Popular Electronics®

WORLD'S LARGEST SELLING ELECTRONICS MAGAZINE AUGUST 1982/$1.25

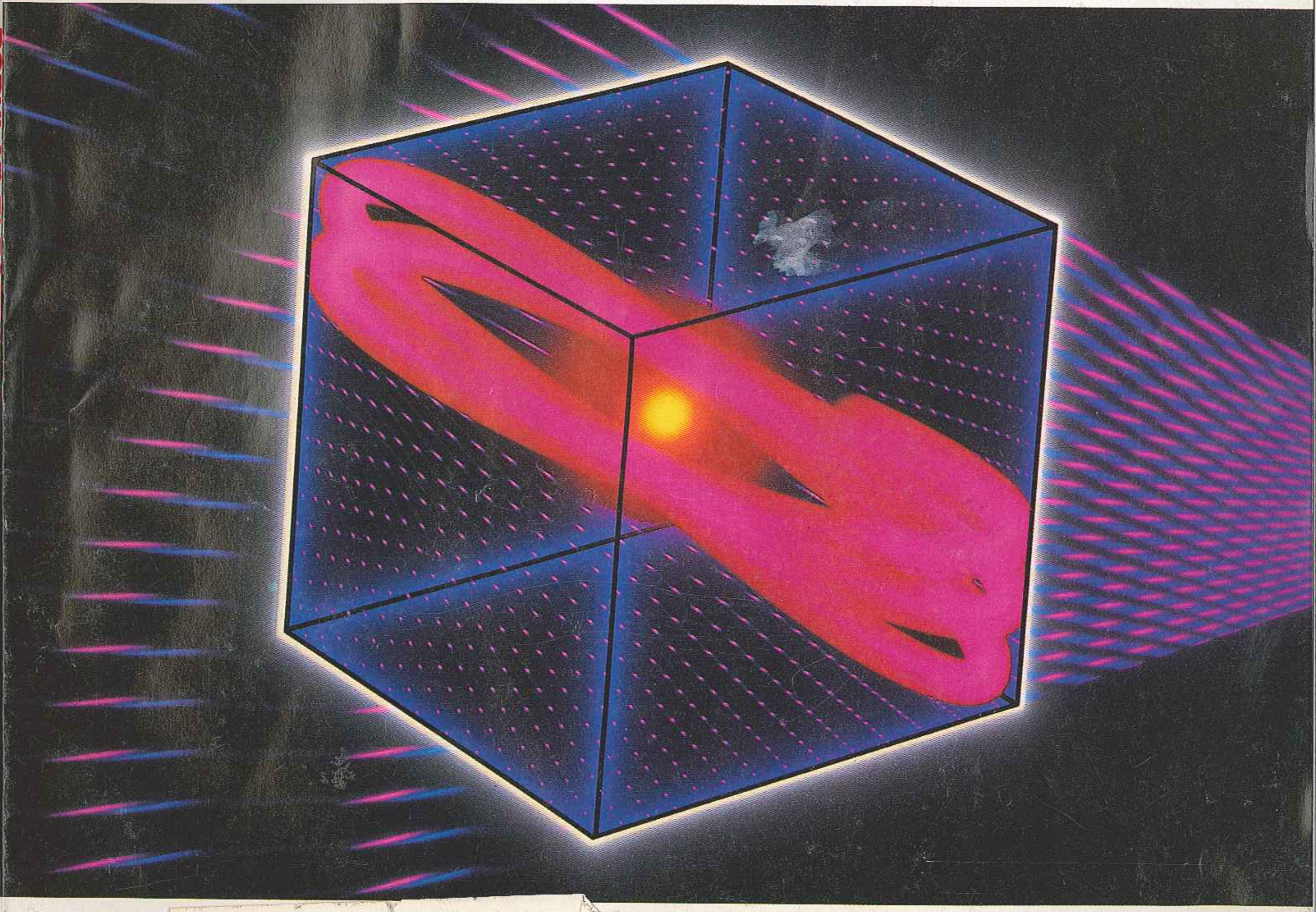
Loran-C for Boat Navigating

Math Software for Elf Computers

The State of Stereo TV Sound

SPECIAL FOCUS ON

Home Energy-Saving Applications



ssue:

4 Microcomputer

on PM650 Stereo Amplifier

GE 19PC3708W 19" Color TV Receiver

0 14024 1427

A 16-BIT MATH PACKAGE FOR ELF COMPUTERS

Software provides all basic mathematic functions and operations on a minimum 1802-cpu configuration

BY R.S. FITZGERALD

WHILE there are many cassette-based programs for the Elf microcomputer based on the 1802 cpu, there are few that run on its minimum configuration of 256 bytes. This article describes a 16-bit mathematics package that will indeed operate in a minimum system. The package includes operators and functions for addition, subtraction, multiplication, division, negation, and absolute values. All of these are implemented in a package using only 134 bytes.

Since this package, which we call "MATH 16," depends upon the use of the stack operations, it would be best to review them before discussing operation of the package. The stack is a variable-sized data-storage area that can be located in any convenient memory area. It is addressed by a pointer called the stack pointer (SP). The stack grows in size as the SP moves upward (lower memory addresses), much like a stack of dishes in a spring-loaded dish compartment. Also like a stack of dishes, it shrinks as the bytes are removed. Stack operations are defined in "Stack Operations—A Review," accompanying this article.

The object code representation of "MATH 16" appears in Table I. The program is assembled starting at location 70 hex. This allows room for 5 operands on the stack in a 256-byte Elf (probably more than you'll ever need), and ample space for some type of monitor in low memory. Of course, if your Elf (or any other 1802-based microcomputer has more than 256 bytes of memory, "MATH 16" can be moved around to suit your needs. If you intend to do this, remember to modify the addresses in your calling routines and the jump addresses within "MATH 16."

The stack can exist any place in memory that is convenient, the only restriction being that the stack pointer (RX) must be R_2. All of the subroutines leave RX pointing at the next available byte on the stack. Thus, the value in the stack pointer will be one less than the address of the most-significant byte of the number on the top-of-stack (TOS). Each of the functions is terminated with a SEP R_3 instruction to return to the calling routine. If your driver routine uses some other register than R_3 as the program counter, these SEP instructions must be changed to reflect this.

Table II lists the entry point addresses for the various "MATH 16" subroutines. The functions are executed by performing a subroutine call to the address corresponding to the function desired. For example, the recommended calling procedure for the multiply function is shown in Table III.

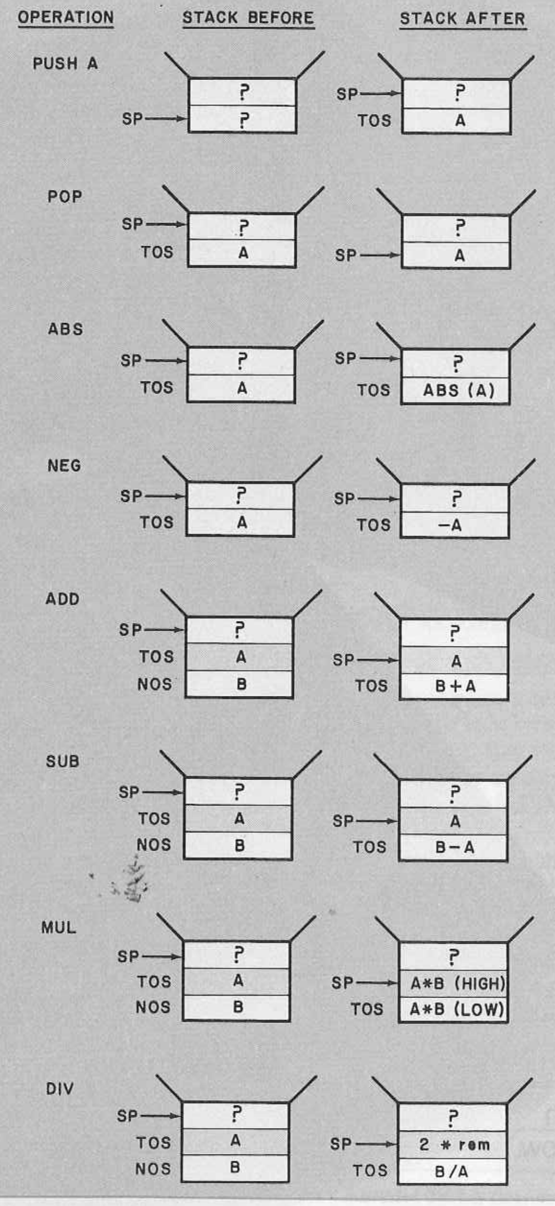
TABLE I—MATH 16 MACHINE-LANGUAGE PROGRAM

70	60	72	BF	72	60	F5	73	9F	75	73	D3	60	72	BF	72	60
80	F4	73	9F	74	73	D3	60	F0	73	FE	CF	D3	C4	60	60	F8
90	00	F7	73	F8	00	77	73	D3	60	72	BF	F0	AF	F7	FE	73
A0	52	F8	11	AE	F8	04	AD	F0	76	52	60	2D	8D	3A	A7	22
B0	22	22	2E	8E	C6	D3	C4	22	3B	A4	12	8F	F4	73	9F	74
C0	52	30	A4	60	72	BF	F0	AF	F7	FE	73	52	F8	11	AE	60
D0	60	60	F8	04	AD	F0	7E	73	2D	8D	3A	D5	60	60	2E	8E
E0	C6	D3	C4	8F	F5	73	9F	75	52	33	CF	60	8F	F4	73	9F
F0	74	52	FC	00	30	CF	00	00	00	00	00	00	00	00	00	00

STACK OPERATION—A REVIEW

Shown here are some pictorial representations of various stack operations such as PUSH, POP, and all of the operations performed by "MATH 16." Each box on the stacks in the drawing represents one data item or two bytes. Remember that the stack pointer is left pointing at the next available byte on the stack. For reference purposes, here are the terms involved.

Term	Meaning
LIFO Stack	The computer analogy to a pile of dishes. The last item (plate) to be placed on the stack (pile) will be the first one removed, hence the name Last In First Out Stack. Most often, the LIFO is omitted.
TOS	An acronym for Top Of Stack. This name refers to the data item that has been most recently placed on the stack.
NOS	An acronym for Next On Stack. This refers to the second data item on the stack.
Stack Pointer	The register that is used as a pointer to reference data items on the stack. Denoted RX in the 1802.
PUSH	To PUSH an item on the stack means to add another item to the stack and adjust the stack pointer so that it is pointing at the next available byte. A PUSH causes the stack to "grow" toward address 0000. STXD (73) is used in the 1802 to PUSH data.
POP	The opposite of PUSH. The IRX (60), LDXA (72) instruction sequence is used to implement this function.



At this point, it will be instructive to refer again to "Stack Operations—A Review." The illustrations show what happens when the various arithmetic operations are performed. As you can see, some of the functions operate on only one number (ABS, and NEG), while others operate on two numbers. Both types will be discussed.

The one-operand functions take their operand from TOS, perform the selected operation on that number, and return the result to TOS. With these functions, the stack neither grows nor shrinks.

The function NEG replaces TOS with the negative of itself. As an example, 0001 becomes FFFF (−1 in two's complement notation). ABS will return the absolute value of TOS. If the TOS contains FFFE (−2) before the call to ABS is made, the value after ABS has been called will be 0002.

The two-operand functions take both of their operands from the stack and return the result to the old NOS. The stack pointer is then adjusted so that the old next on-stack (NOS) becomes the new TOS (that is, the stack shrinks by 2 bytes). Care must be taken if some intermediate result on the stack needs to be saved for a later operation since the stack pointer is modified and NOS is overwritten by the new TOS. A good place to save these intermediate results is in RC because it is not modified by any of the "MATH 16" operations. Actually, the only registers that "MATH 16" deals with are: R_2, R_3, R_4, RD, RE, and RF.

As the name implies, ADD will sum TOS and NOS. As with all of the two-operand functions, the stack pointer will end up pointing two bytes higher than when it started out (SP = SP + 2).

The function SUB will subtract TOS from NOS. If you are really short on memory space, the SUB routine can be deleted. This same function can be achieved by a call to NEG, then to ADD. This saves 11 bytes.

The subroutine MUL computes the product of the *unsigned* numbers at TOS and NOS. The fact that unsigned numbers are used means that a result of FFFF does not indicate a result of −1 as might be expected. Rather, it indicates a result of 65535.

As with the multiply subroutine, DIV operates on unsigned numbers. In this case, the operand at NOS is divided by the operand at TOS. Since all of the subroutines operate on integers, no fractions can be obtained with DIV. The division routine does not check for division by zero. This condition must be avoided for DIV to produce meaningful results.

(Continued on page 66)

Here's why we're Number One.

When it comes to logic probes, more people buy Global Specialties. Because no one can match us for value. Our four logically-priced probes—including our remarkable new 150 MHz ECL—deliver more speed, accuracy, flexibility and reliability than others costing considerably more! So why compromise? Discover for yourself why we're the number-one logical choice!

Standard LP-1, $50.00* with memory —captures pulses to 50 nsec, 10 MHz, guaranteed.
Economy LP-2, $32.00*, to 50 nsec, 1.5 MHz.
High-speed LP-3, $77.00* with memory, guaranteed to 10 nsec (6 nsec, typical), 50 MHz!
New ECL LP-4, $150.00*, the new industry standard—with memory, guaranteed to 4 nsec (2 nsec, typical), 150 MHz!

GLOBAL SPECIALTIES CORPORATION

70 Fulton Terr. New Haven, CT 06509 (203) 624-3103, TWX 710-465-1227
OTHER OFFICES: San Francisco (415) 648-0611, TWX 910-372-7992
Europe: Phone Saffron-Walden 0799-21682, TLX 817477
Canada: Len Finkler Ltd., Downsview, Ontario

Call toll-free for details
1-800-243-6077
During business hours

*Suggested U.S. resale. Prices, specifications subject to change without notice

CIRCLE NO. 20 ON FREE INFORMATION CARD

TABLE II—ARITHMETIC FUNCTION ENTRY POINTS

Name	Function	Call Address
SUB	Subtraction	70
ADD	Addition	7B
MUL	Multiplication	98
DIV	Division	C3
ABS	Absolute Value	86
NEG	Negation	8D

TABLE III—CALLING PROCEDURE

```
Address  Contents  Mnemonic     Comments
N        F898      LDI A(MUL)   .SET UP SUBROUTINE
N+2      A4        PLO R4       .PROGRAM COUNTER
N+3      D4        SEP R4       .CALL MULTIPLY FUNCTION
                      *
                      *
                      *
```

It is interesting to note that the multiply and divide subroutines have some nice features which have not yet been mentioned. The multiply routine actually performs a 32-bit multiplication, but the stack pointer is adjusted, upon exit, to point at the least-significant 16 bits of the product. If you desire a 32-bit product, simply decrement the stack pointer twice by including two DEC R_2 instructions in your driver routine after the call to MUL.

The divide subroutine, on the other hand, only operates on 16-bit numbers. However, if the stack pointer is decremented by 2 after the division operation, the number on the TOS is two times the remainder of the previous division. The true remainder can be obtained either by shifting this number right one bit or by pushing a word of 0002 on the stack and calling the divide subroutine again.

Applied Example. The program segment in Table IV shows how to multiply two numbers using "MATH 16." Note that the least-significant byte of a 16-bit number is stored at address x, while the most-significant byte is stored at address x−1. This convention is used throughout.

After this routine is executed, the new TOS will contain 2710, which is 10000 in decimal notation.

"MATH 16" does have some limitations, such as error checking and integer-only arithmetic. But it can still provide one with a powerful tool for the 1802 system. ◇

SOFTWARE AVAILABILITY

The following are available from the Softek Co., Box 4232, Santa Fe, NM 87501: A documented source listing of "MATH 16" plus several sample applications of the package for $2.50 (item MATH 16S), and a quick-reference guide for the 1802 cpu's instructions and respective opcodes for $1 (item 1802Q).

TABLE IV—MULTIPLICATION PROGRAM

```
Address  Contents  Mmemonic     Comments
N        F8E8      LDI E8       .PUSH THE
N+2      73        STXD         .DECIMAL NUMBER
N+3      F803      LDI 03       .1000 ONTO
N+5      73        STXD         .THE STACK
N+6      F80A      LDI 0A       .PUSH THE
N+8      73        STXD         .DECIMAL NUMBER
N+9      F800      LDI 00       .10 ONTO
N+11     73        STXD         .THE STACK
N+12     F898      LDI A(MUL)   .SET R4 EQUAL TO
N+14     A4        PLO R4       .MUL ENTRY POINT
N+15     D4        SEP R4       .CALL MULTIPLY
N+16     00        IDL          .HALT
```

PROGRAMMING EPROM's WITH A SMALL COMPUTER

BY JOHN DOOLITTLE AND SLOBODAN TKALCEVIC

Part 2

IN Part 1 of this article, we discussed how EPROMs can be put to good use in expanding your digital design work. We then presented a circuit that can be used as an interface with a small computer such as the TRS-80 to program EPROMs. We are ready to proceed now with constructing the circuit.

Construction. The EPROM Programmer is designed so that all components except for the ac power supply are mounted on a double-sided printed circuit board such as that shown in Fig. 7. The four lines from the transformer secondary (color-coded blue, yellow, red, and green) are soldered to the corresponding points on the printed circuit board marked B, Y, R, and G. Connection to the computer is done through a 40-pin card-edge connector that is compatible with the expansion bus of the TRS-80 CPU or Expansion Interface unit. All components are mounted on the bottom side of the board—except the zero insertion force EPROM socket, the personality module socket, and the indicator LEDs. These parts mount on top of the board. The printed circuit can be neatly mounted in a 7" by 6" by 3" aluminum enclosure using ¼-inch spacers and allowing the topside components to protrude through the top panel. Access to the board's edge connector is made through a notch in the rear panel.

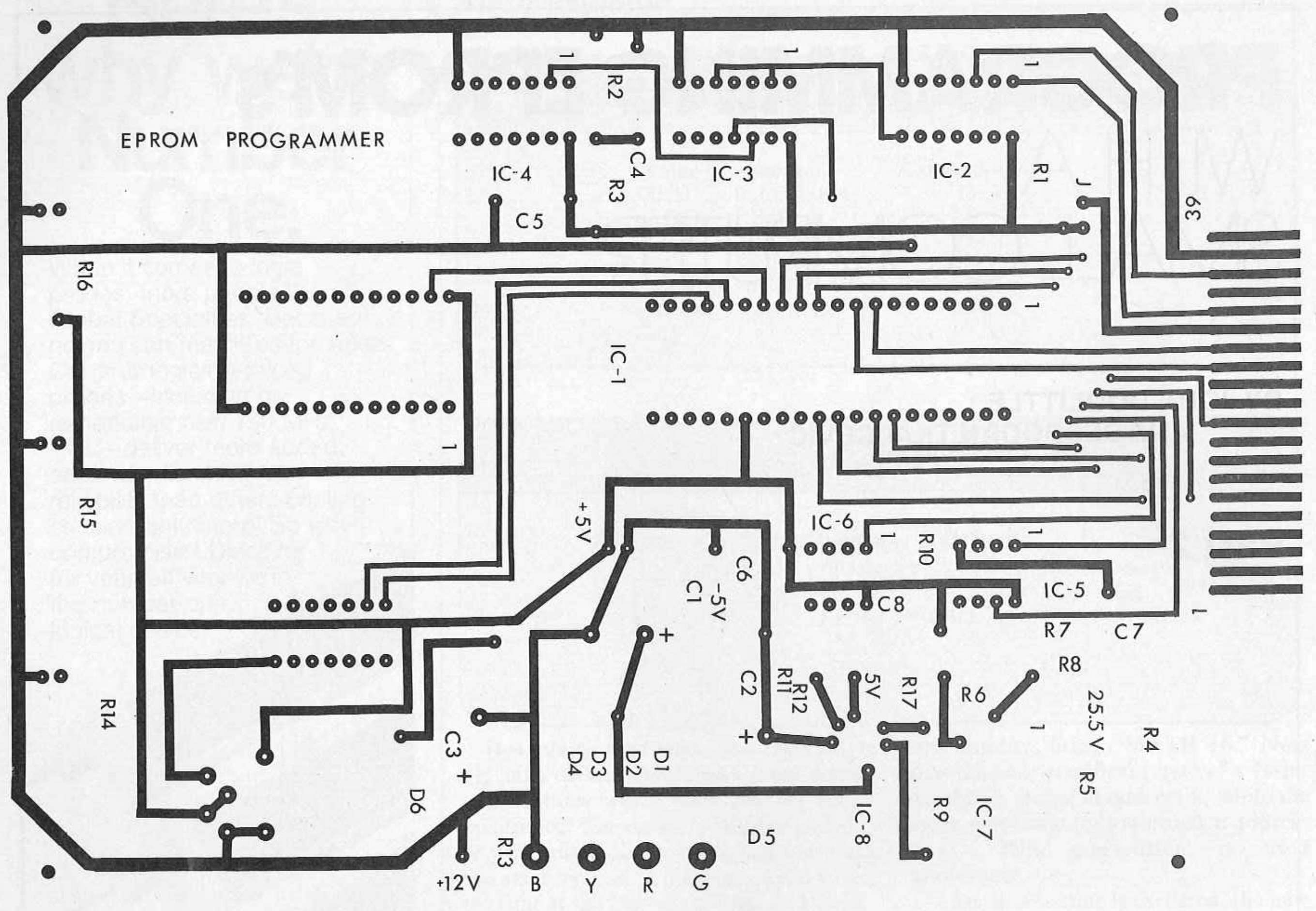
With all components soldered in place, switchable voltage regulators *IC7* and *IC8* should be adjusted (*R8* and *R12*) to develop 25.5 V and 5.0 V, respectively, before connecting the EPROM Programmer to the computer. The personality module socket is a convenient test point for the constant −5-V (pin 1) and 12-V (pin 4) supplies as well as for the switchable 5-V (pin 2) and 25.5-V (pin 3) supplies. The constant 5-V supply can be measured at pin 24 of the EPROM socket.

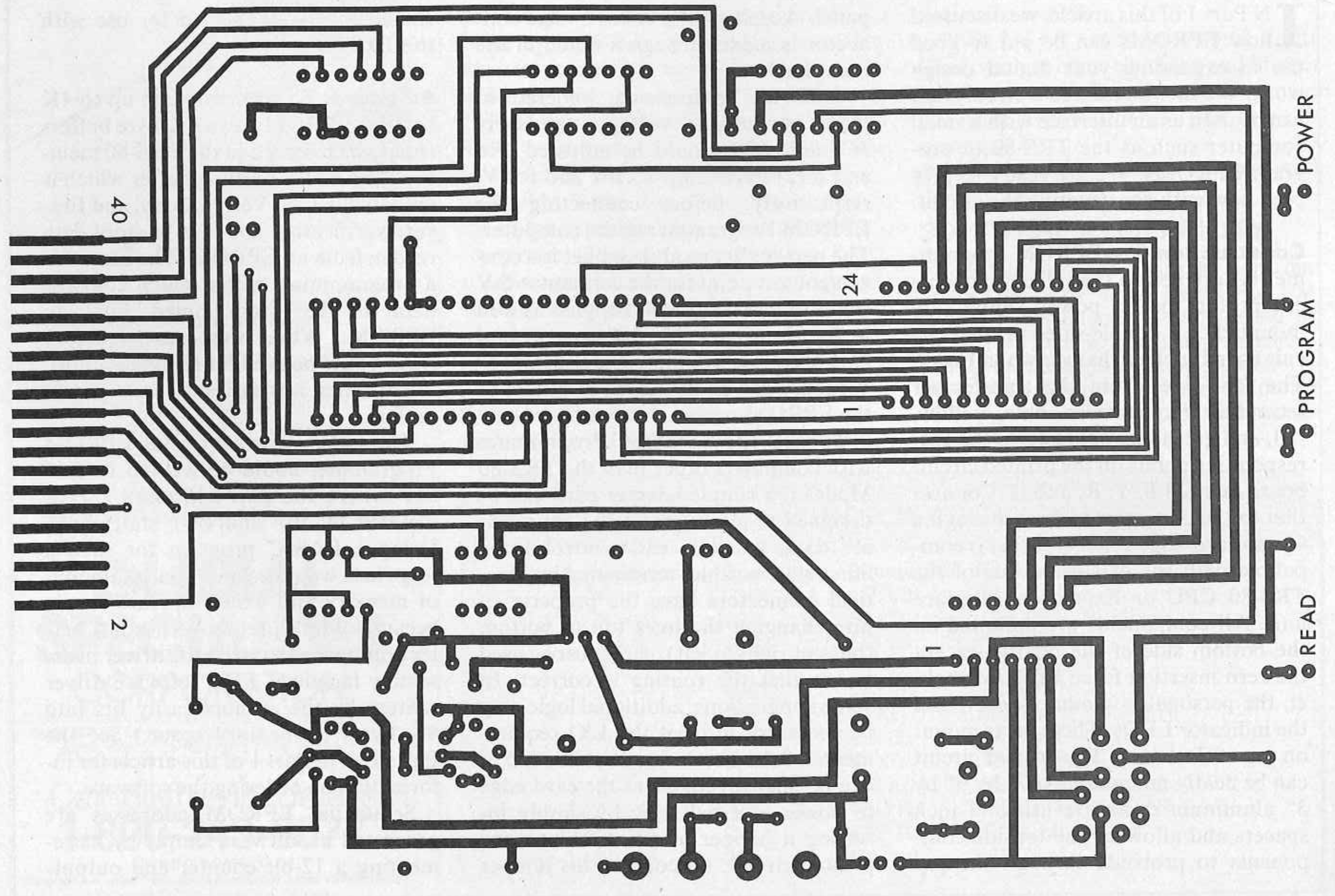
To use the EPROM Programmer with computers other than the TRS-80 Model I, a simple adapter card can be designed to interconnect the appropriate data, address, and control lines. Since ribbon cables terminated by identical connectors have the property of interchanging the lines top to bottom (but not right to left), care must be used to see that the routing is correct. In some applications, additional logic may be necessary to meet the I/O requirements of the host computer. If so, 5 V can be made available at the card edge to power external logic by simply inserting a jumper at point "J" on the printed circuit. Of course, this jumper should *not* be connected for use with the TRS-80.

Software. To accommodate up to 4K by 8-bit EPROMs, two 4K-byte buffers should be reserved in the TRS-80 memory. The first is a listing buffer which is used by the List, Verify, Copy, and Erasure verification routines to store data read in from an EPROM. The second is a programming buffer which contains data to be programmed into an EPROM. While the same memory space could be used for each task at different times, it is easiest to use separate buffers.

The software driver for the EPROM Programmer could be written in BASIC where the port-addressed I/O is handled by INP and OUT statements. Using a BASIC program for such a large task would require a large amount of memory and would operate slowly because of interpreter overhead. A better approach is to write the driver in assembly language. (The software driver written by the authors easily fits into 4K bytes of memory space.) See the Parts List in Part 1 of this article for information on ordering the software.

Sequential EPROM addresses are generated in software simply by incrementing a 12-bit counter and output-





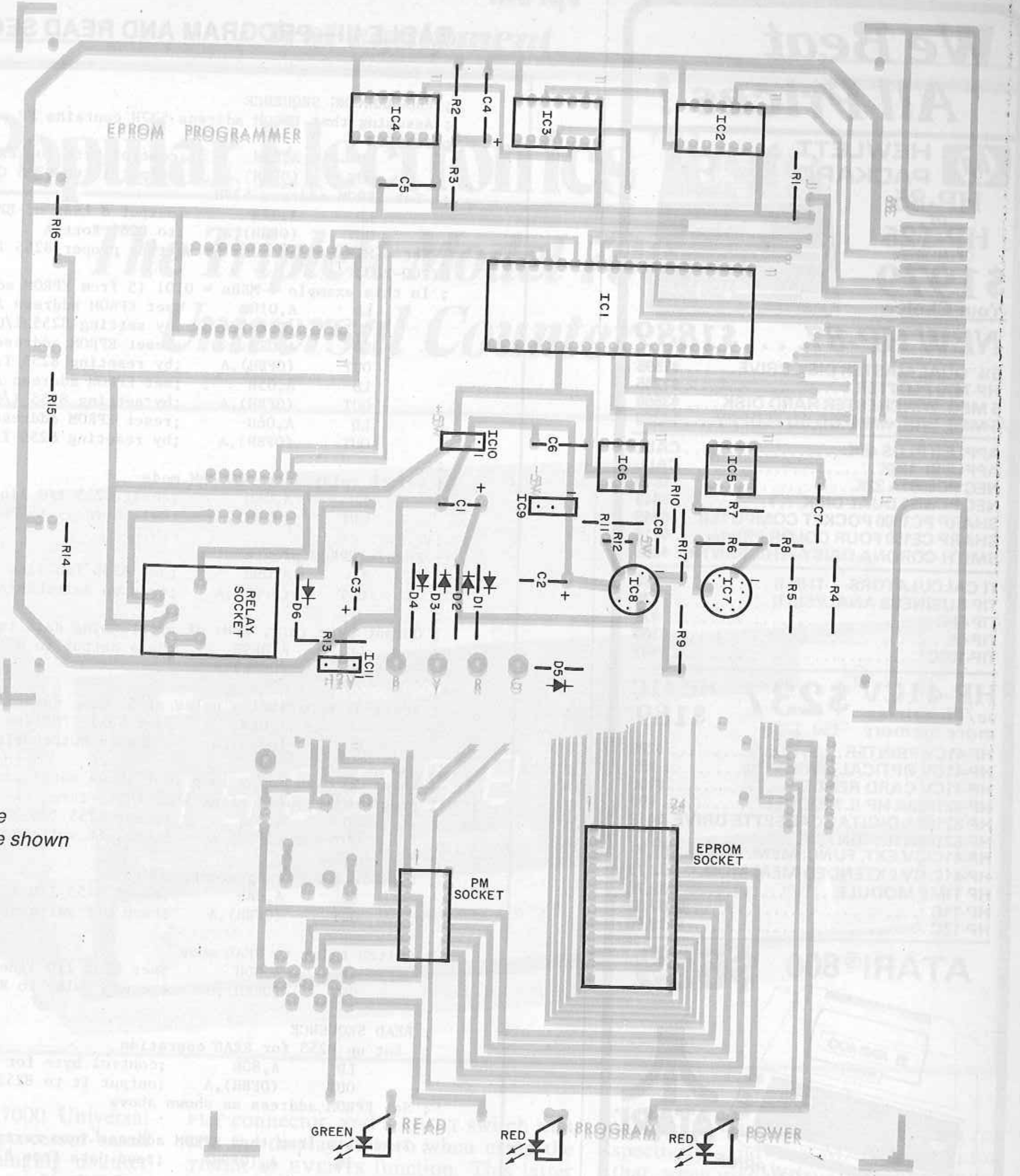


Fig. 7. Foil patterns for the double-sided pc board are shown on opposite page, while component layout is above and at right.

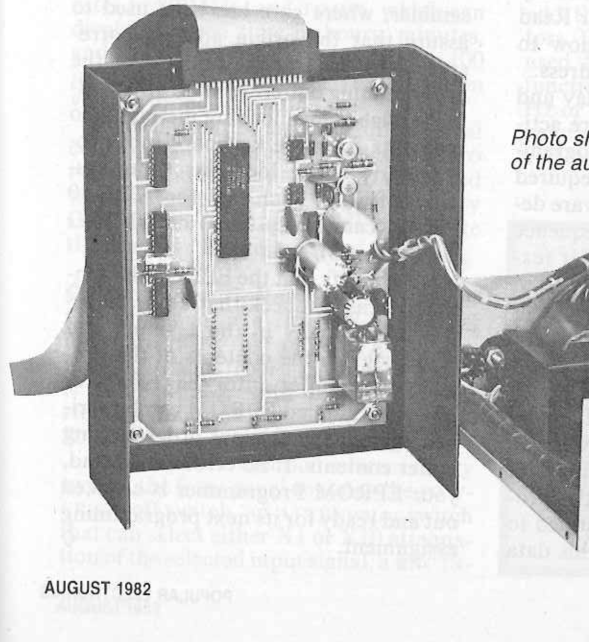


Photo shows internal layout of the author's prototype.

ting the low-order eight-bits to the 8255A's port A while the remaining high-order four-bits determine whether the port C bits (PC0-PC3) are set or reset. Address pointers to the corresponding buffer locations in the computer memory can be determined by adding the buffer starting address to the counter value. Table II shows the appropriate control bytes which must be sent to the 8255A's control port to set or reset the various lines of port C. Also shown in Table II are the control bytes which configure the 8255A for Read or Program modes.

Table III shows how to program and read data for a single location of a 2716 EPROM using Z80 assembly language. (The CPU I/O port addresses and control bytes used are those listed in Tables

We Beat All Prices!

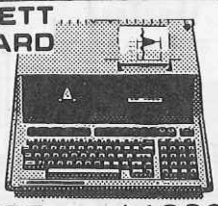
HEWLETT PACKARD

HP-85 or HP-125

CP/M® based.

$1979

Your Choice



NEW HP-87....$1889

5¼"DUAL MASTER DISK DRIVE......	$1595
HP-7470 PLOTTER..................	$1245
5 MEG. WINCHESTER HARD DISK....	$3599
5 MEG. WIN. w/5¼"DD,SD FLOPPY....	$4399
APPLE II PLUS 48K..................	CALL
APPLE III 128K.....................	CALL
NEC PC8001A 32K....................	$849
NEC PC8031 DUAL DISK...............	$849
SHARP PC1500 POCKET COMPUTER...	$249
SHARP CE150 FOUR COLOR PRinter....	$215
SMITH-CORONA DAISY-WHL. PRNTR..	$695
TI CALCULATORS— TI-55 II...........	$36
TIP BUSINESS ANALYSIS II...........	$36
TIP-58C............................	$79
TIP-59.............................	$169
TIP-100C...........................	$149

HP-41CV w/Five times more memory $237 List. $325

HP-41C $189

HP-41CV PRINTER....................	$289
HP-41CV OPTICAL WAND...............	$99
HP-41CV CARD READER................	$165
HP-821060A HP-IL MODULE............	$99
HP-82161A DIGITAL CASSETTE DRIVE,	$419
HP-82162A IL PRINTER...............	$375
HP-41C/CV EXT. FUNC. MEM. MOD. .	$62.75
HP-41C/CV EXTENDED MEM.MOD. .	$62.75
HP TIME MODULE.....................	$62.75
HP-11C.............................	$107.95
HP-12C.............................	$119.95

ATARI® 800 $659

ATARI

NEW LOW PRICE

ATARI 400 $319

ATARI 16K MEMORY MODULE......	$75.00
ATARI 823 80 COL. IMPACT PRNTR....	$565
ATARI 410 PROGRAM. RECORDER.....	$79
ATARI 810 DISK DRIVE...............	$419

—APPLE OWNERS—

APPLE II CP/M® Conversion Special

Included MicroSoft Softcard, (a board with Z-80, with MicroSoft BASIC and CP/M® included). MicroSoft's buffered 16K RAM board, the VIDEX VIDEOTERM 80 Column board, with a soft video switch and inverse video.

*All for only, $599. (Limited time only)

Lowest prices on CCS & SSM APPLE Interface Boards!

Special prices on VisiCalc® and other VisiCorp Software

(Formerly Personal Software)

Personal Computer Systems (315) 478-6800

All prices are subject to change without notice.

609 Butternut St.
P.O. Box 1073
Syracuse, N.Y. 13201

Prices do not include shipping

CIRCLE NO. 38 ON FREE INFORMATION CARD

TABLE III—PROGRAM AND READ SEQUENCE

```
; PROGRAMMING SEQUENCE
; Assuming that EPROM address 537H contains FF program it
; to OE9H
        LD      A,82H       ;control byte for PROGRAM mode
        OUT     (OFBH),A    ;output it to 8255 CONTROL port
;  Set EPROM address 537H:
        LD      A,37H       ;output 8 LSBs of EPROM address
        OUT     (OF8H),A    ;to 8255 Port A
; Set 4 MSBs of address by setting proper 8255 I/O lines
;(PCO-PC3)
; In this example 4 MSBs = 0101 (5 from EPROM address)
        LD      A,01H       ;set EPROM address A8
        OUT     (OFBH),A    ;by setting 8255 I/O line PCO
        LD      A,02H       ;reset EPROM address A9
        OUT     (OFBH),A    ;by reseting 8255 I/O line PC1
        LD      A,05H       ;set EPROM address A10
        OUT     (OFBH),A    ;by setting 8255 I/O line PC2
        LD      A,06H       ;reset EPROM address All
        OUT     (OFBH),A    ;by reseting 8255 I/O line PC3
;
; Switch relay into PROGRAM mode
        LD      A,OCH       ;reset 8255 I/O line PC6 to
        OUT     (OFBH),A    ;switch relay to PROGRAM mode
;
; Enable EPROM programming
        LD      A,09H       ;set 8255 I/O line PC4 to
        OUT     (OFBH),A    ;turn on switchable 25.5V
;
; Output byte (e.g. E9H) of programming data to EPROM
        LD      A,OE9H      ;data output to 8255 Port B
        OUT     (OF9H),A
;
; Apply 5V programming pulse of 50 msec duration
        LD      A,OBH       ;set 8255 I/O line PC5 to
        OUT     (OFBH),A    ;turn on switchable 5V
;
; Enter 50 msec delay loop (not shown here) and turn off
; programming pulse after loop falls thru
        LD      A,OAH       ;reset 8255 I/O line PC5 to
        OUT     (OFBH),A    ;turn off switchable 5V
;
; Disable EPROM programming mode
        LD      A,08H       ;reset 8255 I/O line PC4 to
        OUT     (OFBH),A    ;turn off switchable 25.5V
;
; Switch relay to READ mode
        LD      A,ODH       ;set 8255 I/O line PC6 to
        OUT     (OFBH),A    ;return relay to READ mode
;
; READ SEQUENCE
;  Set up 8255 for READ operation
        LD      A,80H       ;control byte for READ mode
        OUT     (OFBH),A    ;output it to 8255 CONTROL port
; Set EPROM address as shown above
;
; Read data from that EPROM address into register A:
        IN      A,(OF9H)    ;read data from 8255 Port B
```

I and II.) This example shows exactly how the 8255A is configured for Read and Program operations and how to specify a particular EPROM address.

In the Program mode the relay and switchable voltage regulators are activated by the software. The programming pulse is held on for the required programming time using a software delay loop. The programming sequence for other EPROM types is similar (except for the delay loop period). And in some cases, the 5-V and 25.5-V switchable supplies exchange roles as sources for the programming enable signal and the programming pulse.

With the hardware assembly complete and the software driver written, check-out of the EPROM Programmer can proceed by first loading the programming buffer with data destined to reside in the EPROM. Often this data will be a binary file generated by an assembler, where care has been used to assure that the origin address corresponds to the starting address of the programming buffer. A more direct, although laborious, loading method is to type data from the keyboard into programming buffer locations using system software such as the TRS-80 TBUG (cassette systems) or DEBUG (disk systems).

Next, verify that the EPROM is completely erased (check that every bit location is initially in the 1 state), and then program the contents of the programming buffer into the EPROM. The programmed EPROM can be verified against the original programming buffer contents. If no errors are found, your EPROM Programmer is checked out and ready for its next programming assignment. ◇