$19.95 additional. We will be considering the TC-3 cassette interface and the CFM/3 cassette operating system as a single product. This is because we feel the average person should consider them as a single product, which sells for sixty nine dollars ninety cents.

The most impressive thing about the TC-3 is the overwhelming amount of documentation. There are twenty-five pages of documentation with the TC-3 and an additional fifty-nine pages with the CFM/3. In addition to the basic assembly instructions, schematics, etc. which we expect to find in a product of this kind; there is a section on tape recorders which makes several recommendations on the selection of a recorder for use with the TC-3. The CFM/3 documentation is really easy to use because the name of the command being described is printed in letters 1/2 inch high at the beginning of the discussion. This makes it very easy to find the command you are looking for. We should also mention that fully annotated listings of all programs & patches are included as part of the documentation.

The basic TC-3 cassette interface includes a set of utility driver subroutines. Also included is a subroutine that allows MIKBUG format Kansas City standard tapes to be loaded. You could write a complete cassette operating system around these subroutines. But why bother? The CFM/3 is a complete cassette operating system. With the CFM/3 operating system, you get patches for SWTPC BASIC, SWTPC CORES editor/assembler, the TSC editor, and TSC assembler. The patches allow you to save and load named files through the CFM/3 file manager. It would be fairly easy to modify BASIC for data files through the file manager, too. But you are on your own there, JPC Products does not provide the necessary patches for that.

In use in the 68 Micro Journal lab, we have found the TC-3 (with CFM/3) to be both fast and reliable. More reliable

than our National Multiplex CC-8 digital recorder at 4800 baud, in fact. With the CFM/3 operating system the TC-3 is very convenient to use. Considering the TC-3 cassette interface without the CFM/3 operating system, it earns a 68 Micro Journal lab rating of AA. The hardware is excellent, but the driver subroutines are a bit impractical to use without some additional software to tie it all together. Considering the TC-3 and the CFM/3 together as a single product, the rating must be AAA. Rated AAA because it is an excellent low cost alternative to a floppy disk system for the person on a tight budget.

MEF

BASIC PROGRAMMING QUICKIES

David Eagle
3330 S. Garland Way
Lakewood, CO 80227

Two routines for SWTPC BASIC, Version 2.3.

```
9010 REM ARC-COSINE SUBROUTINE
9020 REM C=ARC COS(X), RADIANS
9030 REM -1 <X <=1
9040 P0=3.14159265
9050 IF X=0 THEN C=P0/2: RETURN
9060 IF ABS (X) >=1 THEN C=(1-SGN (X))*P0/2: RETURN
9070 C= ATAN (SQR(1-X*X)/X)+(1-SGN (X))*P0/2: RETURN
```

```
8000 REM ARC-SINE SUBROUTINE
8010 REM S=ARC SIN(X), RADIANS
8020 REM -1 <X <=1
8030 P0=3.14159265
8040 IF ABS (X) >= 1 THEN S=SGN (X)*P0/2: RETURN
8050 S=ATAN (X/SQR(1-X*X)): RETURN
```

Submitted by: David Eagle
3330 S. Garland Way
Lakewood, CO 80227
303-985-5049

LOCATE (UPDATE)

Dennis Womack
Rt. 4, 109 Foster Dr.
Ringgold, GA 30738

It was back last August, when Don Williams prodded me into writing a program called FILES. I had just installed a SWTPC MF-68 minifloppy on an AMI EVK 6800 computer for my employer. Because of the differences in the monitors, I needed to know where the new MINIFLEX operating system was going to go in memory. The old FDOS had a command called FILES which

would list the directory of files. Each file's starting address, ending address, and transfer address was listed. Don kept saying, "I wish TSC had included a FILES command. Why don't you write one?" (That's the way he is.) My reply was always: "When you get me the 'Advanced Programmers Guide', maybe I will."

In August push turned into shove and the program was written. It had one bug. If the file did not have a transfer address record as the last record, it would not print the ending address of the last memory segment. Other than that, the new FILES.CMD would handle a segmented, binary format file correctly. An example of this kind of file is DOS.SYS, which is composed of many non-contiguous memory segments. FILES.CMD satisfied an immediate need and several people have received copies of it.

As Don was gearing up to do 68 Micro he asked for an article on FILES.CMD. The bug was corrected and a small article was written to accompany the source listing. After some thought, the program was renamed LOCATE.CMD. The new name seemed to make more sense than FILES.

One day after the article had been submitted, Don called. "Guess what? I've got another LOCATE. What do you want to do?" Robert Pigford's article was more tutorial, so Don decided to run it. Robert's article was great. Later, while reading his article, I realized he was not handling segmented, binary files. I am not being critical. His program probably does exactly what he intended. Segmented, binary files are created when you:

1. append two or more binary files, or
2. segment an assembly language program (ie: with RMB's).

The irony of it all, is that two people, independently, used the same name for a program, which did nearly the same thing. Truth is stranger than fiction...

I am not going to discuss how FLEX handles files. Read Robert Pigford's LOCATE in the last issue for that. Better yet, buy the "Advanced Programmer's Guide". What follows is a description of how segmented, binary files are handled.

LOCATE functions by opening the specified file for read, and then reading

the file record by record. As each record is read, the program decides how to handle it. Transfer addresses are saved, and a flag is set to indicate there is a transfer address for the file. When a binary record is read, it is determined if this record is contiguous with the last binary record. If it is not, then the starting and ending addresses of the last contiguous segment are printed and a new segment is started. When the end-of-file error occurs the program finishes by printing the starting and ending addresses for the current segment. It then prints the transfer address, if any. To use this utility simply type "LOCATE (FILE SPEC)". Happy hunting.

NAM LOCATE 1/27/79

```

**
* LOCATE UTILITY
*
* THIS PROGRAM READS A BINARY FILE AND
* LISTS EACH SEGMENT OF MEMORY LOADED.
* THE ULTIMATE TRANSFER ADDRESS (IF ANY)
* IS ALSO LISTED.

```

```

* TO USE:
* TYPE "LOCATE,(FILE SPEC)"
*
* WRITTEN BY DENNIS WOMACK
* COPYRIGHT JANUARY 27, 1979
* ALL RIGHTS RESERVED
**

```

```

*
* ROUTINES USED FROM MINIFLEX
*

```

WARMS	EQU	\$7103	DOS WARM START ENTRY
GETFIL	EQU	\$7127	GET FILE SPECS
PUTCHR	EQU	\$7112	O/P AR
PSTRNG	EQU	\$7118	O/P CRLF THEN STRING
PCRLF	EQU	\$711E	O/P CRLF
SETEXT	EQU	\$712D	SET DEFAULT EXT
OUTHEX	EQU	\$7139	O/P 2 HEX DIGITS
RPTERR	EQU	\$713C	REPORT DISK ERROR
FMS	EQU	\$7806	FMS CALL ENTRY
FMSCLS	EQU	\$7803	FMS CLOSE ENTRY

```

*
* FILE CONTROL BLOCK
*

```

FCB	EQU	\$7740	SYSTEM FCB
-----	-----	--------	------------

```

37      * LOCATE UTILITY
38      *
39      1000      ORG      $1000
40      1000 20 01  LOCATE  BRA      LOC1
41      *
42      1002 02      FCB      2      VERSION NUMBER
43      *
44      1003 BF 11 36 LOC1      STS      STEMP      SAVE CURRENT STACK REG
45      1006 7F 11 28      CLR      TFLAG      SET TA FLAG TO NO TA
46      1009 CE 77 40      LDX      #FCB      POINT TO FCB
47      100C BD 71 27      JSR      GETFIL      GET FILE SPEC
48      100F 24 03      BCC      LOC2      ANY ERRORS?
49      1011 7E 10 D9      JMP      LOC14      YES, GO HANDLE
50      1014 CE 77 40 LOC2      LDX      #FCB      NO, POINT TO FCB
51      1017 86 00      LDA A      #0      SETUP .BIN
52      1019 BD 71 2D      JSR      SETEXT      DEFAULT
53      101C CE 77 40      LDX      #FCB      POINT TO FCB
54      101F 86 01      LDA A      #1      OPEN FOR READ CODE
55      1021 A7 00      STA A      0,X      STORE IN FCB
56      1023 BD 78 06      JSR      FMS      DO OPEN FUNC
57      1026 27 03      BEQ      LOC3      ERRORS?
58      1028 7E 10 CB      JMP      LOC12      YES, GO HANDLE
59      102B 86 FF LOC3      LDA A      #$FF      NO
60      102D A7 3B      STA A      59,X      INHIBIT SP COMP
61      102F BD 71 1E      JSR      PCRLF      PRINT CRLF
62      1032 BD 71 1E      JSR      PCRLF      PRINT CRLF
63      *
64      * READ A RECORD
65      *
66      1035 BD 10 EA LOC4      JSR      LOC17      GET NEXT BYTE
67      1038 81 02      CMP A      #$02      BYTE=2?
68      103A 27 14      BEQ      LOC6      YES, HANDLE BINARY REC
69      103C 81 16      CMP A      #$16      BYTE=$16?
70      103E 27 02      BEQ      LOC5      YES, HANDLE TA REC
71      1040 20 F3      BRA      LOC4      NO, GO GET NEXT BYTE
72      *
73      * TRANSFER ADDRESS RECORD HANDLER
74      *
75      1042 BD 10 FB LOC5      JSR      LOC19      GET NEXT 2 BYTES AS ADDR
76      1045 FE 11 34      LDX      XTEMP      ADDR IS IN XTEMP
77      1048 FF 11 2A      STX      TADDR      MOVE TO TADDR
78      104B 7C 11 28      INC      TFLAG      BUMP FLAG TO YES
79      104E 20 E5      BRA      LOC4      GO READ ANOTHER RECORD
80      *
81      * BINARY RECORD HANDLER
82      *
83      1050 BD 10 FB LOC6      JSR      LOC19      GET NEXT 2 BYTES AS ADDR
84      1053 FE 11 34      LDX      XTEMP      ADDR IS IN XTEMP
85      1056 FF 11 30      STX      BEGREC      BEGINNING OF REC=ADDR
86      1059 BD 10 EA      JSR      LOC17      GET NEXT BYTE
87      105C B7 11 27      STA A      COUNT      SAVE AS COUNT
88      105F 4A      DEC A
89      1060 BB 11 31      ADD A      BEGREC+1      ADD COUNT-1 TO
90      1063 B7 11 33      STA A      ENDREC+1      BEGINNING OF RECORD
91      1066 86 00      LDA A      #0      SAVE IT AT
92      1068 B9 11 30      ADC A      BEGREC      END OF RECORD
93      106B B7 11 32      STA A      ENDREC
94      106E BD 10 EA LOC7      JSR      LOC17      READ REST OF RECORD
95      1071 7A 11 27      DEC      COUNT

```

96	1074	26	F8		BNE	LOC7	
97	1076	FE	11	2E	LDX	FNDSEG	
98	1079	08			INX		
99	107A	BC	11	30	CPX	BEGREC	I+ENDSEG=BEGREC?
100	107D	26	08		BNE	LOC8	NO, HANDLE AS NOT CONTINUOUS
101	107F	FE	11	32	LDX	ENDREC	
102	1082	FF	11	2E	STX	ENDSEG	ENDSEG=ENDREC
103	1085	20	AE		BRA	LOC4	READ ANOTHER RECORD
104					*		
105	1087	7D	11	29	LOC8	TST	SKIP
106	108A	26	03		BNE	LOC9	SKIP FIRST PRINT
107	108C	BD	11	06	JSR	LOC20	PRINT BEG + END OF SEG
108	108F	7F	11	29	LOC9	CLR	SKIP
109	1092	FE	11	30	LDX	BEGREC	
110	1095	FF	11	2C	STX	BEGSEG	BEGSEG=BEGREC
111	1098	FE	11	32	LDX	ENDREC	
112	109B	FF	11	2E	STX	ENDSEG	ENDSEG=ENDREC
113	109E	20	95		BRA	LOC4	READ ANOTHER RECORD
114					*		
115					* END OF FILE HANDLER		
116					*		
117	10A0	BD	11	06	LOC10	JSR	LOC20
118	10A3	7D	11	28		TST	TFLAG
119	10A6	27	12			BFQ	LOC11
120	10A8	CE	11	57		LDX	#MSG1
121	10AB	BD	71	18		JSR	PSTRNG
122	10AE	CE	11	2A		LDX	#TADDR
123	10B1	BD	71	39		JSR	OUTHEX
124	10B4	CE	11	2B		LDX	#TADDR+1
125	10B7	BD	71	39		JSR	OUTHEX
126	10BA	BD	71	1E	LOC11	JSR	PCRLF
127	10BD	CE	77	40		LDX	#FCB
128	10C0	86	04			LDA A	#4
129	10C2	A7	00			STA A	0,X
130	10C4	BD	78	06		JSR	FMS
131	10C7	26	15			BNE	LOC15
132	10C9	20	19			BRA	LOC16
133					*		
134					* ERROR HANDLER		
135					*		
136	10CB	A6	01		LOC12	LDA A	1,X
137	10CD	81	04			CMP A	#4
138	10CF	26	0D			BNE	LOC15
139	10D1	CE	11	38		LDX	#NOFST
140	10D4	BD	71	18	LOC13	JSR	PSTRNG
141	10D7	20	0B			BRA	LOC16
142					*		
143	10D9	CE	11	45	LOC14	LDX	#ILLST
144	10DC	20	F6			BRA	LOC13
145					*		
146	10DE	BD	71	3C	LOC15	JSR	RPTERR
147	10E1	BD	78	03		JSR	FMSCLS
148	10E4	BE	11	36	LOC16	LDS	STEMP
149	10E7	7E	71	03		JMP	WARMS
150					*		
151					* GET NEXT BYTE ROUTINE		
152					*		
153	10EA	CE	77	40	LOC17	LDX	#FCB
154	10ED	BD	78	06		JSR	FMS

GET NEXT BYTE

```

155 10F0 27 08      BEQ      LOC18
156 10F2 A6 01      LDA A    1,X      GET ERROR STATUS
157 10F4 81 08      CMP A    #B      IS IT EOF?
158 10F6 26 E6      BNE      LOC15    NO, ERROR
159 10F8 20 A6      BRA      LOC10    YES, GO HANDLE
160 10FA 39          LOC18    RTS
161
162                *
163                * GET NEXT TWO BYTES FOR ADDR ROUTINE
164                *
164 10FB 8D ED      LOC19    BSR      LOC17    GET NEXT BYTE
165 10FD B7 11 34      STA A    XTEMP      XTEMP MSB=NEXT BYTE
166 1100 8D E8      BSR      LOC17    GET NEXT BYTE
167 1102 B7 11 35      STA A    XTEMP+1    XTEMP LSB=NEXT BYTE
168 1105 39          RTS
169
170                *
171                * PRINT BEGINNING AND END OF SEGMENT ROUTINE
172                *
172 1106 CE 11 2C      LOC20    LDX      #BEGSEG
173 1109 BD 71 39      JSR      OUTHEX      PRINT BEGSEG MSB
174 110C CE 11 2D      LDX      #BEGSEG+1
175 110F BD 71 39      JSR      OUTHEX      PRINT BEGSEG LSB
176 1112 86 2D      LDA A    #'-'
177 1114 BD 71 12      JSR      PUTCHR      PRINT "-"
178 1117 CE 11 2E      LDX      #ENDSEG
179 111A BD 71 39      JSR      OUTHEX      PRINT ENDSEG MSB
180 111D CE 11 2F      LDX      #ENDSEG+1
181 1120 BD 71 39      JSR      OUTHEX      PRINT ENDSEG LSB
182 1123 BD 71 1E      JSR      PCRLF      PRINT CRLF
183 1126 39          RTS
184
185                *
186                * VARIABLES
187                *
187 1127 00          COUNT    FCB      0      BYTE COUNT
188 1128 00          TFLAG    FCB      0      TRANSFER ADDR FLAG, 0=NO
189 1129 FF          SKIP     FCB      $FF    SKIP FIRST PRINT FLAG
190 112A 00 00      TADDR     FDB      0      TRANSFER ADDR
191 112C 00 00      BEGSEG    FDB      0      BEGINNING OF SEGMENT
192 112E 00 00      ENDSEG    FDB      0      END OF SEGMENT
193 1130 00 00      BEGREC    FDB      0      BEGINNING OF RECORD
194 1132 00 00      ENDREC    FDB      0      END OF RECORD
195 1134 00 00      XTEMP     FDB      0      TEMPORARY FOR XR
196 1136 00 00      STMP      FDB      0      STACK REG SAVE
197
198                *
199                * STRINGS FOR ERROR MESSAGES
200                *
200 1138 4E          NOFST     FCC      /NO SUCH FILE/
201 1139 4F 20
202 113B 53 55
203 113D 43 48
204 113F 20 46
205 1141 49 4C
206 1143 45
207 1144 04          ILLST     FCB      4
208 1145 49          ILLST     FCC      /ILLEGAL FILE NAME/
209 1146 4C 4C
210 1148 45 47
211 114A 41 4C
212 114C 20 46
213 114E 49 4C

```

```

1150 45 20
1152 4E 41
1154 4D 45
203 1156 04      FCB      4
204 1157 54      MSG1    FCC    /TRANSFER ADDRESS=$/
1158 52 41
115A 4E 53
115C 46 45
115E 52 20
1160 41 44
1162 44 52
1164 45 53
1166 53 3D
1168 24
205 1169 04      FCB      4
206                      END    LOCATE

```

NO ERROR(S) DETECTED

SYMBOL TABLE:

BEGREC 1130	BEGSEG 112C	COUNT 1127	ENDREC 1132	ENDSEG 112E
FCB 7740	FMS 7806	FMSCLS 7803	GETFIL 7127	ILLST 1145
LOC1 1003	LOC10 10A0	LOC11 10BA	LOC12 10CB	LOC13 10D4
LOC14 10D9	LOC15 10DE	LOC16 10E4	LOC17 10EA	LOC18 10FA
LOC19 10FB	LOC2 1014	LOC20 1106	LOC3 102B	LOC4 1035
LOC5 1042	LOC6 1050	LOC7 106E	LOC8 1087	LOC9 108F
LOCATE 1000	MSG1 1157	NOFST 1138	OUTHEX 7139	PCRLF 711E
PSTRNG 7118	PUTCHR 7112	RPTERR 713C	SETEXT 712D	SKIP 1129
STEMP 1136	TADDR 112A	TFLAG 1128	WARMS 7103	XTEMP 1134

BASIC RENUMBERING

for
Southwest Tech BASIC

Mickey Ferguson
PO Box 708
Trenton, GA 30752

Did you ever program yourself into a corner? Surely you have decided to write some simple little program while sitting at your terminal. And as you progress, the little program grows becoming much less simple. Then you find it will run properly if you add a line between lines 80 and 90. But you have used ALL possible line numbers from 80 to 90. So, you start re-entering lines, renumbering them as you go. And trying not to make any mistakes. Or you might just want to impress your friends by always having all of your programs begin with line 10 and all line numbers incremented by 10. It sure makes it appear that you knew exactly what you were doing when you wrote the program. Using the program included with this article will get you out of those corners we all sometimes get ourselves into. And if you don't tell your friends about it, I won't either.

The RENUM program renumbers the program currently resident in memory. The first line number after renumbering is 10, and all line numbers are incremented by 10. As presented here, RENUM is written to be used as a FLEX utility but returns control to BASIC instead of to FLEX. It is assembled for a starting address of \$C000 because I had some memory there, and because it can be called from the SWTBUG "2" command once it is in memory. You may assemble it for any location in memory convenient for you. And you could even assemble it to reside contiguous with BASIC (providing you lock out BASIC to keep RENUM from being overwritten by BASIC). RENUM is written for SWTPC Disk BASIC Version 3.0 for the FLEX operating system. But it is easily modified for use with any of the BASIC's written by Robert Uiterwyk. And I will give you the patches for SWTPC's 8k Version 2.0 and miniFlex

BASICs. But first, let's look at how RENUM works.

I will not discuss the operation of RENUM in minute detail, because (hopefully) the source is commented well enough to make that unnecessary. RENUM runs in two passes through the BASIC program. During the first pass, a lookup table is built in memory following the BASIC program. In the second pass, all line numbers in the BASIC program are replaced with the new line numbers from the table. An entry in the lookup table consists of the old line number followed by the new line number. In the table the line numbers are stored in packed BCD (Binary Coded Decimal) format; thus each entry in the table requires four bytes to store two line numbers. RENUM has to use a lookup table because there is no way for a program to calculate a new line number based on the old one. If humans were as logical as computers, it would be simple to devise a method to calculate the new line numbers. But then renumbering programs would be unnecessary.

BASIC adds its share of headaches for RENUM during the second pass. As I've already mentioned, line numbers are stored in packed BCD format. But this is not the case when a line number is used in a BASIC statement (like: GOTO 200). Line numbers referenced in a BASIC statement are stored in ASCII format. So RENUM must be able to recognize both BCD and ASCII line numbers, and be able to distinguish them from constants used in the BASIC program. If this weren't enough, BASIC further complicates things by converting the first keyword used in a statement to a token. While any subsequent keywords are merely stored in ASCII. (Your BASIC programs would run a lot faster if ALL keywords were converted to tokens.) So RENUM must be able to recognize keywords in ASCII and by their tokens. The tokens BASIC uses can be considered to be "a pointer to a pointer". The token is the address of the pointer to the keyword routine in BASIC's lookup table. The source for one entry in BASIC's lookup table looks something like this:

```
FCC /GOTO/  
FCB 0  
FDB GOTO ---the token points to here
```

It would be very helpful to RENUM if BASIC

did not allow multiple statement lines. Or very complex single statement lines like:

```
IF X=1 THEN IF Y=2 THEN ON Z GOTO 100,200
```

But BASIC does allow all these things and we would be very unhappy if it didn't. So RENUM must allow for them as well. Additionally, line numbers change in length when a program is renumbered, GOTO 1000 might become GOTO 90. And it would not be desirable to have the line say GOTO 0090 after the program is renumbered. So RENUM suppresses leading zeros. But GOTO 90 could become GOTO 100 and might look like GOTO100 after renumbering. So RENUM opens up the necessary room when a line number grows.

If you are very familiar with the internal workings of your BASIC, you will notice there are routines in RENUM that are also in BASIC. If you are that familiar with your BASIC, I would recommend using the routines in BASIC, instead of duplicating them in RENUM. You might also incorporate RENUM into your BASIC's keyword table and add an optional argument. So that you could say RENUM,200. And your program would be renumbered beginning with line number 200. A second argument could be added to specify the increment value. These changes would be quite simple to implement; but should not be attempted unless you really understand how your BASIC works. I chose not to do this here to make it easier to patch RENUM for different versions of BASIC. Speaking of patching RENUM for different versions of BASIC; only one change is necessary for RENUM to run with minIFLEX BASIC. You will note the line in the SYSTEM EQUATES that says:

```
BASERR EQU $CE9
```

For RENUM to work with minIFLEX BASIC, this line must be changed to:

```
BASERR EQU $CE6
```

For RENUM to run with SWTPC 8k BASIC Version 2.0, three changes must be made. As above, the BASERR line must be changed and should read:

```
BASERR EQU $BEB
```

In 8k V2.0 the tokens for GOTO and GOSUB are \$022D & \$0234 respectively. This requires two changes to the RENUM6 routine. The source as shown reads:

```
CMP A #2
BNE RENU65
CMP B #$30
BEQ GO
CMP B #$37
BEQ GO
```

To use RENUM with 8k BASIC V2.0 this code must be changed to read as follows:

```
CMP A #2
BNE RENU65
CMP B #$2D
BEQ GO
CMP B #$34
BEQ GO
```

One word of caution. It is wise to save a copy of your BASIC program to tape or disk prior to renumbering it. Should RENUM encounter an error (GOTO 200 - when there is no line 200, for example), the renumbering process is halted and control is given to BASIC through its error handling routine. If this occurs your program will be in a partially renumbered state, and completely useless to either BASIC, RENUM, or to you. All you have to do is lose a two or three hundred line program and you will really appreciate that backup copy.

As I mentioned earlier, you should be able to get RENUM to run with any of Robert Uiterwyk's BASICs. This includes MSI BASIC, Computerware BASIC, and PERCOM SuperBASIC. And possibly others that I am not aware of. All you need to know to modify RENUM for these different BASICs is:

- 1.- The address of BASIC's error handling routine
- 2.- The tokens for GOTO and GOSUB
- 3.- The memory locations used for NEXTBA, SOURCE, MSLINE, & LSLINE (if different from those shown in the listing)

But don't ask me about a version for SWTPC 4k BASIC. I wrote a renumbering program for 4k BASIC about two and one half years ago. And sold an article about it to Kilobaud - it has never been published.

So if you need renumbering for SWTPC 4k BASIC; write to Kilobaud, not to me.

SYMBOL TABLE:

ASCON	C14E	ASTM1	C2CB
BASFERR	0CE9	BCDCH1	C1C0
BCDC02	C1DB	BCDC03	C1F4
DEXGET	C148	FNDLI1	C197
FNDLIN	C18B	GO	COAF
GO35	C103	GO4	C10A
GO6	C12C	GO7	C142
LSLINE	0033	MOVE1	C277
ON	C1FD	ON1	C200
RENU00	C006	RENU45	C063
RENUM0	C009	RENUM1	C013
RENUM6	C086	RESTRT	0103
SOURCE	002E	TEMP1	C2C1
TEMP5	C2C9	THEN	C20F
TSTNO2	C2B3	TSTNUM	C14B
ASTM2	C2CC	ASTM3	C2CD
BCDCH2	C1D6	BCDCH3	C1EB
BCDCON	C1B0	CLOSUP	C286
FNDLI5	C19E	FNDLI1	C18D
GO1	COBD	GO2	CO05
GO44	C11C	GO45	C11F
GO8	C145	GOSUB	C24B
MOVRTS	C281	MSLINE	0032
ON2	C204	ON3	C208
RENU55	C080	RENU65	COAC
RENUM3	C03F	RENUM4	CO57
SKIP1	C2B6	SKIP2	C2BF
TEMP2	C2C3	TEMP3	C2C5
THEN1	C224	TSTNO	C2A7
TSTOVF	C25B		
		ASTM4	C2CE
		BCDC01	C1C9
		CLOSUP	C283
		FNDLI2	C1A4
		GO3	COF0
		GO5	C126
		GOTO	C230
		NEXTBA	002A
		ON4	C20C
		RENUM	C000
		RENUM5	C079
		SKIPSP	C2BC
		TEMP4	C2C7
		TSTNO1	C2B0

```

NAM      RENUMBER
*
* BASIC PROGRAM RENUMBERING FOR
* SWTPC DISK BASIC V3.0 FLEX
*
* SYSTEM EQUATES
*
002A     NEXTBA EQU    $2A      POINTER TO FIRST BYTE AFTER PROGRAM
002E     SOURCE EQU    $2E      POINTER TO BEGINNING OF PROGRAM
0032     MSLINE EQU    $32      TOP HALF OF HIGHEST LINE # IN PROGRAM
0033     LSLINE EQU    $33      BOTTOM HALF OF HIGHEST LINE #
0103     RESTRT EQU    $103     BASIC WARM START ENTRY
0CE9     BASERR EQU    $CE9     BASIC'S ERROR ROUTINE
C000     ORG    $C000          OR WHEREVER HANDY
*
*CHECK TO SEE IF SOURCE PROGRAM IS PRESENT
*
C000 DE 2A     RENUM    LDX    NEXTBA    GET FIRST BYTE AFTER BASIC SOURCE
C002 9C 2E           CPX    SOURCE    SAME AS FIRST BYTE OF BASIC SOURCE?
C004 26 03           BNE    RENUM0     IF NOT, CO RENUMBER
C006 7E 01 03     RENU00 JMP    RESTRT   BACK TO BASIC WARM START
*
*START PASS ONE
*BUILD TABLE OF LINE NUMBERS FOLLOWING
*THE SOURCE PROGRAM IN MEMORY
*EACH ENTRY CONSISTS OF FOUR BYTES
*THE FIRST TWO ARE THE OLD LINE NO. IN BCD
*THE NEXT TWO ARE THE NEW LINE NO. IN BCD
*
C009 FF C2 C5     RENUM0 STX    TEMP3    POINTER TO END OF LINE NUMBER TABLE
C00C 4F           CLR A              INIT LINE NO COUNTER
C00D 97 32           STA A  MSLINE
C00F 97 33           STA A  LSLINE
C011 DE 2E           LDX    SOURCE    POINT TO SOURCE START
C013 A6 00     RENUM1 LDA A  0,X      GET LINE NO.
C015 E6 01           LDA B  1,X
C017 FF C2 C3       STX    TEMP2     SAVE SOURCE POINTER
C01A FE C2 C5       LDX    TEMP3     GET LINE NO. TABLE POINTER
C01D A7 00           STA A  0,X      PUT LINE NO. IN TABLE
C01F E7 01           STA B  1,X
C021 08             INX              BUMP THE POINTER
C022 08             INX              AND AGAIN
C023 96 33           LDA A  LSLINE    GET LSB OF LAST NEW LINE NO.
C025 8B 0A           ADD A  #10      ADD TEN TO IT
C027 19             DAA              BE SURE IT IS DECIMAL
C028 97 33           STA A  LSLINE    PUT BACK NEW LSD OF LINE NO.
C02A 96 32           LDA A  MSLINE    GET MSB OF LAST NEW LINE NO.
C02C 89 00           ADC A  #0       ADD CARRY TO IT
C02E 19             DAA              BE SURE IT IS DECIMAL
C02F 97 32           STA A  MSLINE    PUT BACK NEW MSB OF LINE NO.
C031 D6 33           LDA B  LSLINE    GET LSB OF NEW LINE NO.
C033 A7 00           STA A  0,X      PUT NEW LINE NO. IN TABLE
C035 E7 01           STA B  1,X
C037 08             INX              BUMP END OF TABLE POINTER
C038 08             INX
C039 FF C2 C5       STX    TEMP3     SAVE IT
C03C FE C2 C3       LDX    TEMP2     GET SOURCE POINTER
C03F 9C 2A     RENUM3 CPX    NEXTBA    ARE WE DONE?
C041 27 14           BEQ    RENUM4    IF YES, DO SECOND PASS
C043 08             INX              BUMP THE SOURCE POINTER

```

C044 A6 00	LDA A	0,X	GET A BYTE OF SOURCE
C046 26 F7	BNE	RENUM3	BRANCH IF NOT END OF LINE
C048 FF C2 C7	STX	TEMP4	SAVE SOURCE POINTER
C04B F6 C2 C4	LDA B	TEMP2+1	GET OLD SOURCE POINTER+1
C04E 5C	INC B		ADD 1 TO IT
C04F F1 C2 C8	CMP B	TEMP4+1	CHECK FOR LINE NO. ENDING IN 00
C052 27 EB	BEQ	RENUM3	
C054 08	INX		BUMP THE SOURCE POINTER
C055 20 BC	BRA	RENUM1	REPEAT
* *START PASS TWO *			
C057 FE C2 C5	RENUM4	LDX	TEMP3
C05A 09	DEX		GET END OF TABLE POINTER
C05B 09	DEX		DECREMENT THE END OF TABLE POINTER TO
C05C 09	DEX		POINT TO ACTUAL END OF TABLE
C05D 09	DEX		
C05E FF C2 C5	STX	TEMP3	SAVE END OF TABLE POINTER
C061 DE 2E	LDX	SOURCE	GET POINTER TO SOURCE BEGINNING
* *REPLACE BCD LINE NUMBER HERE *			
C063 9C 2A	RENU45	CPX	NEXTRA
C065 27 9F	BEQ	RENU00	DONE?
C067 FF C2 C9	STX	TEMP5	SAVE POINTER TO START OF SOURCE LINE
C06A A6 00	LDA A	0,X	GET LINE NO. FROM LINE
C06C E6 01	LDA B	1,X	
C06E B7 C2 C1	STA A	TEMP1	SAVE IT
C071 F7 C2 C2	STA B	TEMP1+1	
C074 FF C2 C3	STX	TEMP2	SAVE SOURCE POINTER
C077 DE 2A	LDX	NEXTBA	POINT TO START OF LINE NO. TABLE
C079 A6 00	RENUM5	LDA A	0,X
C07B B1 C2 C1	CMP A	TEMP1	GET MSB OF LINE NO. FROM TABLE
C07E 27 06	BEQ	RENUM6	IS IT THE ONE WE WANT?
C080 08	RENU55	INX	NO. POINT TO NEXT ENTRY
C081 08	INX		
C082 08	INX		
C083 08	INX		
C084 20 F3	BRA	RENUM5	REPEAT
C086 E6 01	RENUM6	LDA B	1,X
C088 F1 C2 C2	CMP B	TEMP1+1	GET LSB OF LINE NO. FROM TABLE
C08B 26 F3	BNE	RENU55	IS IT THE ONE WE WANT?
C08D A6 02	LDA A	2,X	IF NOT REPEAT
C08F E6 03	LDA B	3,X	YES. GET NEW NO. FROM TABLE
C091 FE C2 C3	LDX	TEMP2	GET SOURCE POINTER
C094 A7 00	STA A	0,X	PUT NEW NO. IN TABLE
C096 E7 01	STA B	1,X	
C098 08	INX		BUMP THE SOURCE POINTER
C099 08	INX		
C09A A6 00	LDA A	0,X	GET MSB OF KEYBYTE FROM SOURCE
C09C E6 01	LDA B	1,X	GET LSB OF KEYBYTE FROM SOURCE
C09E 08	INX		BUMP THE SOURCE POINTER
C09F 08	INX		
* *LOOK FOR TOKEN FOR GOTO & GOSUB *THESE ARE \$0237 & \$0230 *			
COA0 81 02	CMP A	#2	IS MSB THE ONE WE WANT?
COA2 26 08	BNE	RENU65	
COA4 C1 30	CMP B	#\$30	IS IT A GOSUB?

COA6 27 07		BEQ	GO	
COA8 C1 37		CMP B	#\$37	IS IT A GOTO?
COAA 27 03		BEQ	GO	
COAC 7E C1 FD	RENU65	JMP	ON	
*				
*REPLACE OLD ASCII LINE NUMBER WITH NEW ASCII				
*LINE NUMBER AND DELETE LEADING ZEROS				
*				
COAF 86 FF	GO	LDA A	#\$FF	PRESET ASCII LINE NO STORAGE
COB1 B7 C2 CB		STA A	ASTM1	MSB
COB4 B7 C2 CC		STA A	ASTM2	
COB7 B7 C2 CD		STA A	ASTM3	
COBA B7 C2 CE		STA A	ASTM4	LSB
COBD 08	GO1	INX		INC. POINT, SKIP SPACE, GET BYTE
COBE A6 00		LDA A	0,X	
COC0 27 13		BEQ	GO2	BRANCH ON END OF LINE
COC2 81 2C		CMP A	#\$2C	IS IT A COMMA?
COC4 27 0F		BEQ	GO2	
COC6 81 3A		CMP A	#\$3A	IS IT A COLON?
COC8 27 0B		BEQ	GO2	
COCA 81 20		CMP A	#\$20	IS IT A SPACE?
COCC 27 07		BEQ	GO2	
COCE 8D 7B		BSR	TSTNUM	TEST FOR NUMERIC CHAR.
COD0 24 EB		BCC	GO1	BRANCH IF NUMERIC
COD2 7E C2 00		JMP	ON1	
COD5 8D 71	GO2	BSR	DEXGET	DEC POINTER + GET BYTE
COD7 B7 C2 CE		STA A	ASTM4	PUT IN ASCII TEMP
CODA 8D 6C		BSR	DEXGET	GET ANOTHER
CODC 25 12		BCS	GO3	BRANCH IF NOT NUMERIC
CODE B7 C2 CD		STA A	ASTM3	PUT IN ASCII TEMP
COE1 8D 65		BSR	DEXGET	
COE3 25 0B		BCS	GO3	
COE5 B7 C2 CC		STA A	ASTM2	
COE8 8D 5E		BSR	DEXGET	
COEA 25 04		BCS	GO3	
COEC B7 C2 CB		STA A	ASTM1	
COEF 09		DEX		DECREMENT SOURCE POINTER
COFO 08	GO3	INX		BUMP SOURCE POINTER
COF1 FF C2 C3		STX	TEMP2	SAVE SOURCE POINTER
COF4 8D 58		BSR	ASCON	RETURN WITH NEW LINE NO. IN ASCII
COF6 FE C2 C3		LDX	TEMP2	GET SOURCE POINTER
COF9 B6 C2 CB		LDA A	ASTM1	GET MSB ON NEW NO.
COFC 81 30		CMP A	#\$30	IS IT ZERO?
COFE 26 03		BNE	GO35	
C100 BD C2 83		JSR	CLOSUP	DELETE IT IF ZERO
C103 81 FF	GO35	CMP A	#\$FF	IS IT NOT USED?
C105 27 03		BEQ	GO4	
C107 A7 00		STA A	0,X	PUT IT IN SOURCE
C109 08		INX		BUMP THE SOURCE POINTER
C10A B6 C2 CC	GO4	LDA A	ASTM2	GET NEXT DIGIT
C10D 81 30		CMP A	#\$30	IS IT ZERO?
C10F 26 0E		BNE	GO45	
C111 F6 C2 CB		LDA B	ASTM1	GET MS DIGIT
C114 C1 30		CMP B	#\$30	IS IT ALSO A ZERO?
C116 27 04		BEQ	GO44	
C118 C1 FF		CMP B	#\$FF	IS MS DIGIT NOT USED?
C11A 26 03		BNE	GO45	
C11C BD C2 83	GO44	JSR	CLOSUP	DELETE LEADING ZEROS
C11F 81 FF	GO45	CMP A	#\$FF	IS DIGIT NOT USED?

C121 27 03	BEQ	GO5	
C123 A7 00	STA A	0,X	PUT DIGIT IN SOURCE
C125 08	INX		BUMP THE SOURCE POINTER
C126 B6 C2 CD GO5	LDA A	ASTM3	GET NEXT DIGIT
C129 A7 00	STA A	0,X	PUT IT IN SOURCE
C12B 08	INX		BUMP THE SOURCE POINTER
C12C B6 C2 CE GO6	LDA A	ASTM4	GET LS DIGIT
C12F BD C2 5B	JSR	TSTOVF	GOT TEST FOR OVERFLOW
C132 A7 00	STA A	0,X	PUT LS DIGIT IN SOURCE
C134 BD C2 BC	JSR	SKIPSP	SKIP SPACES+GET CHAR.
C137 81 2C	CMP A	#\$2C	IS IT A COMMA?
C139 27 0A	BEQ	GO8	
C13B 4D	TST A		IS IT END OF LINE?
C13C 26 04	BNE	GO7	
C13E 08	INX		BUMP THE SOURCE POINTER
C13F 7E C0 63	JMP	RENU45	DO NEXT LINE
C142 7E C1 FD GO7	JMP	ON	
C145 7E C0 AF GO8	JMP	GO	
C148 09	DEXGET	DEX	DECREMENT SOURCE POINTER
C149 A6 00	LDA A	0,X	GET A BYTE OF SOURCE
C14B 7E C2 A7	TSTNUM	JMP TSTNO	GO TEST FOR NUMERIC
*			
*CONVERT ASCII AT ASTM1 THRU 4 TO BCD AT TEMP1			
*			
C14E CE 00 00	ASCON	LDX	#0
C151 FF C2 C1	STX	TEMP1	CLEAR THE INDEX REG
C154 CE C2 C1	LDX	#TEMP1	CLEAR TEMPORARY
C157 B6 C2 CE	LDA A	ASTM4	GET LSB OF ASCII LINE NO
C15A 84 0F	AND A	#\$F	STRIP OFF ASCII BIAS
C15C A7 01	STA A	1,X	SAVE IT
C15E B6 C2 CD	LDA A	ASTM3	GET NEXT ASCII DIGIT
C161 81 FF	CMP A	#\$FF	IS IT USED?
C163 27 26	BEQ	FNDLIN	
C165 84 0F	AND A	#\$F	STRIP OFF ASCII BIAS
C167 48	ASL A		SHIFT TO LEFT SIDE OF BYTE
C168 48	ASL A		
C169 48	ASL A		
C16A 48	ASL A		
C16B AB 01	ADD A	1,X	PACK THE BYTE
C16D A7 01	STA A	1,X	SAVE IT
C16F B6 C2 CC	LDA A	ASTM2	GET NEXT ASCII DIGIT
C172 81 FF	CMP A	#\$FF	IS IT USED?
C174 27 15	BEQ	FNDLIN	
C176 84 0F	AND A	#\$F	
C178 A7 00	STA A	0,X	
C17A B6 C2 CB	LDA A	ASTM1	
C17D 81 FF	CMP A	#\$FF	
C17F 27 0A	BEQ	FNDLIN	
C181 84 0F	AND A	#\$F	
C183 48	ASL A		
C184 48	ASL A		
C185 48	ASL A		
C186 48	ASL A		
C187 AB 00	ADD A	0,X	
C189 A7 00	STA A	0,X	
*			
*SEARCH TABLE FOR LINE NO			
*STORED AT TEMP1			
*AND PUT NEW NO AT TEMP1			
*			

THE TERMINAL-



Until recently all terminal functions were designed with hardware logic. A relatively simple terminal with limited functions could easily require as many as sixty or more integrated circuits. More sophisticated terminals with a moderate amount of intelligence could easily have over a hundred IC's. All this has now changed. With the introduction of MOS video controller circuits it has become possible to design a terminal using a controller and a microprocessor that will perform almost any imaginable function with software. The CT-82 has one hundred twenty-eight separate functions—all of which are software driven. It contains fewer parts than most "dumb" terminals.

The normal screen format is 16 lines (20 lines selectable) with 82 characters per line. This is an upper-lower case display with a 7 x 12 dot matrix. The high resolution characters are displayed on a Motorola Data Products M-2000 series monitor with a green P-31 phosphor. This monitor has a 12 MHz video bandwidth and dynamic focus circuits to insure a crisp well focused display over the entire face of the tube. An alternate all capital letter format is available (optional) with 16, 20 or 22 lines and 92 characters per line. The lower case portion of this character set has graphic symbols. In this mode the lines may be moved together to give a solid figure or line. Direct cursor addressing combined with the plotting capability makes it possible to indicate the end points of a line and then to automatically draw a line between them.

Both the monitor and the character generator have sockets provided for alternate material in the form of an EPROM. This

makes it possible to have special terminal functions, or character sets that can be switched in under computer control.

The CT-82 has its own internal editing functions. This allows inserting and deleting lines and characters, erasing quadrants, or lines; doing rolls, scrolls, slides and other similar functions. The CT-82 can block transmit completed material to the computer, or output material to its own remote printer through the built-in parallel printer I/O port. The terminal can be programmed to operate at any system baud rate that is normally used from 50 to 38,400. The baud rate may be changed at any time within this range with a software command.

The cursor position, type of cursor, cursor ON-OFF and blinking are all provided. A command is provided to print control characters and also to turn on and off a tape punch, or tape reader. Protected fields, shift inversion, dual intensity and many other miscellaneous features make the CT-82 one of the most flexible terminals available.

A fifty-six key alphanumeric keyboard plus a twelve key cursor pad is standard. A numeric pad may be substituted for the cursor pad (optional). Connection to the terminal is through a standard DB-25 connector and RS-232 signal levels. The CT-82 operates from 100, 115, 220, or 240 VAC at 50 to 60 Hz. It weighs 20 lbs, and is a compact 18" wide, 10" high and 18" deep.

CT-82 Intelligent Terminal

assembled and tested . . . \$795.00 F.O.B. San Antonio



SOUTHWEST TECHNICAL PRODUCTS CORPORATION

219 W. Rhapsody

San Antonio, Texas 78216

(512) 344-0241

THE EDITOR-

The only microprocessor editor with all the features and ease of use normally found only on large machines. "THE EDITOR" lets you fully use the CT-82's capabilities.

LINE POINTER —Now you understand why the CT-82 has 82 columns. The left two columns are used for a line pointer, which indicates the line of text being edited.

FILE WRAPAROUND—"THE EDITOR" may make multiple passes over the file being edited without restarting the editor.

AUTOMATIC CARRIAGE RETURN—The last word in a line will automatically be started on the next line if it will not fit in the space remaining on the line.

SIMPLE COMMANDS—Commands consists of a single letter, or a key press on the cursor pad. No complicated format to be learned and remembered.

MULTIPLE COMMANDS and REPEATS—Command line may have more than one command. "THE EDITOR" will execute command strings sequentially. Repeat function allows changes in a string through the text file.

SOURCE TEXT TABS—Tab stops appropriate for source text input may be set to operate from the space bar, or any other key.

SHIFT INVERSION—The keyboard may be set to produce either capital, or lower case letters when shift is used.

SCREEN POSITIONING—Scroll up, scroll down, line pointer up, line pointer down, home file, top of memory, bottom of memory, move relative to pointed line and form feed are provided.

"THE EDITOR" is available only for Southwest Technical Products computer systems using the CT-82 and running under FLEX-5®, or FLEX-8® operating systems. It may be used to edit any files, or programs compatible with the DOS, except binary files. Edited files are compatible with the TSC Text Processing program. The combination makes a powerful and inexpensive word processing system.

Editor FLEX-5® or FLEX-8® \$25.00 ppd. in Continental USA

®FLEX is a registered trademark of TSC Inc.



SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. Rhapsody
San Antonio, Texas 78216
(512) 344-0241

C18B DE 2A	FNDLIN	LDX	NEXTBA	GET START OF TABLE POINTER
C18D BC C2 C5	FNDLI1	CPX	TEMP3	REACHED END OF TABLE?
C190 26 05		BNE	FNDLI1	
C192 C6 07		LDA B	#7	ERROR
C194 7E 0C E9		JMP	BASERR	GO REPORT
C197 A6 00	FNDLI1	LDA A	0,X	GET MSB OF NO FROM TABLE
C199 B1 C2 C1		CMP A	TEMP1	IS IT THE ONE WE WANT?
C19C 27 06		BEQ	FNDLI2	
C19E 08	FNDLI5	INX		POINT TO NEXT ENTRY
C19F 08		INX		
C1A0 08		INX		
C1A1 08		INX		
C1A2 20 F9		BRA	FNDLI1	REPEAT
C1A4 E6 01	FNDLI2	LDA B	1,X	GET LSB OF NO. FROM TABLE
C1A6 F1 C2 C2		CMP B	TEMP1+1	IS IT THE ONE WE WANT
C1A9 26 F3		BNE	FNDLI5	
C1AB EE 02		LDX	2,X	GET NEW NO. FROM TABLE
C1AD FF C2 C1		STX	TEMP1	SAVE IT

*

*CONVERT BCD LINE NUMBER AT TEMP1 TO

*ASCII AT ASTM1 THRU 4

*

C1B0 B6 C2 C1	BCDCON	LDA A	TEMP1	GET MSB
C1B3 16		TAB		SAVE IT
C1B4 84 F0		AND A	#\$F0	STRIP OFF MS NYBBLE
C1B6 26 08		BNE	BCDCH1	IS IT ZERO?
C1B8 86 FF		LDA A	#\$FF	SET NOT USED FLAG
C1BA B1 C2 CB		CMP A	ASTM1	IS IT ALREADY SET?
C1BD 27 0A		BEQ	BCDC01	
C1BF 4F		CLR A		SET TO ZERO, NOT FF
C1C0 44	BCDCH1	LSR A		SHIFT TO RIGHT NYBBLE
C1C1 44		LSR A		
C1C2 44		LSR A		
C1C3 44		LSR A		
C1C4 8B 30		ADD A	#\$30	ADD ASCII BIAS
C1C6 B7 C2 CB		STA A	ASTM1	PUT IN ASCII TEMP
C1C9 17	BCDC01	TBA		GET MSB OF BCD LINE NO
C1CA 84 0F		AND A	#\$F	STRIP OFF MS NYBBLE
C1CC 26 08		BNE	BCDCH2	IS IT ZERO
C1CE 86 FF		LDA A	#\$FF	SET NOT USED FLAG
C1D0 B1 C2 CC		CMP A	ASTM2	IS IT ALREADY SET
C1D3 27 06		BEQ	BCDC02	
C1D5 4F		CLR A		NO. SET IT TO ZERO
C1D6 8B 30	BCDCH2	ADD A	#\$30	ADD ASCII BIAS
C1D8 B7 C2 CC		STA A	ASTM2	SAVE IT
C1DB B6 C2 C2	BCDC02	LDA A	TEMP1+1	GET LSB OF BCD LINE NO.
C1DE 16		TAB		
C1DF 84 F0		AND A	#\$F0	
C1E1 26 08		BNE	BCDCH3	
C1E3 86 FF		LDA A	#\$FF	
C1E5 B1 C2 CD		CMP A	ASTM3	
C1E8 27 0A		BEQ	BCDC03	
C1EA 4F		CLR A		
C1EB 44	BCDCH3	LSR A		
C1EC 44		LSR A		
C1ED 44		LSR A		
C1EE 44		LSR A		
C1EF 8B 30		ADD A	#\$30	
C1F1 B7 C2 CD		STA A	ASTM3	

C1F4 17	BCDC03	TBA	
C1F5 84 0F		AND A	#\$F
C1F7 8B 30		ADD A	#\$30
C1F9 B7 C2 CE		STA A	ASTM4
C1FC 39		RTS	

*

*LOOK FOR ASCII KEYWORDS THEN,GOTO, &GOSUB

*AND/OR END OF LINE

*

C1FD BD C2 BC	ON	JSR	SKIPSP	SKIP SPACES
C200 81 54	ON1	CMP A	#\$54	IS CHAR A T
C202 27 0B		BEQ	THEN	
C204 81 47	ON2	CMP A	#\$47	IS CHAR A G
C206 27 28		BEQ	GOTO	
C208 4D	ON3	TST A		END OF LINE?
C209 26 F2		BNE	ON	
C20B 08		INX		BUMP THE SOURCE POINTER
C20C 7E CO 63	ON4	JMP	RENU45	DO NEXT LINE
C20F 08	THEN	INX		BUMP THE SOURCE POINTER
C210 A6 00		LDA A	0,X	GET A CHARACTER
C212 81 48		CMP A	#\$48	IS IT AN H
C214 26 EE		BNE	ON2	
C216 08		INX		BUMP THE SOURCE POINTER
C217 A6 00		LDA A	0,X	GET A CHARACTER
C219 81 45		CMP A	#\$45	IS IT AN E
C21B 26 E7		BNE	ON2	
C21D 08		INX		BUMP THE SOURCE POINTER
C21E A6 00		LDA A	0,X	GET A CHARACTER
C220 81 4E		CMP A	#\$4E	IS IT AN N
C222 26 E0		BNE	ON2	
C224 BD C2 BC	THEN1	JSR	SKIPSP	GO SKIP SPACES
C227 BD C2 A7		JSR	TSTNO	GO TEST FOR NUMERIC
C22A 25 D8		BCS	ON2	BRANCH IF NOT
C22C 09		DEX		DEC THE SOURCE POINTER
C22D 7E CO AF		JMP	GO	GO RENUMBER IT
C230 08	GOTO	INX		BUMP THE SOURCE POINTER
C231 A6 00		LDA A	0,X	GET A CHARACTER
C233 81 4F		CMP A	#\$4F	IS IT AN O
C235 26 D1		BNE	ON3	
C237 08		INX		BUMP THE SOURCE POINTER
C238 A6 00		LDA A	0,X	GET A CHARACTER
C23A 81 53		CMP A	#\$53	IS IT AN S
C23C 27 0D		BEQ	GOSUB	
C23E 81 54		CMP A	#\$54	IS IT A T
C240 26 C6		BNE	ON3	
C242 08		INX		BUMP THE SOURCE POINTER
C243 A6 00		LDA A	0,X	GET A CHARACTER
C245 81 4F		CMP A	#\$4F	IS IT AN O
C247 27 DB		BEQ	THEN1	
C249 20 BD		BRA	ON3	ALL TESTS FAIL, REPEAT
C24B 08	GOSUB	INX		BUMP THE SOURCE POINTER
C24C A6 00		LDA A	0,X	GET A CHARACTER
C24E 81 55		CMP A	#\$55	IS IT A U
C250 26 B6		BNE	ON3	
C252 08		INX		BUMP THE SOURCE POINTER
C253 A6 00		LDA A	0,X	GET A CHARACTER
C255 81 42		CMP A	#\$42	IS IT A B
C257 27 CB		BEQ	THEN1	
C259 20 AD		BRA	ON3	ALL TESTS FAIL, REPEAT

*
 *TEST FOR OVERFLOW WHEN PUTTING NEW ASCII
 *LINE NO IN SOURCE, OPEN A HOLE FOR IT
 *IF NECESSARY
 *

C25B 36	TSTOVF	PSH A	SAVE CHARACTER
C25C A6 00		LDA A 0,X	GET A CHARACTER FROM SOURCE
C25E BD C2 A7		JSR TSTNO	GO TEST FOR NUMERIC
C261 24 1E		BCC MOVRTS	IF IT IS, DONE
C263 FF C2 C7		STX TEMP4	SAVE THE POINTER
C266 FE C2 C9		LDX TEMP5	GET START OF LINE POINTER
C269 6C 04		INC 4,X	ADD 1 TO LINE BYTE COUNT
C26B DE 2A		LDX NEXTBA	GET ADR OF FIRST BYTE AFTER SOURCE
C26D 08		INX	BUMP IT
C26E DF 2A		STX NEXTBA	SAVE IT
C270 FE C2 C5		LDX TEMP3	GET ADR OF END OF TABLE
C273 08		INX	BUMP IT
C274 FF C2 C5		STX TEMP3	SAVE IT
C277 09	MOVE1	DEX	DECREMENT THE POINTER
C278 A6 00		LDA A 0,X	GET A CHARACTER
C27A A7 01		STA A 1,X	MOVE IT UP ONE BYTE
C27C BC C2 C7		CPX TEMP4	DONE???
C27F 26 F6		BNE MOVE1	
C281 32	MOVRTS	PUL A	RESTORE A
C282 39		RTS	

*
 *CLOSE HOLE MADE BY DELETEDING LEADING ZEROS
 *

C283 FF C2 C7	CLOSUP	STX TEMP4	SAVE POINTER
C286 A6 01	CLOSUI	LDA A 1,X	GET A BYTE
C288 A7 00		STA A 0,X	MOVE IT DOWN ONE
C28A 08		INX	BUMP THE POINTER
C28B BC C2 C5		CPX TEMP3	DONE???
C28E 26 F6		BNE CLOSUI	
C290 FE C2 C5		LDX TEMP3	GET END OF TABLE POINTER
C293 09		DEX	SUBTRACT 1
C294 FF C2 C5		STX TEMP3	SAVE THE NEW POINTER
C297 FE C2 C9		LDX TEMP5	GET START OF LINE POINTER
C29A 6A 04		DEC 4,X	SUBTRACT ONE FROM LINE BYTE COUNT
C29C DE 2A		LDX NEXTBA	GET POINTER TO FIRST BYTE AFTER SOURCE
C29E 09		DEX	SUBTRACT ONE
C29F DF 2A		STX NEXTBA	SAVE NEW POINTER
C2A1 4F		CLR A	SET A TO
C2A2 4A		DEC A	\$FF
C2A3 FE C2 C7		LDX TEMP4	GET POINTER
C2A6 39		RTS	DONF

*
 *TEST FOR NUMERIC CHARACTER
 *

C2A7 36	TSTNO	PSH A	
C2A8 81 30		CMP A #'0	
C2AA 2B 04		BMI TSTNO1	
C2AC 81 39		CMP A #'9	
C2AE 2F 03		BLE TSTNO2	
C2B0 0D	TSTNO1	SEC	
C2B1 32		PUL A	
C2B2 39		RTS	
C2B3 0C	TSTNO2	CLC	
C2B4 32		PUL A	
C2B5 39		RTS	

```

      *
      *SKIP SPACES
      *
C2B6 A6 00  SKIP1  LDA A  0,X
C2B8 81 20          CMP A  $$20
C2BA 26 03          BNE   SKIP2
C2BC 08          SKIPSP INX
C2BD 20 F7          BRA   SKIP1
C2BF 4D          SKIP2  TST A
C2C0 39          RTS
C2C1          TEMP1  RMB   2
C2C3          TEMP2  RMB   2
C2C5          TEMP3  RMB   2
C2C7          TEMP4  RMB   2
C2C9          TEMP5  RMB   2
C2CB          ASTM1  RMB   1
C2CC          ASTM2  RMB   1
C2CD          ASTM3  RMB   1
C2CE          ASTM4  RMB   1
                        END      RENUM

```

NO ERROR(S) DETECTED

TIME PROMPTS FOR FLEX[®]

R. Dembinak
12 Richard Rd.
Medway, MA 02053

I recently purchased the JPC Products CK-7 clock board for my SWTPc 6800 system. First, I am very pleased with JPC's response to my order, as UPS attempted delivery only 9 days after the day I mailed it. The product was as advertised and even included better than average documentation. The board was built and completely tested within a couple of hours. That time included keying in the source for the program that was supplied to test the clock board.

So, now that I had a clock on my system, what was I going to do with it? Well, since I run with Mini Flex, the first thing I decided to do was to extend the "+++" prompt to include the time of day, like some advanced systems. The result is the attached patch to Mini Flex. This patch develops a prompt of the form:

+++HHMM SS

which is then followed by a carriage return and line feed. The patch assumes the clock board to be on port 4 and that it has been properly initialized for use by the get digit routine. That initialization should include setting the clock to the current time of day. If RESET is used at any

time, the port must be reinitialized before the prompt will display a valid time. The following routine accomplishes this:

```

CLR      PORT+1
CLR      PORT+3
LDA A    #$1C
STA A    PORT+2
LDA A    #4
STA A    PORT+1
STA A    PORT+3

```

The area used for this patch wasn't large enough to have this routine as part of it, so its function must be provided elsewhere.

Since this patch is always resident, assembly language and BASIC program could make use of it. Essentially, there are two useful routines called by:

```

JSR $7EA6      print time on console
JSR $7EB1      get and save current time

```

The stored time is kept in packed hex format.

With BASIC resident and setting location \$0067 to \$7EA6, the the statement `Z9=USER(0)` will cause the time to be typed out.

Modification of the patch for use with Flex 1.0 or Flex 2.0 should be a matter of finding an unused area within them and changing a few service routine addresses. Also, the interrupt portion of the CK-7 board could probably be used to drive the print spooling feature of these systems. Maybe someone can look into this modification. Good clocking.

```

+++1320 23
P,CAT 1
+++1320 57
(carriage return)
+++1321 05

```

Simulated example of prompts with time.

```

                                NAM      SPATCH
*
* PATCHES TO MINI FLEX VER 1.0
*
* BY R. DEMBINSKI
*
EOCA      PHEX      EQU      $EOCA      USE MIK/SWT BUG

```

EOC8		PHX2	EQU	\$EOC8	
		*			
7118		PSTRNG	EQU	\$7118	
711E		PCRLF	EQU	\$711E	
		*			
8010		PORTA	EQU	\$8010	
8011		CRTIA	EQU	PORTA+1	
8012		PORTB	EQU	PORTA+2	
8013		CRTLB	EQU	PORTA+3	
		*			
7EFC		HOUR	EQU	\$7EFC	
7EFD		MIN	EQU	HOUR+1	
7EFE		SEC	EQU	HOUR+2	
		*			
		*			
			ORG	\$718E	
718E BD 7E A3			JSR	NEWPT	ALTER PROMPT WRITE
7EA3			ORG	\$7EA3	NEW CODE
7EA3 BD 71 18	NEWPT		JSR	PSTRNG	COMPLETE PROMPT
7EA6 8D 09			BSR	TIME	GET TIME
7EA8 BD E0 C8			JSR	PHX2	EDIT TIME ALWAYS
7EAB BD E0 CA			JSR	PHEX	ON CONSOLE
7EAE 7E 71 1E			JMP	PCRLF	
7EB1 86 18	TIME		LDA A	#24	GET SEC
7EB3 CE 7E FC			LDX	#HOUR	SET INDEX
7EB6 5F			CLR B		MBZ ON INITIAL RUN
7EB7 8D 24			BSR	GETD	
7EB9 36			PSH A		SAVE SEC
		*			
		*			
7EBA 86 1C			LDA A	#\$1C	TEN HOUR
7EBC 8D 1F			BSR	GETD	
7EBE 86 0C			LDA A	#\$0C	HOUR
7ECO 8D 1B			BSR	GETD	
7EC2 08			INX		SET X FOR MIN
7EC3 86 14			LDA A	#\$14	TEN MIN
7EC5 8D 16			BSR	GETD	
7EC7 86 10			LDA A	#\$10	MIN
7EC9 8D 12			BSR	GETD	
7ECB 08			INX		SET X FOR SEC
7ECC 86 08			LDA A	#8	TEN SEC
7ECE 8D 0D			BSR	GETD	
7ED0 86 18			LDA A	#24	SEC
7ED2 8D 09			BSR	GETD	
7ED4 84 0F			AND A	#\$0F	GET SEC
7ED6 33			PUL B		GET INITIAL SEC
7ED7 10			SBA		SUB FROM A B
7ED8 25 D7			BCS	TIME	REDO IF CARRY SET
7EDA 09			DEX		LEAVE INDEX AT HOUR
7EDB 09			DEX		
7EDC 39			RTS		
		*			
7EDD 37		GETD	PSH B		PRESERVE B
7EDE F6 80 12			LDA B	PORTB	
7EE1 C4 E3			AND B	#\$E3	
7EE3 1B			ABA		
7EE4 B7 80 12			STA A	PORTB	SET FOR DESIRED DIGIT
7EE7 33			PUL B		
7EE8 B6 80 10			LDA A	PORTA	
7EEB 84 0F			AND A	#\$0F	GET LOW 4 BITS

	*		
7EED 58	ASL B	MOVE RIGHT 4	
7EEE 58	ASL B	TO LEFT 4	
7EEF 58	ASL B		
7EF0 58	ASL B		
7EF1 1B	ABA	GENERATE AND SAVE	
7EF2 A7 00	STA A 0,X	HH, MM OR SS	
7EF4 16	TAB	MOVE RESULT TO B	
7EF5 39	RTS		
	END		

A REVIEW OF SOME 6800 MONITORS

Robert C. Boyd
Woodlawn Ave., RFD 2
Kennebunkport, ME 04048

One of the most important features of Motorola 6800 systems is the use of read-only-memory (ROM) to provide a permanently stored monitor program. This feature enables the operator to enter commands through the control terminal immediately upon power-up without first keying-in a routine as on other systems.

Early M6800 systems utilized the MIKBUG¹ monitor and the 6820 Peripheral Interface Adapter (PIA) which connected the control terminal (usually a Teletype or video display device) to the microcomputer. Most user programs and commercial software were written to branch to MIKBUG subroutines to accomplish input or output operations; this reduced both the task of writing a program as well as the amount of memory required at execution time.

When Motorola announced the 6850 Asynchronous Communications Interface Adapter (ACIA), manufacturers and users quickly recognized its advantages over the PIA, particularly for interrupt-driven or high transfer-rate serial I/O operations. (The PIA was designed for 8-bit parallel I/O and not serial I/O. MIKBUG keeps the microcomputer in a timing loop while reading or writing data through the PIA, and, while in the loop, the microcomputer cannot be doing useful work. The maximum transfer rate possible with MIKBUG and the PIA is about 1200 bits per second(bps);

¹ registered trademark of Motorola Inc.

an ACIA is capable of rates in excess of 19,200 bps.) Many M6800 systems now provide an ACIA to connect terminals and other serial devices to the system.

The changeover from PIA to ACIA necessitates that the MIKBUG ROM be replaced with a monitor which will control the ACIA and be somewhat software compatible with MIKBUG in order to minimize changes to software which had used MIKBUG subroutines. Today the M6800 user is presented with a variety of ROM's and EPROM's containing monitors which may or may not be MIKBUG-compatible. Several monitors are designed to facilitate program development by including single-step, memory-dump, and register examine/change commands. There is at least one monitor available which will execute more than one task at a time. Some monitors contain commands to transfer control to user-written routines which would be stored elsewhere in EPROM or RAM, or to disk-based operating systems. The features of four MIKBUG-compatible monitors are presented in this article to assist the user in determining which monitor would be most useful in his M6800 system.

Table I lists the prominent commands and features of the SMARTBUG, SWTBUG², MSIBUG, and RT/68 monitors. The documentation supplied with each monitor contains details on hardware configuration and command usage as well as an assembly listing of the monitor. Each occupies 1024 bytes of area addressed starting at E000, and requires at least 128 bytes of RAM at A000 for work-area and stack. All four monitors contain commands to display registers, to examine and/or to change memory, to load or to punch tape, and to execute a user program. Each has the capability to jump to additional user-written commands which would be stored in EPROM or RAM. The SWTBUG has the feature of being able to direct IO operations to any port addressed from 8000 to 801C. The SWTBUG is available only in ROM; the other three monitors may be obtained in 2708 EPROM, and may therefore be easily customized by the user.

² registered trademark of Southwest Technical Products Corporation.

SWTBUG will communicate with the control terminal through either a PIA or ACIA at 8004; an optional PIA may be used at 8000 for tape load and punch commands. Furthermore, input and output operations may be vectored to other ports by first entering the port address into memory locations A00A-A00B. SWTBUG will then examine that port and configure itself for either a PIA or ACIA. These features facilitate using a video terminal at high transfer rate (or a Teletype at 110 bps) on the control port, a tape cassette system at 300 baud on port 0, and a line printer at yet another transfer rate on a third port. SWTBUG includes a command to home-up and clear the screen on the SWTP CT-1024 terminal, and another command to bootstrap the MF-68 minifloppy disk system. The byte-search command examines memory within the specified range and displays each address containing the specified data; this would be useful during certain phases of program development.

SMARTBUG requires an ACIA at 8008 to communicate with the control terminal. Commands are provided to examine and change the A or B accumulators, the condition code register, and the index register. The insert command enables the user to place a character into one or more memory locations; it would be used to clear memory to blanks or to any character (such as 3F - software interrupt) prior to loading and executing a program. The echo, no-echo, and hard-copy commands allow flexible control over those systems which contain more than one output device where output may be desired on one device but not on both at the same time. The trace command puts the microcomputer into single-step mode where one instruction is executed, the registers are displayed, and opportunity is granted to change the A, B, C, or X registers before continuing. Two commands are included to control the BFD-68 floppy disk system. Another command transfers control to E400 where additional EPROM or RAM may be located to contain the user's favorite programs or subroutines.

MSIBUG will communicate with the control terminal through an ACIA at 8000.

A tape cassette interface may be connected to either port 0 or port 1. As with SWTBUG, this facilitates tape operation at a different transfer rate from that of the control terminal. The lister or print memory command displays a block of memory in instruction format, i. e., one, two, or three byte instructions are displayed one instruction per line. The calculate checksum command will generate a three-byte checksum on the specified area of memory so that a user can easily determine if memory content has been altered by program execution. The commands to permit and inhibit print-out enable a section of a listing to be skipped entirely; another command will cause execution of a user-program to halt.

RT/68 requires either an ACIA at 8000 or a PIA at 8004 to connect the control terminal to the system. The PIA is always required to enable RT/68 to determine interface options, and it may be used to implement a real-time clock or a front panel abort switch. The RT/68 operates in console mode (to load, save, or debug programs), single-task mode (to execute MIKBUG-type software), or multi-task mode (where up to 16 tasks may execute concurrently). A task is defined as "a complete unit of object code that can compete for system resources independently." A comprehensive 80-page manual is included with the RT/68, and considerable detail is provided on multitasking, interrupt processing, and reentrant code.

This article has discussed the features of SWTBUG, SMARTBUG, MSIBUG, and RT/68, four MIKBUG-compatible monitors for the Motorola 6800 systems. Each monitor has a number of unique time-saving commands which will greatly increase the usefulness of a system which is currently using the MIKBUG monitor. The user who hopes someday to expand his system to include an EPROM board or a minifloppy controller can install one of these monitors today knowing that it has the features built in to facilitate that future expansion. Once it is installed, you'll wonder how you ever got along without it!

TABLE 1

Manufacturer	SWTBUG	SMARTBUG	MSIBUG	RT/68
Media	Southwest Technical Products Corp.	Smoke Signal Broadcasting	Midwest Scientific Instruments, Inc.	Microvare Systems Corp
Required interface	ROM	EPROM	EPROM	ROM or EPROM
Task capability	ACIA and/or PIA	ACIA	ACIA	PIA, optional ACIA
Commands and features:	single	single	single	up to 16 tasks
display registers	R	R	R	R
load memory from tape	0 L	L	L n (1)	L
execute program	J aaaa	J aaaa	G aaaa	E, aaaa
save memory onto tape	0 P	P aaaa bbbb (4)	P n aaaa bbbb	P, aaaa, bbbb
write end of file on tape	0 E	(5)	part of P command	-----
memory examine/change	M aaaa	M aaaa	M aaaa	M, aaaa
set breakpoint	B aaaa	K aaaa	-----	B, aaaa
go to program/return	-----	-----	-----	-----
from interrupt	G (5)	G (5)	-----	G
jump to user routine	Z (at C000)	4 (at E4000)	X (at E4000)	ESC (at 7000)
print memory contents	-----	-----	T aaaa bbbb	D, aaaa, bbbb
home and clear CRT	C	-----	-----	-----
disk bootstrap	D	q	-----	-----
byte search	F bbbbaaac	A,B,C, or X	-----	-----
examine/change registers	-----	I aaaa bbbb cc	-----	-----
insert into memory	-----	T aaaa	-----	-----
program trace	-----	-----	C aaaa bbbb	-----
calculate checksum	-----	-----	control S	-----
start/stop output routine	-----	-----	control E	-----
terminate output routine	-----	-----	control D	-----
activate multi-task mode	-----	-----	-----	S
abort current function	-----	-----	-----	optional switch
real-time clock	yes	-----	-----	optional
vectored SWI and I/O	yes	-----	-----	-----
Price	\$19.95	\$39.95 to \$49.95	\$60.00	\$55.00

aaaa = beginning memory address 0 = optional port
 bbbb = ending memory address n = I/O port number
 cc = data X = user defined commands

- 1 optional echo on control terminal
- 2 single-task mode
- 3 beginning address previously entered into A002-A005
- 4 does not write end of file marker
- 5 start address previously entered into A048-A049
- 6 execution may be resumed

The following pictures were taken during our coverage of the Fourth West Coast Computer Faire - San Francisco, California - May 11, 12, 13th 1979. The 6800 crowd was outnumbered by the 'Appliance Crowd' and many 6800 vendors suggested this may be their last. The crowds were estimated at between 5000-7500 for all three days. See you next in Atlanta the 15th - 16th & 17th of June. 68 Micro Journal will be there also. Look for us at our booth #3, just to the right as you come in. See you then.....Don.



The SWTPC 'Gang' Dan Meyer behind the 'Kung-Fu' Moustache flanked by (r) Joe, (l) Gary and Dennis and Ron (software) on the other end.



Motorola's Terry Ritter (6809 pappa) and Noland Lewis (r) at the 'Big Fist' booth.



TSC's Dave Shirk, explaining his software 'goodies' to an attentive faire-goer.



Advanced Computer Product's president, early 6800 device supplier (still is), David Freeman.



Mr. GIMIX himself - Richard Don with a full CHOST display.



Dave Freeman (l) and Dr. Adam Osborne (r) getting it all together.



MicroSys's John Stuppy (r) with Charles Hotchkiss (l) who just bought a 6802 CPU board for his S100 machine.



John Craig - Editor (and good friend) of Creative Computing. Note: Editor's Remarks this issue.



MSI's VP and General Manager Dennis Seager demonstrating the new 'Business' package to a faire-goer.



The SWTPC Winchester Disk, should be available in the next couple months for about four biggies.

MOTOROLA M6809 EMULATOR

RUN 6809 SOFTWARE BEFORE
THE CHIP IS AVAILABLE!

E6809 is a 6800 machine language program that will emulate all of the functions of the Motorola 6809 third generation micro-processor. Developed for use on any 6800 computer system, the program allows software development and debugging prior to 6809 availability. 6809 object code may be placed in the 6800's memory and executed or single-step traced by E6809. The 3K byte program is complete with a 6809 mini-monitor and console I/O routines for ease of use. A fully commented source listing is included. Specify Smoke Signal Broadcasting or FLEX™ disk, or KCS cassette. \$49.95

THE MICRO
WORKS

Master Charge
and
BankAmericard

DEPT K, P.O. BOX 1110, DEL MAR, CA 92014 (714) 754-2687

MAGIC SQUARES

Paul Pennington
CSRA Computer Club
Martinez, GA 30907

A magic square is an array of numbers arranged so that the sum of each row and column, as well as the two diagonals, is the same. An example is shown for order 5. A method of generating a magic square whose order is an odd number is to start with 1 on the middle square of the top row, then proceed numbering up and to the left diagonally (when you run off the edge "wrap around" to the other side) until a filled square is reached. Then drop down one square from the most-recently-filled square and continue. Incidentally, this algorithm was not developed by a computer programmer. It was brought from Siam to France by S. de La Loubere in 1687. (Reference: Knuth, The Art of Computer Programming, Vol 1, p 158)

```
0010 REM MAGIC SQUARES
0020 REM SEE KNUTH, VOL 1, PG 158
0030 PRINT "MAGIC SQUARE GENERATOR"
0040 INPUT "ORDER", D
0050 IF D/2=INT(D/2) THEN PRINT "ODD ORDER ONLY": GOTO 40
0060 DIM M(D,D)
0070 R=1
0080 C=INT(D/2)+1
0090 N=N+1
0100 M(R,C)=N
0105 IF N=D*D THEN 1000
0110 R=R-1
0120 IF R=0 THEN R=R+D
0130 C=C-1
0140 IF C=0 THEN C=C+D
0150 IF M(R,C)≠0 THEN 90
0160 R=R+2: IF R>D THEN R=R-D
0170 C=C+1: IF C>D THEN C=C-D
0180 M(R,C)=N
0190 GOTO 90
1000 FOR R=1 TO D
1010 FOR C=1 TO D
1020 PRINT TAB(3*C); M(R,C);
1030 NEXT C
1040 PRINT
1050 NEXT R
9999 END
```

Sample RUN:

```
MAGIC SQUARE GENERATOR
ORDER? 5
 15 8 1 24 17
 16 14 7 5 23
 22 20 13 6 4
 3 21 19 12 10
 9 2 25 18 11
```

Extracted with THANKS from: CSRA Computer Club Newsletter
Paul Pennington - Editor
Martinez, GA 30907

EDITORS REMARKS

The subject of this column, this month, does not apply to our normal theme of '6800/09 information only'. As you read on you will understand why. Bear in mind that this has not happened to us in the 6800 crowd, but it does not mean it cannot, unless we all keep our guard up.

This is a true but sad tale of what can happen if a magazine does not check out it's advertisers. The losers are not only the rooked buyers but also us as the advertising media and of course the other good guys who honor their advertised promises.

Since we started we have turned down three advertisers. This means thousands in advertising revenue over a period of time. All for the same reason; I felt that their past track record was indicative of shoddy products and service, or promises unkept. The other magazines carry their advertising and we could sure use the advertising revenue. However, from the outset it was my firm intention to insure that we would attempt to protect our readers from unscrupulous advertisers. To this end I pledge my continued vigilance.

All of the advertisers in 68 Micro Journal are known to me personally, either by product use or report of others considered fair in their appraisal. This means that to the best of my knowledge and investigation ALL THOSE ADVERTISERS IN 68 MICRO JOURNAL are dependable and honest. We are very fortunate that the average 6800/09 vendor and manufacturer is as dedicated to your satisfaction as any sales group in the computer field.

I use this space to tell you the following because it seems that we need to be aware, you and I together. Also I want to commend John Craig, Editor of Creative Computing and Bill Godbout for their desire to keep the trade clean. Their actions have saved a lot of unsuspecting micro buyers additional financial loss. That they are competitors or \$100 vendors makes not a quid of difference in this instance; what counts is that we are all members of the human race and as such should have concern for one another.

It seems that a year or so back a company (?) by the name of DataSync started advertising heavily in most of the other computer magazines. It appears that the advertising was accepted by these magazines without much background knowledge of the product or principals involved. DataSync was headed up by a person known as Col. Dave Winthrop. It was discovered later (too late) that Col. Dave Winthrop had also other names. It was also belatedly discovered that Winthrop had headed other shoddy or dishonest operations. Needless to say; by the time his dishonest operation was uncovered, many micro hobbyist had lost a bundle (estimated at over \$100,000.00).

The ads were mockups of other company's products, with the names deleted or covered up and additional devices pasted to the product. The line included everything from video terminals to memory kits. Actually not one item existed. Due to the action of John Craig and others, Winthrop was exposed, arrested and sentenced to prison. While in a California minimum security facility (where he had convinced many officials that he was on the verge of a 'solar energy' breakthrough) he escape.

Act Two: A few months back a new and heavy advertising company called 'World Power Systems' began advertising in nearly all the computer magazines. Believe it or not; the April issues. Not just one or two pages but 4 or 5 full pages, each month, in each magazine. Apparently none of the magazines became suspicious of such a large volume by a new and unknown concern. It is not surprising that by the time it was over they were many thousands of dollars lighter. World Power Systems was headed up by a man by the name of Jim Anderson. Yep, you guess it, Winthrop up to his old tricks again. Seems some folks never learn. Oh, incidentally Winthrop's real name is supposed to be Norman Henry Hunt, but of course he rarely uses it.

Old Winthrop or Hunt or whatever would probably still be gathering it in if it had not been for the action of Bill Godbout of Godbout Electronics and John Craig of Creative Computing. One of the boards displayed in a World Power Systems ad looked suspicious to Bill. Bill got in touch with John and they decided to investigate World Power Systems (a little

late for some, as WPS had collected in about a half million, more or less). John made a 'bluff' call to world power last month (April), stating that he was going to travel to their fine company for a get acquainted visit. This threw them into a panic and they fled, leaving most of the loot behind in their haste. By this time John had come to realize that Winthrop was the moving power of World Power Systems, but the bad guys had saddled up with part of the diggings (also a couple of lassies included) and departed in typical western style. Not on horseback but in a new motor van. End of Act Two.....but not quite all.

All this has a moral; I guess. If so, it has to be something like this: DEAL WITH MERCHANTS WHO YOU KNOW TO BE HONEST (OUR ADVERTISERS FOR INSTANCE). None of us can police the whole industry, but all of us collectively can go a long way in insuring that what has happened to others, will not be our lot, as concerns the above.

The end is not yet. Hunt is still at large, whoopeeing it up at the expense of fellow computer hobbyist, also some red-faced (red-inked also) computer magazines. John and Bill did us all a service, to them we all say 'THANKS'. The whole story is due out in Creative Computing in July. By then maybe Hunt will be apprehended and some of the latest victims will have recouped part of their losses; I sure hope so.

So in ending I will repeat a promise I made to you the first time out. 68 Micro Journal will continue to accept advertising only from those who WE KNOW TO BE HONEST. You can trust our advertisers. If in doubt call me. If you desire information on companies not advertising in 68 Micro Journal, call me. I have a pretty good file on most 6800 vendors, and some that are not. Together we can all enjoy this; the greatest hobby in the world...computing.

DMW

BASIC CASSETTE FORMATS

Phil Schuman
Reprint from '6800 Bits'
Chicago Area 6800 Newsletter

For those that have saved programs on cassette, and need to know if they can use those tapes on other BASIC's. the answer

is a very simplemaybe. It seems that each version is a little different from the next, but for the most part, they all follow a standard set when the 1st SWTPC BASIC came out. As you know, the data on the cassette is nothing more than a listing of the program. This is true, but there are a few more 'control characters' on the tape to help out.

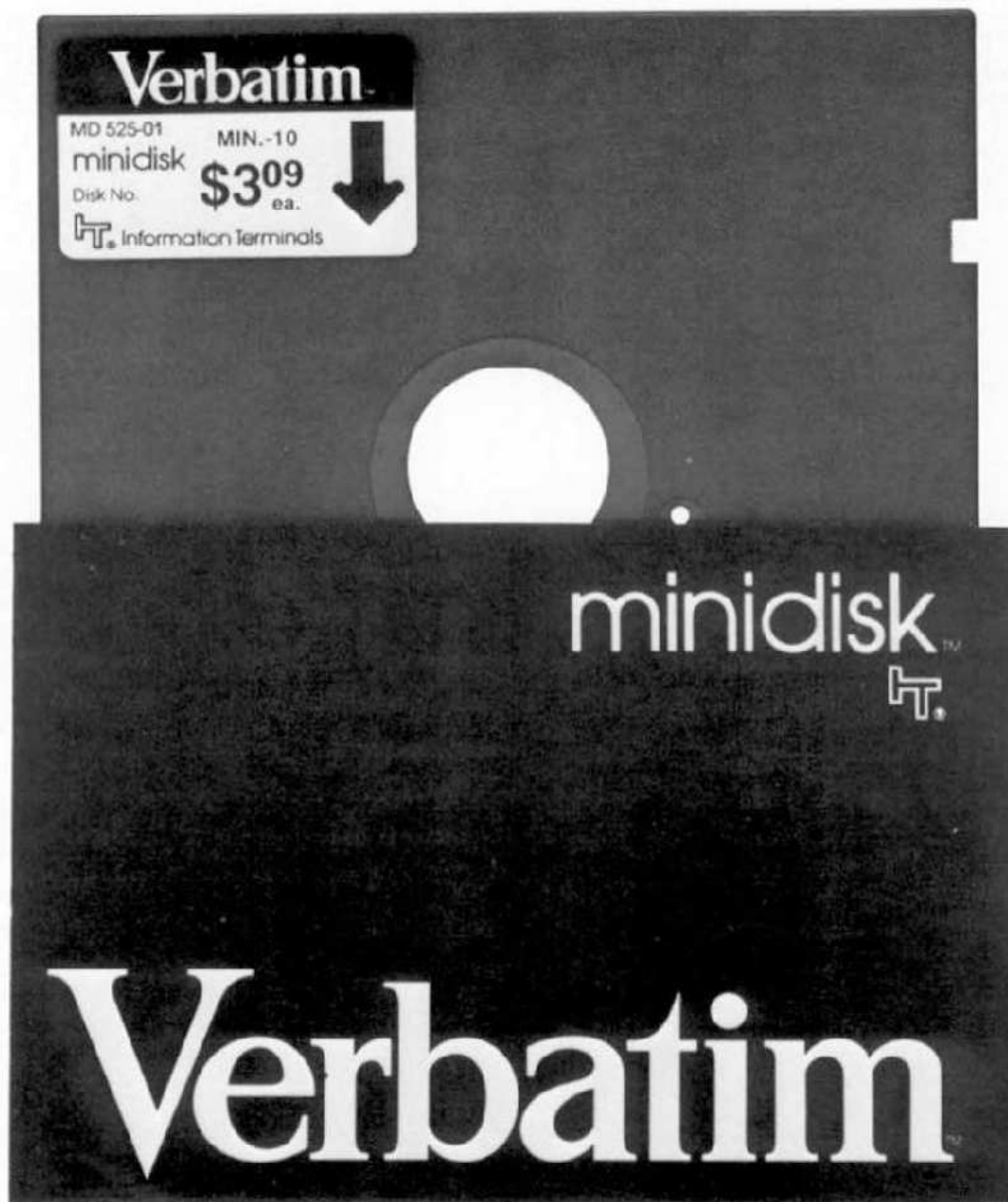
NULL Null or nothing '00' or '2F'
used to fill space or time.
STX Start-of-text '02'
used to signal that valid data will follow this character.
ETX End-of-text '03'
end of valid data for this block.
CR Carriage-return '0D'
LF Line-feed '0A'

The format of the tape is;
-EOT- signal 1 character name follows.
X 1 character user supplied name.
-STX- indicate start of real data line.
X 4 digit line number
X text portion of line
-CR- carriage-return/line terminator
-LF- line feed
-NULL- time delay - the number of nulls varies from version to version. this 'time delay' is used by the interpreter to 'digest' the line and convert it into it's internal format.
-STX- next line
.....
-CR-
-LF-
-ETX- indicates end of this BASIC program.

Some of the versions have a set of nulls at the beginning, and the end. The early versions of SWTPC used '2F' for the null, which meant you could 'see' them on the AC-30 LED's. The SWTPC versions use 4 nulls as the time delay, while the Computerware version uses only 2. Another interesting point is the LET verb. The SWTPC versions just output it if it is in the source.... LET A=1. But, if it is not there, a '08' is put on the tape to indicate the implied LET. Computerware BASIC always outputs a LET, even if it is not in the source.....
10 A=1 is written 10 LET A=1.

Reprint from '6800 BITS'
Chicago Area 6800 Newsletter
Phil Schuman, Editor

5-1/4" Minidisk — Soft or Hard Sector



S
A
V
E

D
I
S
K

SOUTH EAST MEDIA SUPPLY

6131 Airways Blvd.

615-892-1328

Chattanooga, TN 37421

H H H ENTERPRISES

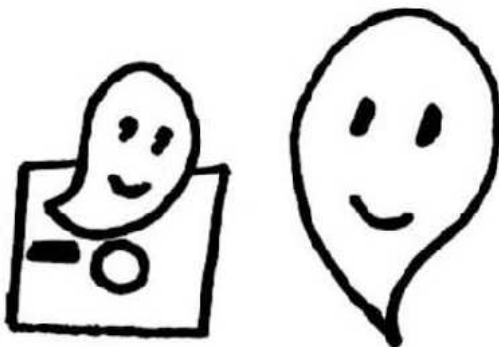
Box 493, Laurel, Md.
20810 301-953-1155

'SPIRIT' IS NOW AVAILABLE.

SPIRIT was called 'fourth' in our previous add, and has been renamed to reduce confusion. SPIRIT is a FOURTH (TM) like language with a built in dictionary of over 200 words. It is stack oriented and in normal use you would continue to define new 'words' that are immediately compiled to become part of the language. Smoke disk (5 or 8) only, at the moment, and includes a large manual. It is available by mail or phone order and dealer inquiries are invited.

It also lends itself very nicely to using those extra chapters of memory that are accessed by the GIMIX ghostable 16K boards.

'Get into the SPIRIT folks.' Priced at \$69.95, Visa & Mastercharge OK.



See GIMIX Ad. Page 3

Don't forget our Super Starter System: A Gimix box, 16K of ram, I/O card, Documentation, Keyboard, Monitor, GMXBUG 2.0 & 64X16 memory mapped video board for only \$1844.29. For dual Smoke disk add \$1000.

Prom burners special: 1 ea., 4KPPD (a 4K prom bd and a mass prom burner), 1 ea., 8KPBD (an 8K prom bd) & 10 ea., 2708s for only \$348.30.

Disk Special, subject to supply limitations, 5 inch, dual drive, DOUBLE SIDED, Smoke Signal Broadcasting, only \$1300.

'FOURTH' is a trademark of FOURTH INC.

JPC PRODUCTS FOR

6800

COMPUTERS

SWTPC and MSI

AD-16 DATA ACQUISITION - 69.95

- 16 CHANNEL precision analog input board.
- VERSATILE - supports individual channel programmable gain.
- ACCURATE - 0.39% resolution with less than 0.70% total conversion error.
- FAST - more than 3300 samples per second.
- SOFTWARE - included with documentation.

TC-3 CASSETTE INTERFACE - 49.95

- FAST - 4800 Baud Loads 4K in 8 Seconds!
- RELIABLE - Error Rate Less Than 1 in 10⁶ BYTES.
- CONVENIENT - Plugs Directly Into The Motherboard.
- PLUS - Read and Write Kansas City Standard Format at 300 Baud.

CFM/3 SOFTWARE - 19.95

- CASSETTE OPERATING SYSTEM for the TC-3 cassette interface. 2K memory required.
- FILE MANAGER supports named files, load, save, run, find, list, move, dir., etc.
- BASIC, ASSEMBLER and EDITOR can now support named files through the file manager.
- OPTIONAL CFM/3 on cassette - 6.95 additional.

etc.

- MX-6 - main board extender - \$19.95
- BT-9 - bus terminator - eliminate bus noise - \$9.95

"Designed By Professionals For Outstanding Performance"



P.O. BOX 5615
ALBUQUERQUE, N.M. 87185

15051 294-1623
ADD \$2.00 PER ORDER
FOR SHIPPING & HANDLING

ADVANCED 6800 SOFTWARE

CP/68™ OPERATING SYSTEM

(\$149.95)

- PIP Peripheral Interchange Program Transfers Data Between Physical Devices
- Wildcard Filenames and Extensions
- Relocatable Anywhere in Memory
- Extended Instruction Set Includes 6809-type Instructions (PSHX, PULX, etc)
- Complete Device-Independent I/O
- Random and Sequential Files
- Fits in Less Than 8k
- Chaining and Overlaying
- Single Supervisor Call Furnishes All DOS Services
- Easily Interfaced to New Devices and Peripherals
- Dynamic File Allocation

STRUctured BASic Language (STRUBAL +™) COMPILER

for both business and science uses

(\$249.95)

- Variable Precision From 4 to 14 Digits
- Structured Programming Forms
- Produces Relocatable and Linkable Code
- COMMON and DUMMY Sections
- Extensibility
- String Handling
- Full Scientific Package
- Data Structures with Mixed Data Types

LNKEDT68™ LINKAGE EDITOR

(\$49.95)

- Link and Relocate Files Produced by STRUBAL + or RA6800ML Macro Assembler
- Two-Pass Disk-to-Disk
- Libraries of Relocatable Files Can be Searched
- Extensive Listing Outputs

XREF68™

(\$29.95)

- Produces a Cross-Reference Listing of the XREF File Produced by STRUBAL + or RA6800ML Macro Assembler
- Alphabetic Listing of Every Symbol in a Source Program Followed by the Line Number of Every Reference

RA6800MLD™ MACRO LINKING ASSEMBLER

(\$79.95)

- Full Macro Facilities
- COMMON Section for the Production of ROMable Code
- Conditional Assembly
- Generates Linkable and Relocatable Code
- Sorted Symbol Table Listing
- Hash-coded Symbol Table for Speed

EDIT68™ TEXT EDITOR

(\$39.95)

- Change, Delete, Replace Text Globally
- Powerful Macro Facilities
- Supports Horizontal Tabs
- Disk-to-disk Edit Accommodates File of any Length
- Find a Character or Character String
- Block Moves of a Line or Lines

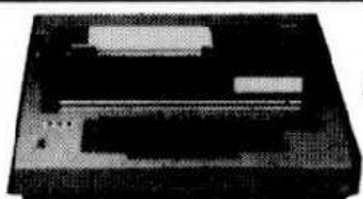
Versions are available for Percom, ICOM, Smoke Signal and SWTPC systems.

HEMENWAY ASSOCIATES, INC.

101 Tremont St. Suite 208
Boston, MA 02108
(617) 426-1931

SURPLUS ELECTRONICS

ASCII



ASCII

**WITH FLEX DRIVERS®
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, NH 03060

Phone orders accepted using VISA

or MC. Toll Free 1-800-258-1036

In N.H. 603-889-7661

DISK MIXER

**SMOKE SIGNAL DISK USERS
RUN MIXED 8 AND
5 INCH DRIVES**

The DM-85 Disk Mixer is an add-on board for the Smoke Signal Broadcasting BFD-68A Disk Controller which allows operation of both 8" and 5" drives. Controller mode (8" or 5") is selected on a drive-by-drive basis, so any mix of 5" and 8" drives is allowable. The 2" x 3" PC board mounts inconspicuously on the back of the BFD-86A. Its operation is completely transparent to software. An oscilloscope is required for the setup procedure. Kit Price: \$39.95.

THE **MICRO
WORKS**

Master Charge

and

BankAmericard

P.O. BOX 1110, DEL MAR, CA 92014 (714) 756-2687



**FINALLY, A
TELEPHONE
WITH BYTE!**

6800 AUTOMATIC TELEPHONE DIALER PROGRAM \$9.95 postpaid

Have your 6800 system dial your phone • Uses only 5 external components • Stores 650 variable length phone numbers • Operates in less than 1K bytes of memory

Includes: Paper tape in Mikbug® format and object code • Circuit diagram and instructions • Instructions for adapting to other 6800 systems

6800 TELEPHONE ANSWERING DEVICE PROGRAM \$4.95 postpaid

Have your 6800 system answer your phone and record messages automatically. Compatible with any 6800 system.

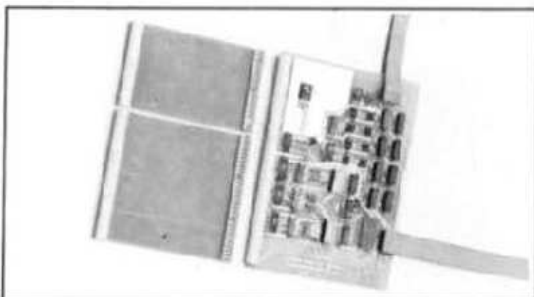
Includes: Assembly listing and object code • Circuit diagram and instructions

Write to: **SOFTWARE EXCHANGE**

2681 PETERBORO

W. BLOOMFIELD, MICH. 48033

Mikbug® is a registered trademark of Motorola Inc.



6800 OWNERS

At last a real world **fully addressable** SS-50 control interface. Control robots, appliances, organs, solar devices, etc. Applications limited only by your imagination. Easy to use with machine language as well as basic. Fully buffered board plugs directly onto mother board and responds to any address defined by user. 8 fast relays latch data while 8 opto-isolators allow handshaking capacity.

Kit \$98.00

Assembled and tested \$125.00

EXTENDER BOARDS

Extend both the 30 and 50 pin buses in SWTP 6800.

Both for \$19.95.

Visa & Master Charge • Anz. Res. add 5% Sales Tax

WRITE FOR DETAILS

TRANSITION ENTERPRISES INC.

Star Route, Box 241, Buckeye, AZ 85326

SUPER SOFTWARE!

MICROWARE 6800 SOFTWARE IS INNOVATION AND PERFORMANCE

NEW LISP Interpreter

The programming language LISP offers exciting new possibilities for microcomputer applications. A highly interactive interpreter that uses list-type data structures which are simultaneously data and executable instructions. LISP features an unusual structured, recursive function-oriented syntax. Widely used for processing, artificial intelligence, education, simulation and computer-aided design. 6800 LISP requires a minimum of 12K RAM.

Price: \$75.00

A/BASIC Compiler

The ever-growing A/BASIC family is threatening old-fashioned assembly language programming in a big way. This BASIC compiler generates pure, fast, efficient 6800 machine language from easy to write BASIC source programs. Uses ultra-fast integer math, extended string functions, boolean operators and real-time operations. Output is ROMable and RUNS WITHOUT ANY RUN-TIME PACKAGE. Disk versions have disk I/O statements and require 12K memory and host DOS. Cassette version runs in 8K and requires RT/68 operating system.

Price: Disk Extended Version 2.1 \$150.00

Cassette Version 1.0 \$65.00

NEW A/BASIC Source Generator

An "add-on" option for A/BASIC Compiler disk versions that adds an extra third pass which generates a full assembly-language output listing AND assembly language source file. Uses original BASIC names and inserts BASIC source lines as comments. SSB and SWTPC Miniflex version available.

Price: \$60.00

NEW A/BASIC Interpreter

Here it is—a super-fast A/BASIC interpreter that is source-compatible with our A/BASIC compiler! Now you can interactively edit, execute and debug A/BASIC programs with the ease of an interpreter—then compile to super efficient machine language. Also a superb stand-alone applications and control oriented interpreter. Requires 8K RAM. The cassette version is perfect for Motorola D2 Kits.

Price: \$75.00

RT/68 Real Time Operating System

MIKBUG—compatible ROM that combines an improved monitor/debugger with a powerful multitasking real-time operating system. Supports up to 16 concurrent tasks at 6 priority levels plus real time clock and interrupt control. Thousands in use since 1976 handling all types of applications. Available on 8830 (MIKBUG-1 pe) or 2706 (EPROM-type) ROM. Manual is a classic on 6800 real-time applications and contains a full source program listing.

Price: RT68MX (8830) \$55.00

RT68MXP (2706) \$65.00

6800CHESS

A challenging chess program for the 6800. Two selectable difficulty levels. Displays formatted chess board on standard terminals. Requires 8K memory. Machine language with A/BASIC source listing.

Price: \$50.00

ELIZA

6800 version of the famous MIT artificial intelligence program. The computer assumes the role of a psychoanalyst and you are the patient. This unusual program is unique because the dialog with the computer is in unstructured plain English. An impressive demonstration program.

Price: \$30.00

Our software is available for most popular 6800 systems on cassette or diskette unless otherwise noted. Disk versions available on SSB, SWTPC, or Motorola DOS. Please specify which you require. Phone orders are welcomed. We accept ASTERCHARGE and VISA. We try to ship orders within 24 hours of receipt. Please call or write if you require additional information or our free catalog. Microware software is available for OEM and custom applications.

MICROWARE
SYSTEMS CORPORATION

P.O. BOX 4885
DES MOINES, IA 50304
(515) 281-6121

'68' MICRO JOURNAL

- ★ The only ALL 6800 Computer Magazine.
- ★ More 6800 material than all the others combined:

MAGAZINE COMPARISON

(2 years)

Monthly Averages

6800 Articles

KB	BYTE	CC	DOBB'S	TOTAL PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: \$5.88
(Based on advertised 1-year subscription price)

'68' cost per month: 88¢

(\$10.50 Charter Subscription Rate)

That's Right! Much, Much More
for

1/6 the Cost!

CHARTER SUBSCRIPTION SPECIAL

1-Year \$10.50 2 Years \$18.50 3 Years \$26.50

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge ☐ — VISA ☐

Card # _____ Exp. Date _____

For ☐ 1-Year ☐ 2 Years ☐ 3 Years

Enclosed: \$ _____

Name _____

Street _____

City _____ State _____ Zip _____

My Computer Is: _____

68 MICRO JOURNAL

3018 Hamill Road

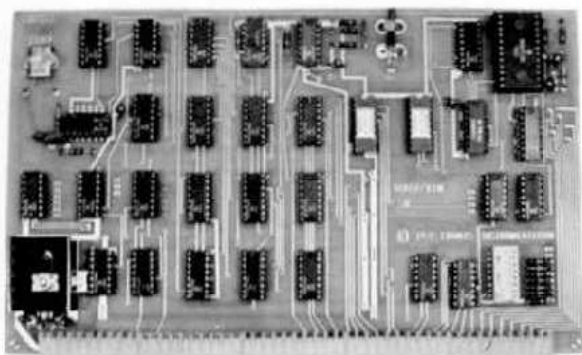
HIXSON, TN 37343

Foreign surface add \$9.50 per year.

Foreign Air Mail add \$29.00 per year.



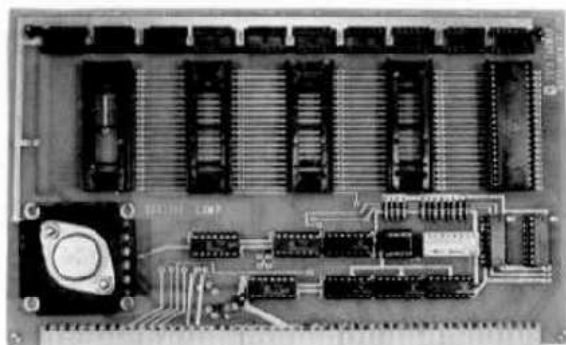
GOOD DEAL EXTENDED!
PRICES SHOWN ARE FOR ORDERS RECEIVED BY JUNE 30
 NORMAL PRICES APPROX. 10% HIGHER . . . CALL FOR DEALER, OEM, AND GROUP PRICING



SS-50 VIDEO RAM

- Fully Synchronous Operation to Eliminate Jitter
- Full 128 Set of 7 by 9 Characters & Reverse Image
- 1K of Memory can be Mapped in any 1K Increment
- 10 Pages of Documentation Including Software

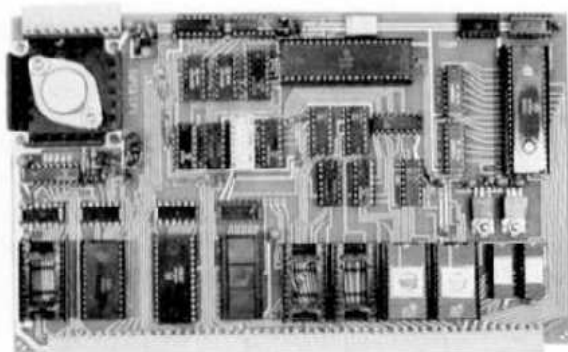
ASSEMBLED \$135.00
CARD, CRYSTAL, & DOC \$35.00



SS-50 PARALLEL I/O

- 2 Ports Expandable to 10 (Just Add PIA'S @ \$6.95 ea.)
- Each Port has 8 I/O Plus 2 Control . . . 100 I/O Lines . . .
- Large Regulator Supplies Power to I/O Connectors
- Board Can be Mapped to any 32 Word Boundary

ASSEMBLED (W/PIA) \$89.00
CARD ONLY \$32.00



SS-50 SUPER CPU

- SS-50 Buss . . . or . . . Stand Alone Computer
- 1K of Ram at \$A000, I/O on Board at \$A400 (Relocatable)
- 2K Monitor in 2708 Eprom (Expandable to 4K)
- 2 Parallel 8 bit ports with 2 Control Bits and Power
- 1 RS-232 ACIA Port (Expandable to 2nd TTL ACIA)
- 3-16 Bit Counter Timers (Expandable to 6 . . . Add 2nd 6840)
- Additional 128 Byte Ram at 0000 can be Jumped out
- Battery Back-up Available on 32 Bytes of Ram
- Back/Back with other SS-50 Cards to Make Small Systems

ASSEMBLED (AS SHOWN) \$179.00
CARD ONLY \$44.00

Gimix 16K Without Soft Addressing . . . \$298.13
 Gimix 16K Static Ram W/Softadd . . . \$368.16
 Our SS-50 Wire-Rap Card . . . \$24.00

3, 4, and 7 slot SS-50 Backplanes
We also Make Non SS-50 Single Board 6800 Sys.

We have been in business for over 8 years, designing industrial machine tool controls and monitors . . . Call us for help with your product or laboratory instrumentation problem . . . we are dealers for:

GIMIX SWTPC SSB
 . . . LET US QUOTE YOU A PACKAGE PRICE . . .

THOMAS INSTRUMENTATION

2709 Dune Drive, Avalon, N.J. 08202
 Phone (609) 967-4280



Advanced Disk Operating and File Management System from PERCOM

INDEX™
(Interrupt Driven EXecutive)
only \$99.95



A fast, adaptable disk operating program for 6800 computers featuring:

Fast execution — I/O devices are serviced by interrupt request. Eliminates polling time.

Adaptability — I/O devices and peripherals are accessed the same as disk files. New devices may be added without changing the operating system.

Versatility — An unlimited number of DOS commands may be added. Over 60 system entry points for program linkage provided.

System savings — INDEX™ and user-added commands and routines are disk-resident. No need to add memory for program enhancement.

Optional 108-page Advanced Programmer's Guide: I/O drivers & examples, describes system entry points, etc.... \$45.00

Versions of INDEX™ are available for the PERCOM LFD-400™, SWTP MF-68 and Smoke Signal Broadcasting Company's BFD-68 disk systems, and for Motorola's EXORciser™ development system.

INDEX™ handles both ASCII and binary files, and disk files are automatically created, allocated and deallocated.

Files are referenced by names — of up to eight characters — and file name parameters are added for name extension (to further define the file), drive number, directory level and a file protection flag.

The INDEX™ operating system software also features a versatile BACKUP routine for copying files onto a diskette.

Console Interface Software

The console interface segment of INDEX™ software supports any stan-

dard serial ASCII terminal, and features:

- Program Interrupt (vs. Reset) for runaway programs.
- Operator Start, Stop and Skip display control.
- An interrupt-serviced, type-ahead character-queue buffer.
- A secondary line editing queue buffer.

INDEX™ Commands

Commands are given to INDEX™ a line at a time. Processing begins when the Return key is struck. The type-ahead character-queue buffer allows the operator to input a command while the previous command is being executed. A character may be deleted and corrected with the Backspace (Control H) key, and a line may be cancelled

with the Rub-Out key. In addition to the following commands, the user may add any number of his own commands.

BACKUP • CONVERT • COPY • COUNT • DATE • DELETE • DIRECTORY • DISKEDIT • DISKINIT • EXAMINE • FILL • HELP • RENAME • SAVE • SETSTART • SETVERSION • SYSTEMDISK • TYPE • USERDISK

System Requirements

System requirements are as follows:

1. 8K RAM at address \$A000 — \$BFFF
2. Minimum of 8K RAM beginning at address \$0000
3. ACIA console interface (SWTP MP-C interface)
4. SWTBUG™ or equivalent monitor

INDEX is a trademark of PERCOM Data Company. LFD-400 is a trademark of PERCOM Data Company.

EXORciser is a trademark of the Motorola Corporation. SWTBUG is a trademark of Southwest Technical Products.

PERCOM

PERCOM DATA COMPANY, INC.
Dept. 68 211 N. Kirby Garland, TX 75042
(214) 272-3421

How to Get Your INDEX™ Software

INDEX™ is supplied on two mini-diskettes together with a Users Manual for \$99.95, and may be ordered from PERCOM by dialing, toll-free, 1-800-527-1592. In addition to checks and money orders, Visa and Master Charge credit cards are honored. Texas residents add 5% for sales tax.

PerCom 'peripherals for personal computing'

We're not just blowing smoke

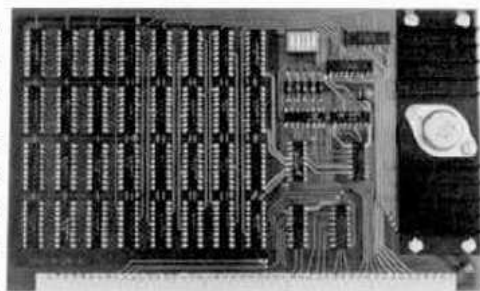
SMOKE SIGNAL BROADCASTING PRESENTS IT'S

\$299.00

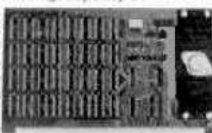
M-16A STATIC MEMORY SYSTEM

- Allows SWTPC 6800 expansion to 48K
- Low Power • Uses Single +8 Volt Supply
- SWTPC 6800 Plug Compatible • STATIC – No refresh required

The M-16A STATIC random access memory system, with a total storage capacity of 16834 words of 8 bits each, is switch selectable to any 4K starting address, and a hardware write protect switch is also included. The system's storage elements are 4K by 1 STATIC memory chips which store 4 times as much in only 12% more space than the low power 2102's. Typical access time is fast enough to work a 6800 based computer operating at 2 MHz and all systems are factory tested at 2 MHz.



The M-16A STATIC random access memory system, with a total storage capacity of 16834 words of 8 bits each, is switch selectable to any 4K starting address, and a hardware write protect switch is also included. The system's storage elements are 4K by 1 STATIC memory chips which store 4 times as much in only 12% more space than the low power 2102's. Typical access time is fast enough to work a 6800 based computer operating at 2 MHz and all systems are factory tested at 2 MHz.



The M-16A STATIC random access memory system, with a total storage capacity of 16834 words of 8 bits each, is switch selectable to any 4K starting address, and a hardware write protect switch is also included. The system's storage elements are 4K by 1 STATIC memory chips which store 4 times as much in only 12% more space than the low power 2102's. Typical access time is fast enough to work a 6800 based computer operating at 2 MHz and all systems are factory tested at 2 MHz.



SMOKE SIGNAL



BROADCASTING®

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

SMOKE SIGNAL BROADCASTING®

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

- ☐ Send information on your M-16A
☐ Send name of nearest dealer

Name _____
Address _____
Company _____
City _____
State/Zip _____

000422 A/E

MR. MICKEY FERGUSON
BOX 708
TRENTON

GA 30752

For your SWTP 6800 Computer...

PERCOM's™ FLOPPY DISK SYSTEM the LFD-400

Ready to plug in and run the moment you receive it. Nothing else to buy, no extra memory. No "booting" with PerCom MINIDOS-PLUSX™, the remarkable disk operating system on EPROM. Expandable to either two or three drives. Outstanding operating, utility and application programs.



To order
or request
literature
call Percom
toll-free
1-800-527-1592.

only
\$599⁹⁵

fully assembled and tested
shipping paid

PERCOM

PERCOM DATA COMPANY, INC.

211 N. Kirby Garland, TX 75042
(214) 272-3421

For the low \$599.95 price, you not only get the disk drive, drive power supply, SS-50 bus controller/interface card, and MINIDOS-PLUSX™, you also receive:

- an attractive metal enclosure • a fully assembled and tested interconnecting cable • a 70-page instruction manual that includes operating instructions, schematics, service procedures and a complete listing of MINIDOS™ • technical memo updates — helpful hints which supplement the manual instructions • a 90-day limited warranty.

SOFTWARE FOR THE LFD-400 SYSTEM

Disk operating and file management system

INDEX™ The most advanced disk operating and file management system available for the 6800. Interrupt Driven EXecutive operating system features file-and-device-independent, queue-buffered character stream I/O. Linked-file disk architecture, with automatic file creation and allocation for ASCII and binary files, supports sequential and semi-random access disk files. Multi-level file name directory includes name, extension, version, protection and date. Requires 8K RAM at \$4000. Diskette includes numerous utilities \$99.95

BASIC Interpreters and Compilers

SUPER BASIC A 10K extended disk BASIC Interpreter for the 6800. Faster than SWTP BASIC. Handles data files. Programs may be prepared using a text editor described below \$49.95
BASIC BANDAID™ Turn SWTP 8K BASIC into a random access data file disk BASIC. Includes many speed improvements, and program disk CHAINing \$17.95
STRUBAL+™ A STRUCTURED BASIC Language compiler for the professional programmer. 14-digit floating point, strings, scientific functions, 2-dimensional arrays. Requires 20K RAM and Linkage Editor (see below). Use of the following text editors to prepare programs. Complete with RUN-TIME and FLOATING POINT packages \$249.95

Text Editors and Processors

EDIT88 Hemenway Associates' powerful disk-based text editor. May be used to create programs and data files. Supports MACROS which perform complex, repetitive editing functions. Permits text files larger than available RAM to be created and edited \$39.95
TOUCHUP™ Modifies TSC's Text Editor and Text Processor for PerCom disk operation. ROLL function permits text files larger than available RAM to be created and edited. Supplied on diskette complete with source listing \$17.95

Assemblers

PerCom 6800 SYMBOLIC ASSEMBLER Specify assembly options at time of assembly with this symbolic assembler. Source listing on diskette \$29.95
MACRO-RELOCATING ASSEMBLER Hemenway Associates' assembler for the programming professional. Generates relocatable linking object code. Supports MACROS. Permits conditional assembly \$79.95
LINKAGE EDITOR — for STRUBAL+™ and the MACRO-Relocating assembler \$49.95
CROSS REFERENCE Utility program that produces a cross-reference listing of an input source listing file \$29.95

Business Applications

GENERAL LEDGER SYSTEM Accommodates up to 250 accounts. Financial information immediately available — no sorting required. Audit trail information permits tracking from GL record data back to source document. User defines account numbers \$199.95
FULL FUNCTION MAILING LIST 700 addresses per diskette. Powerful search, sort, create and update capability \$99.95
PERCOM FINDER™ General purpose information retrieval system and data base manager \$99.95

™ trademark of PERCOM Data Company, Inc.
™ trademark of Hemenway Associates Company

Now! The LFD-800 and LFD-1000. Add one, two or three LFD-800 drives and store 200K bytes per drive on-line. Add one or two (dual-drive) LFD-1000 units and store 800K bytes per unit on-line. Complete with interface/controller, DOS, cable & manuals. Two-drive systems: LFD-800 — \$1549; LFD-1000 — \$2495.

PERCOM 'peripherals for personal computing'