THE TARGET

September/October 1980

-- an AIM65 newsletter

Those readers that ordered the 1979 volume should now have them in hand. If by chance you do not have yours, please write. Copies are available for $6.00. As it turned out this volume required significantly more work than originally anticipated. I hope it reflects the additional effort expended.

I have learned of an AIM 65 users club that is located in Iowa. The club fees are $8.00 a year which includes a subscription to a monthly newsletter. This club occasionally makes volume purchases for its members. Contact Cedar Valley Computer Association PO Box 671 Marion, IA 52302.

One place to purchase the PL/65 ROMs is from Compas Microsystems PO Box 687 Ames, IA 50010. The price is $125. There may be other places to purchase these ROMs. A review of PL/65 will appear in a later issue of the newsletter.

An item I would like to touch on this issue is voice I/O for the AIM 65. There has been little published, to my knowledge, on computer (micro) voice. There are commercially available boards that give a specific computer voice capability, but that doesn't help AIM 65 users. What I would like to see is the accumulation of resources and then we could build from there. If any readers are aware of any resources or have an uderstanding of the concepts, let me know. Perhaps with the collective efforts of readers we can publish a software and hardware model. It should be built with as little hardware as possible to allow readers with little hardware experience to use the model as well.

Tod Loofbourrow's "How To Build A Computer Controlled Robot" has a chapter on voice input to a close relative, KIM-1. I believe this book from Hayden could get us off on the right foot.

Thanks to those that responded to my questions in the May/June issue. Most respondees have about 4K of RAM, either or both of the assembler ROM and BASIC ROMs, use both ROM sets equally, are good to intermediate programmers. Many of the answers suggested peripherals such as additional memory, terminals or video displays, but very little was mentioned about software.

Donald Clem

Donald Clem

TOUCH-TONE Dialer

Larry Hollibaugh
1708 S.E. Bybee
Portland, OR 97202

I have been experimenting with sound effects on my AIM 65, using the AY-3-8910 Programmable Sound Generator (PSG) from General Instrument. While I was hooking it up to my computer, my wife asked me, "So, what can it do?". After a week of whistles and beeps, phasors and falling bombs, she still wanted to know, "What can it do?". So now it dials the telephone. This seems to have answered her question, even though it only scatches the surface of possibilities for this remarable device.

For those of you unfamiliar with the AY-3-8910 PSG, it is a Large Scale Integration I.C. which can produce a wide variety of complex sounds under software control. It contains three tone generators, a noise generator, three mixers, an envelope generator, amplitude control and three D/A convertors. Also in the chip are two 8-bit I/O ports which are independent of the sound generator. This is a complex and powerful device. A detailed discussion of its internal architecture is beyond the scope and intent of this article. The reader is referred to an excellent article by Steve Ciarcia entitled "Sound Off", in the July, 1979 issue of Byte. Or write to General Instrument Corp., Microelectronics Division, 600 W. John St., Hicksville, NY 11802, and request the AY-3-8910 Data Manual.

The purpose of this article is to describe how I used the PSG to produce the dual-tone frequencies used in Touch-Tone dialing, along with a method of storing and retrieving phone numbers to be dialed.

The numbers are stored in memory using AIM's built-in text editor. As many numbers as memory permits may be stored in the text buffer. They must be stored in the format of (LABEL)=(NUMBER) where (LABEL) is any character string to uniquely identify (NUMBER), with the exception that it may not start with the "#" symbol. (NUMBER) is the actual number to be dialed, and it can include dashes, etc., to make it more readable. The number must be terminated with a space or carr. return, and only one entry per line will be recognized by the dialing program. An equals sign (=) is used to seperate (LABEL) and (NUMBER).

Listing 1 is the program to select and dial the numbers stored in the text buffer. The program first prompts the user with a slash (/). The number to be dialed may now be specified in one of three ways:

1. By entering enough characters of (LABEL), starting with the first char., to distinguish it from any that precede it in the text buffer. Type RETURN to dial. The program starts at the top of the text buffer and dials the first number whose label starts with the char.'s entered. If no match is found, ERROR is displayed.

2. By entering the "#" symbol as the first character after the prompt. This is used to dial a number that may not be in the text buffer. The program will re-prompt with "=". Now enter the actual number to be dialed, and type RETURN.

3. By typing RETURN as the first character after the prompt. This will redial the last number dialed.

Actual dialing is accomplished via acoustic coupling (holding the phone's mouthpiece up to the PSG's speaker), or direct connection to the phone lines through a phone company-provided interface device. Be careful about legal restrictions of direct connection.

Figure 1 shows how I have the PSG connected on my AIM. If you choose to connect yours differently, the software will need to be changed accordingly. The subroutines SETUP, ADDRS, and SET1 contain all of the 6522 VIA control instructions.

Of special importance is the frequency of the clock for the PSG. The data tables labeled TBL1 and TBL2 at the end of Listing 1 are for a PSG operating at 1.78977 MHz. To use at a different frequency, the following formula may be used to calculate the necessary values for the data tables:

$$TP10 = \frac{fCLOCK}{16fT}$$

Where:
TP10 = decimal equivalent of tone period (convert to hex for data tables)
fCLOCK = input clock frequency
fT = desired tone frequency

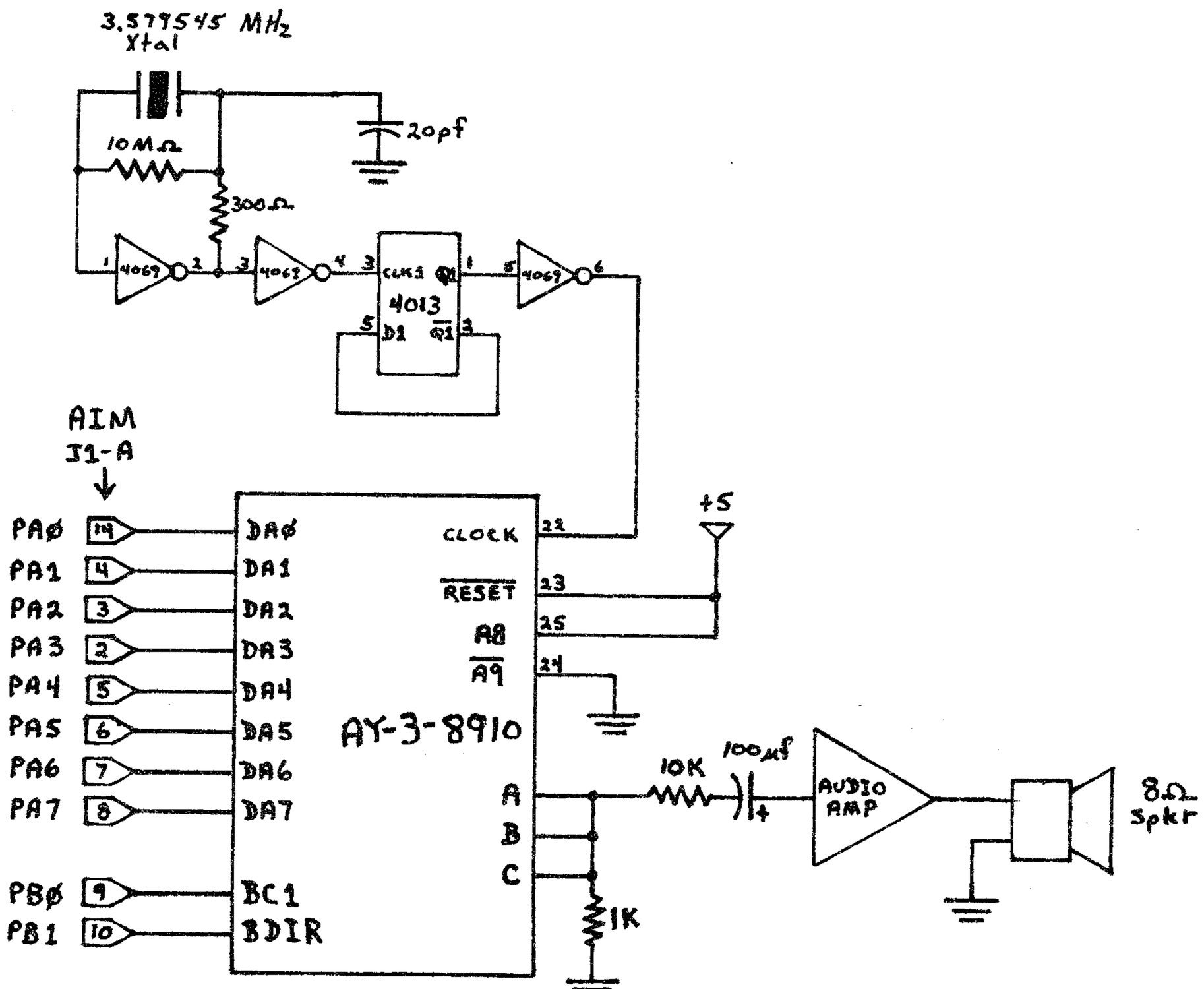
Figure 2 shows the frequencies used in Touch-Tone dialing, and how they relate to the data tables in Listing 1.

The program is fully commented and should be fairly easy to understand. It makes extensive use of AIM's editor and monitor subroutines to conserve memory. The routine NXLINE is a loose copy of UPNO in the monitor listing, and the overall search routine is a modified version of AIM's FCH sub. The tones are generated by the routine starting at TONES.

I hope this helps you explain those strange sounds to YOUR wife. "Look, dear, it does something!".

FIGURE 2

TELEPHONE DIGIT	FREQUENCIES (Hz)	HEXADECIMAL DATA
0	941,1336	77,54
1	697,1209	A0,5C
2	697,1336	A0,54
3	697,1477	A0,4C
4	770,1209	91,5C
5	770,1336	91,54
6	770,1477	91,4C
7	852,1209	83,5C
8	852,1336	83,54
9	852,1477	83,4C
*	941,1207	77,5C
#	941,1477	77,4C



```
;TOUCH-TONE TELEPHONE DIALER
;  USING THE AY-3-8910 PSG
;
;  BY LARRY R. HOLLIBAUGH
;
;ZERO PAGE USAGE:
NOWLN=$DF
STRING=$EB
FLAG=$FE
YSAV=$FF
;
;6522 VIA ACCESS:
UDRB=$A000
UDRA=$A001
UDDRB=$A002
UDDRA=$A003
;MONITOR EQUATES
CKERO=$E38E
EQUAL=$E7D8
PLS1=$E837
BLANK=$E83E
READ=$E93C
RDRUB=$E95F
OUTPUT=$E97A
CRCK=$EA24
HEX=$EA7D
PLNE=$F727
TOPNO=$F8BC
ATBOT=$F8E9
ADDA=$F92A
;
;USER ACCESS HOOK;
*=$10C
JMP DIAL
;
*=$200
;MAIN PROGRAM STARTS HERE
;ENTRY POINT TO DIAL USER SPECIFIED #
DIAL JSR PSL1 ; PROMPT USER WITH "/"
LDY #0  ;SET UP FOR DELETES
FIND1 JSR RDRUB  ;INPUT CHARACTER
CPY #0  ;FIRST CHAR?
BNE FIND2
STY FLAG  ;INIT FLAG FOR TEST SRCH
CMP #$D  ;REDIAL LAST #?
BEQ REDIAL
CMP #"#"  ;CODE FOR NUMERICAL ENTRY
BNE FIND2
STA STRING
JSR EQUAL  ;PROMPT USER WITH "="
INC FLAG  ;BYPASS TEXT SRCH
INY  ;ACCOUNT FOR "=" IN STRING
FIND2 CMP #$D  ;DONE?
BEQ FIND3
STA STRING,Y  ;SAVE IT
INY  ;NEXT CHAR
CPY #$10  ;16 CHAR'S MAX
BNE FIND1
FIND3 LDA #$D  ;IN CASE OF NUM. ENTRY:
STA STRING,Y  ;END STRING WITH CR...
LDA #<STRING  ;SET UP STRING...
STA NOWLN  ;AS CURRENT LINE
LDA #>STRING
STA NOWLN+1
LDA FLAG  ;FLAG NOT ZERO MEANS...
BNE REDIAL  ;NUMERIC ENTRY-DIAL IT
;INITIATE TEXT SEARCH
STY YSAV  ;CHARACTER ENTRY COUNT
JSR TOPNO  ;SET NOWLN=TOP OF BUFFER
JMP FIND5  ;START SEARCH
NXLINE LDY #0
JSR ATBOT  ;CURRENT LINE AT BOTTOM?
BCS NONE
NXT1 LDA (NOWLN)Y  ;TEST FOR END
BEQ NONE  ;OF TEXT BUFFER
INY
CMP #$D  ;LOCATE END OF LINE
BNE NXT1
TYA
JSR ADDA  ;ADVANCE TO NEXT LINE
FIND5 LDY #0  ;START WITH FIRST CHAR
FIND7 LDA (NOWLN)Y  ;GET CHARACTER
BNE FIND8 ;END OF TEXT BUFFER?
NONE JMP CKERO  ;END OF BUFFER=NO MATCH
FIND8 CMP #$D  ;END OF LINE?
BEQ NXLINE
CMP STRING,Y  ;MATCH?
BNE NXLINE  ;NO-NEXT LINE
INY  ;YES-NEXT CHAR
CPY YSAV  ;ALL CHAR'S TESTED?
BNE FIND7
;
REDIAL JSR CRCK  ;PRINT DISPLAY
JSR BLANK  ;OUTPUT A SPACE
JSR PLNE  ;SHOW CURRENT LINE
JSR SETUP  ;GUARANTEE 6522 STATE
JSR CLRPSG  ;RESET PSG
LDY #0
TAB INY  ;ADVANCE THROUGH LINE...
LDA (NOWLN)Y
CMP #$D  ;UNTIL CR...
BEQ EXIT
CMP #"="  ;OR EQUAL SIGN FOUND
BNE TAB
TONES INY
LDA (NOWLN)Y  ;NEXT # TO BE DIALED...
CMP #$D  ;OR "SPACE" ENCOUNTERED
BEQ EXIT
JSR HEX  ;CONVERT TO HEX
BCS TONES  ;IGNORE NON-NUMERIC
STY YSAV  ;SAVE # POINTER
TAY  ;USE Y AS POINTER TO PSG DATA
```

```
LDX #0   ;PSG ADDR
LDA TBL1,Y   ;VALUE FOR PSG TONE A
JSR SET1   ;SEND TO PSG
LDX #2   ;PSG ADDR
LDA TBL2,Y   ;VALUE FOR PSG TONE B
JSR SET1   ;SEND TO PSG
LDX #7   ;PSG ENABLE REG ADDR
LDA #$FC   ;ENABLE TONES ON CH A&B
JSR SET1
INX   ;PSG CH A AMPLITUDE
LDA #$F   ;MAX AMPLITUDE
JSR SET1
INX   ;PSG CH B AMLITUDE
JSR SET1   ;MAX AMLITUDE
LDX #$3C   ;VALUES FOR TONE DURATION
LDY #0
JSR DELAY
JSR CLRPSG   ;TURN OFF TONES
LDX #$14   ;VALUE FOR SILENCE DURATION
JSR DELAY
LDY YSAV   ;RECOVER # POINTER
JMP TONES   ;DIAL NEXT NUMBER
;
;SUBROUTINES START HERE
SETUP LDA UDDRB
ORA #3   ;DDRB BITS 0 & 1 = OUTPUTS...
STA UDDRB   ;BITS 2 - 7 UNCHANGED
LDA UDRB
AND #$FC   ;DRB BITS 0 & 1 = 00...
STA UDRB   ;BITS 2 - 7 UNCHANGED
LDA #$FF
STA UDDRA   ;DDRA = OUTPUTS
EXIT RTS
;
CLRPSG LDX #$F   ;PSG REG ADDR
LDA #0   ;PSG DATA
CLRLP JSR SET1   ;WRITE TO PSG
DEX   ;NEXT ADDR
BPL CLRLP   ;UNTIL ALL SET TO ZERO
RTS
;
ADDRS STX UDRA   ;X=ADDR TO PSG
PHA
LDA UDRB
ORA #3   ;"LATCH ADDRESS" MODE
ADDR1 STA UDRB
AND #$FC   ;"PSG INACTIVE" MODE
STA UDRB
PLA
RTS
;
SET1 JSR ADDRS   ;WRITE X TO PSG AS ADDR
STA UDRA   ;ACC TO PSG AS DATA
PHA
LDA UDRB
ORA #2   ;"WRITE TO PSG" MODE
BNE ADDR1   ;BRANCH ALWAYS
;
DELAY DEX
BNE DLYLP
RTS
DLYLP DEY
BEQ DELAY
JMP DLYLP
;
;DATA FOR PSG TONE FREQUENCIES
;ONE BYTE FROM EACH TABLE FOR
;DUAL-TONES. C-F = SILENCE
TBL1 .BYT $77,$A0   ;0,1
.BYT $A0,$A0,$91,$91;2,3,4,5
.BYT $91,$83,$83,$83;6,7,8,9
.BYT $77,$77,0,0,0,0;*,#,C,D,E,F
                    ;(*=A,#=B)
;
;
TBL2 .BYT $54,$5C   ;0,1
.BYT $54,$4C,$5C,$54 ;2,3,4,5
.BYT $4C,$5C,$54,$4C ;6,7,8,9
.BYT $5C,$4C,0,0,0,0 ;*,#,C,D,E,F
                     ;(*=A,#=B)
;
LAST .END
```

```
;AY-3-8910 "SOUND"
;GEN. PURPOSE SUBS
;
;BY LARRY HOLLIBAUGH
;
;
;ZERO PAGE USAGE:
*=$10
PARAM *=*+$10
XSAV *=*+1
;
;6522 VIA ACCESS:
*=$A000
UDRB *=*+1
UDRAH *=*+1
UDDRB *=*+1
UDDRA *=*+1
;
*=$0200
;INITIALIZE 6522 VIA
SETUP LDA #3
STA UDDRB
SETUP1 LDA #0
STA UDRB
LDA #$FF
STA UDDRA
RTS
;
;TURN OFF ALL SOUND
CLRPSG LDX #$F
LDA #0
CLRLP JSR SET1
DEX
BPL CLRLP
RTS
;
;'X' = PSG ADDRESS
ADDR STX UDRAH
PHA
LDA #3
STA UDRB
LDA #0
STA UDRB
PLA
RTS
;
;WRITES DATA REF. BY
;X TO PSG
SETONE LDA PARAM,X
SET1 JSR ADDR
WRITE STA UDRAH
PHA
LDA #2
STA UDRB
LDA #0
STA UDRB
PLA
RTS
;
;WRITES ALL OF PARAM
;TO PSG AS DATA
SETALL STX XSAV
LDX #$D
SETA1 JSR SETONE
DEX
BPL SETA1
LDX XSAV
RTS
;
;READ FROM PSG AT
;ADDR REF. BY X
FETCH1 JSR ADDR
RDPSG LDA #0
STA UDDRA
LDA #1
STA UDRB
LDA UDRAH
PHA
JSR SETUP1
PLA
AND BITS,X
RTS
;
BITS .BYT $FF,$F,$FF
.BYT $F,$FF,$F,$1F
.BYT $FF,$1F,$1F,$1F
.BYT $FF,$FF,$F
.BYT $FF,$FF
;
LAST .END
```

Offset Load

Steve Bresson
1302 Strawberry Lane
Hanover, MD 21076

OFFSET LOAD

This program loads an object program from tape at an offset from the save address. This is useful for those who have prom programmers. It could also be useful for relocatable programs-save one copy to tape, then load it into where its needed.

Assemble the program to tape, load at an offset so the program is in RAM, then program the PROM with it. It displays the number of hex bytes in each record, the save address, and one past the end address.

```
<C>OFFSET=FF00
IN=T F=OLODC T=2

09  000C-0015
18  0E00-0E18
17  0E18-0E2F
16  0E2F-0E45
18  0E45-0E5D
18  0E5D-0E75
18  0E75-0E8D
11  0E8D-0E9E
00  FF09-
--END--

<*>=E00
<G>/
OFFSET=FE00
IN=T F=OLODC T=2

09  FF0C-
 MEM FAIL FF0C


<E>
EDITOR
```

THE ABOVE RUN USED THE COPY WHICH WAS LOADED IN AT AN OFFSET OF $FF00. WHEN IT ATTEMPTED TO SAVE <F1> AT $FF0C IT ENCOUNTERED A WRITE ERROR SINCE ROMS DON'T WRITE TO WELL.

```
--------------------

==0000
; --- OLOAD ---
LOAD OBJECT PROGRAMS
 AT AN OFFSET
;V1A.7.14.80.SLB
--------------------

==0000 ADDNE=$EAB1

==0000 ADDR=$A41C

==0000 CRLOW=$EA13

==0000 WHEREI=$E840

==0000 INALL=$E993

==0000 CLRCK=$EB4D

==0000 CHEKAR=$E54B

==0000 NUMA=$EA46

==0000 BLANK2=$E83B

==0000 WRITAZ=$E2DB

==0000 OUTPUT=$E97A

==0000 RBYTE=$E3FD

==0000 STBYTE=$E413

==0000 CKSUM=$A41E

==0000 DU13=$E520

==0000 CKERR=$E385

--------------------

==0000
        *=$10C
==010C
4C000F JMP OLOAD
4C000F JMP OLOAD
4C000F JMP OLOAD
--------------------

==0115
        *=$F00
==0F00 OLOAD
20800F JSR MSG0
20B1EA JSR ADDNE
AD1CA4 LDA ADDR
8D9E0F STA OFF
AD1DA4 LDA ADDR+1
8D9F0F STA OFF+1
==0F12
2013EA JSR CRLOW
2048E8 JSR WHEREI
==0F18 OLOAD1
2093E9 JSR INALL
C93B   CMP #';'
D0F9   BNE OLOAD1
2013EA JSR CRLOW
204DEB JSR CLRCK
204BE5 JSR CHEKAR
==0F23
48     PHA
2046EA JSR NUMA
2083F8 JSR BLANK2
204BE5 JSR CHEKAR
8D1DA4 STA ADDR+1
204BE5 JSR CHEKAR
==0F38
18     CLC
6D9E0F ADC OFF
8D1CA4 STA ADDR
AD1DA4 LDA ADDR+1
6D9F0F ADC OFF+1
8D1DA4 STA ADDR+1
==0F48
20DBE2 JSR WRITAZ
A92D   LDA #'-'
207AE9 JSR OUTPUT
68     PLA
F01F   BEQ OLOAD4
AA     TAX
==0F54 OLOAD2
20FDE3 JSR RBYTE
2013E4 JSR STBYTE
CA     DEX
D0F7   BNE OLOAD2
20DBE2 JSR WRITAZ
;CHECK SUM OK??
20FDE3 JSR RBYTE
CD1FA4 CMP CKSUM+1
==0F66
D015   BNE ERR
20FDE3 JSR RBYTE
CD1EA4 CMP CKSUM
D00D   BNE ERR
F0A6   BEQ OLOAD1
==0F72 OLOAD4
2013EA JSR CRLOW
A208   LDX #M2-M1
20820F JSR MSG
4C20E5 JMP DU13
==0F7D ERR
4C85E3 JMP CKERR
--------------------

==0F80 MSG0
A200   LDX #0
==0F82 MSG
BD8E0F LDA M1,X
F006   BEQ MSG1
207AE9 JSR OUTPUT
E8     INX
D0F5   BNE MSG
==0F8D MSG1
60     RTS

==0F8E M1
0F46   BYT 'OFFSET='
,0
00
==0F96 M2
2D2D   BYT '--END--
',0
00

==0F9E OFF
       *=*+2
--------------------

       END
ERRORS= 0000
```

STEVE BRESSON
1302 STRAWBERRY LN
HANOVER,MD 20176

POWERFUL AIM-65 FLOPPY DISK SYSTEM

PRICE $945

Everything is here. Just plug it in and start execution at $9000, then ADOS Operating System is running.

System includes:
FP-950 Controller Board.
ADOS Operating System on a 2532 EPROM.
Single sided/double density drive with power supply & case. One Dysan diskette.
Interface cable. Operating manual.

* Powerful ADOS Operating System allows access of Basic data as well as program files IN and OUT from floppy disk under AIM-65 Basic program control (available library utility).
* Able to assemble multiple source files from disk and routing object code or listing outputs directly to either: disk , memory or to the on board compatible centronics printer port.
* Able to save and load text/object files directly from the Editor or Monitor on the AIM-65.
* Includes infromation on accessing the disk files from user program control.
* Able to execute programs directly from disk.
* Dynamic allocation of files.
* Block transfer mode allows loading 32 K bytes in 4 seconds.

DSAIM-65

The DSAIM-65 is a really complete low cost development system for the 6500 microprocessor family with unsurpassed capability for just

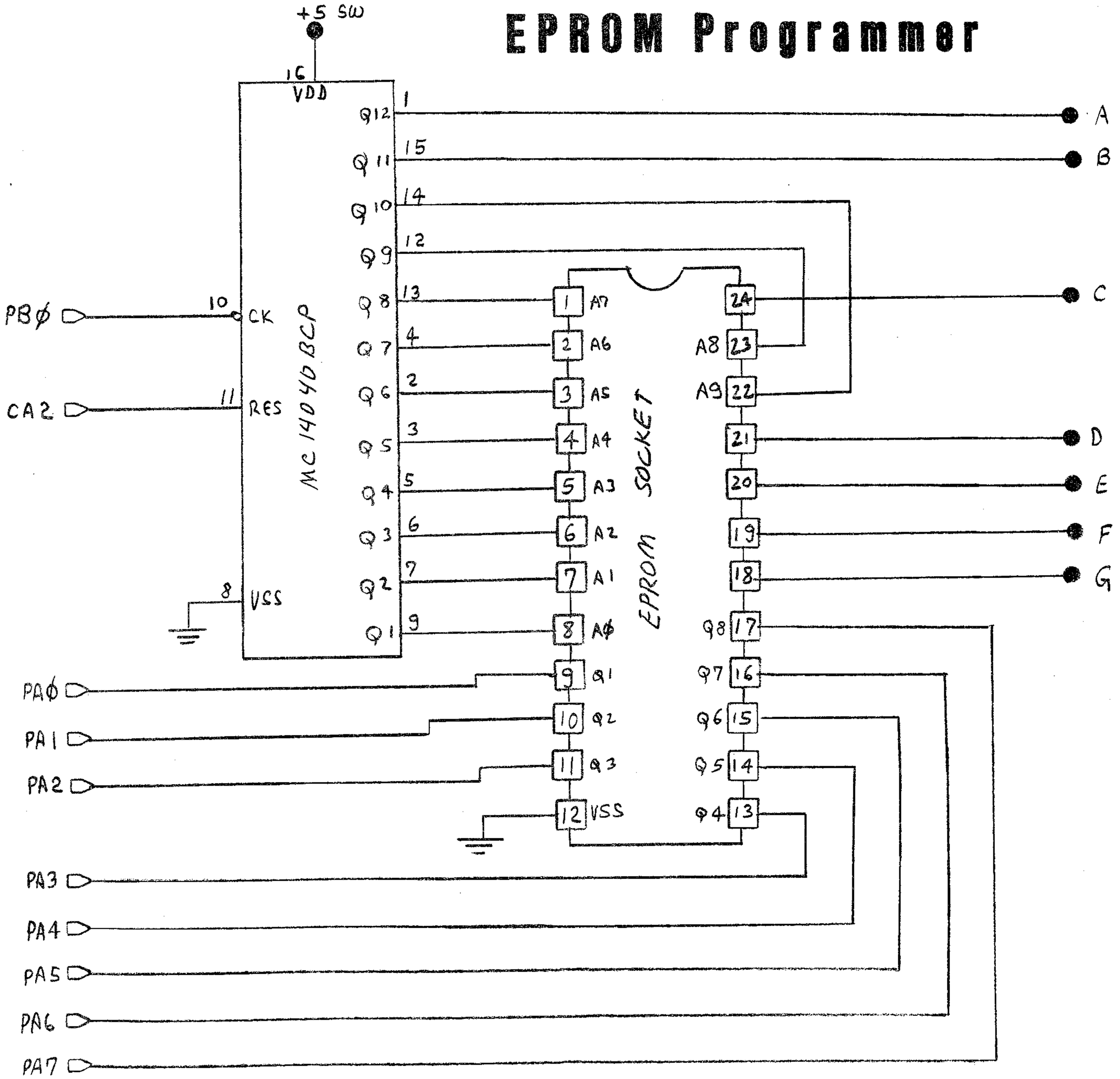
PRICE $3795

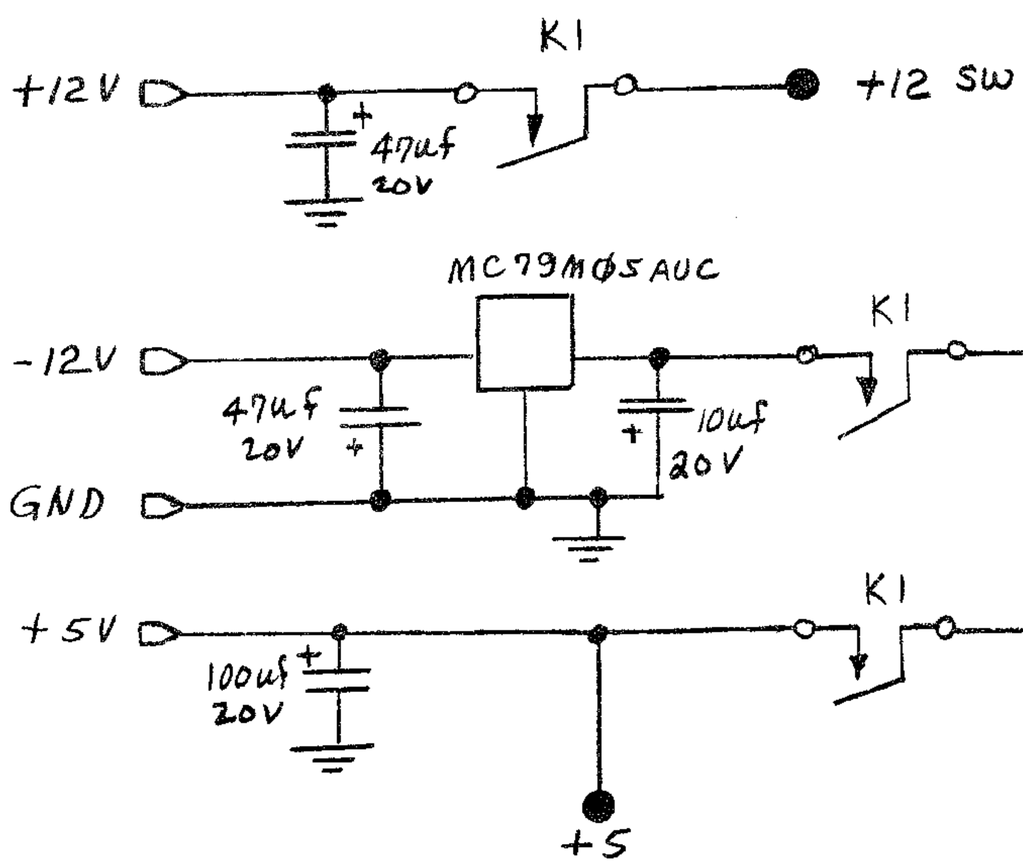
The system includes:
1 AIM-65 Microcomputer.
1 High level language BASIC ROM set.
1 FP-950 Mini-floppy Disk Controller module, capable of driving up to 4 single/double sided, single/double density drives. With Centronic's type printer interface.
1 ADOS Operating System on 2532 EPROM (see powerful AIM-65 Floppy Disk System above)
1 Dysan diskette containing library utilities to support the handling of data files under BASIC language program control.
2 Single sided, double density drivers with power supply and metal cases.
2 Interface cables for floppy drivers
1 CRT-80 Video Controller module with 2k bytes of software on board to replace the 20 character single line display for 80,64 or 40 characters by 25 lines video display. Upper/lower case ASCII characters, 138 semigraphic characters plus 18 control characters. Refresh RAM not on address space of CPU with no wait states when updating. Reverse video for each character.
1 Video Monitor display for 80 x 25 , 64 x 25 or 40 x 25 lines.
1 Coax cable for video monitor.
1 MB-32K Dynamic RAM module , capable to extend the memory up to 64 K bytes. Single +5 v power supply. Totally transparent refresh, no cycle stealing. Memory selectable in 4K blocks by switches. Allows bank selection.
1 EMB-6 Expansion Mother Board. Enabled in 4K increments by switches. Fully buffered. 6 card slots. EXORciser bus compatible. Straight extended from expansion connector on the AIM-65
1 Power supply with +5V @ 6A , 24 V @ 1 A
1 730 Centronics dot matrix printer

Without printer substract $845

APPLIED BUSINESS COMPUTER CO.
707 S. STATE COLLEGE., SUITE G,
FULLERTON, CA. 92631 TEL.(714)871-1411

EPROM Programmer



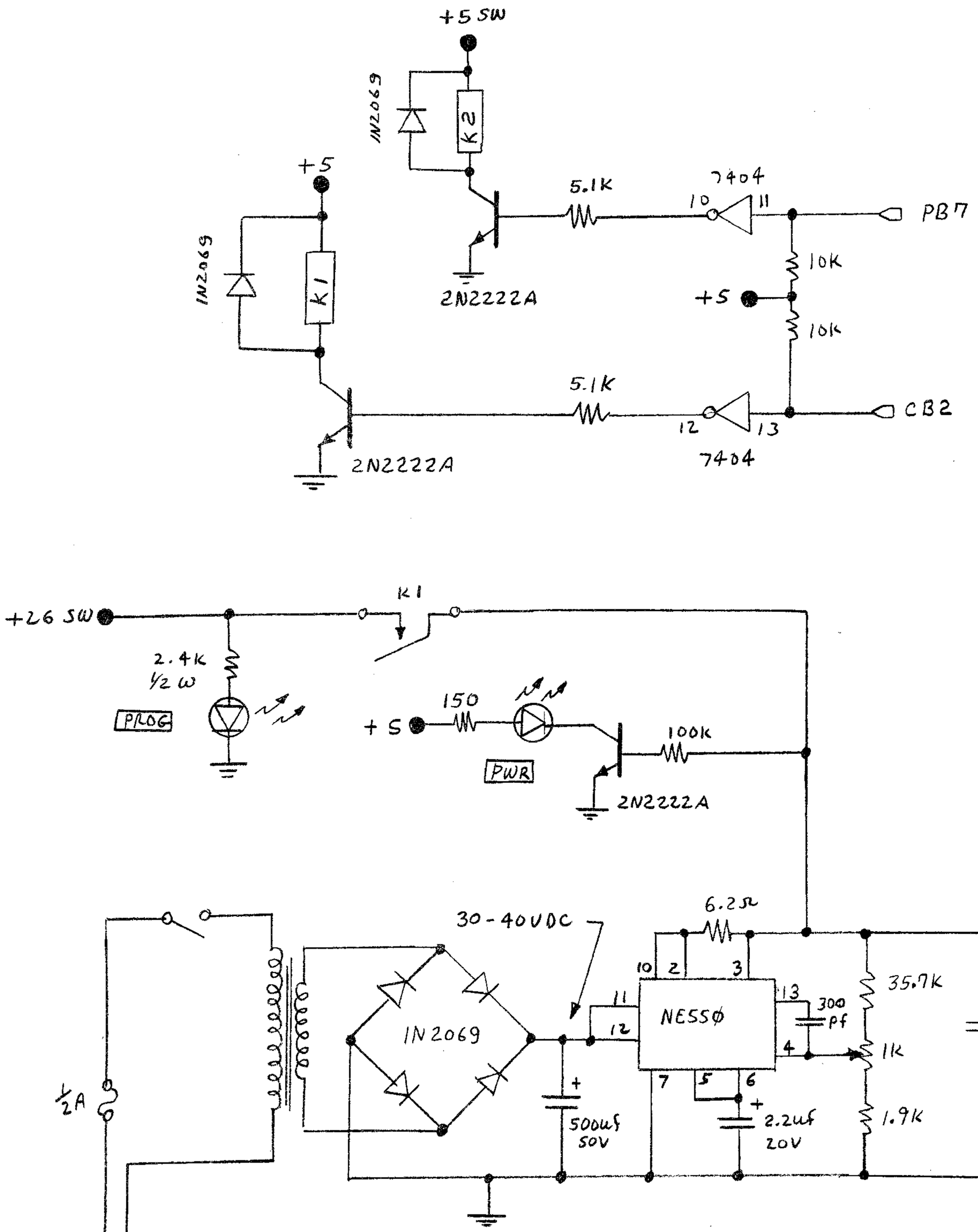


Dick Buchen
21 Alta Loma Dr
Vallejo,CA 94590

NOTE:
R = READ
P = PROGRAM
V = VERIFY

EPROM	PB6	PB7
2708	0	0
2716	0	1
2516	1	0
2532	1	1

+5
10K
10K
PB6
PB5
2716
R P V +5V 0V
+5SW
K2
C
+5 SW
2516 & 2532
R P V +26V +5V
-5 SW
D
+26 SW
IN2069
K2
+5 SW
E
B
+12 SW
F
B
G
A
2708
2716
2516
2532
+12 SW
2.7K
2N2222A
+5
10K
11.5K
14
6
5
7404
PB4
R P V 12V 0V
2708
+5
10K
7404
4
3
11
PB3
R P V 2516
+26 SW
620
2N2222A
1MSEC
R P V 26V 0V
2708 & 2716
5.1K
7404
8
9
PB2
10K
+5
10K
2
1
7404
PB1
2516 R P V
2532 R P V
50MSEC
P


+5 SW
IN2069
K2
+5
5.1K
7404
10
11
PB7
IN2069
K1
10K
+5
2N2222A
10K
5.1K
CB2
12
13
2N2222A
7404
K1
+26 SW
2.4K
½ W
PLOG
150
+5
PWR
100K
2N2222A
30-40VDC
6.2Ω
IN2069
10
2
3
11
12
NESSØ
13
300
pf
4
35.7K
1K
10uf
35V
½A
7
5
6
500uf
50V
2.2uf
20V
1.9K

```
>?
<O?
<ND
!A
;
;
;
;EPROM D BUCHEN JUL 80
;
;
;THIS PROGRAM WAS WRITTEN SEP 79
;
;
;THIS PROGRAM IS TO BE USED IN CONJUNCTION WITH
;A PROM PROGRAMER BOX THAT CONNECTS TO THE AIM
;VIA I/O PORT. IT ALLOWS YOU TO PROGRAM 2708,
;2716, 2516, AND 2532 EPROMS. THREE VOLTAGE 2716
;EPROMS ARE REFERED TO AS 2716, SINGLE VOLTAGE 2716
;EPROMS ARE REFERED TO AS 2516. THE PROGRAM CHECKS IF
;THE EPROM IS BLANK, AND VERIFY IF IT IS
;PROGRAMED PROPERLY.
;
;EPROMS AND THEIR ADDRESS SPACE ARE LISTED BELOW:
;
;       P/N         SIZE    ADDRESS
;
;       2708         1K     000-3FF
;       2716         2K     000-7FF
;       2516         2K     000-7FF
;       2532         4K     000-FFF
;
;
;TO PROGRAM AN EPROM FOLLOW THIS PROCEDURE:
;
;       1.CONNECT PROGRAMER TO VIA AND AC OUTLET.
;       2.SET SWITCH ON PROGRAMER TO MATCH EPROM.
;       3.INSTALL EPROM IN SOCKET.
;       4.START PROGRAM.
;            A.ENTER EPROM PART NO.
;            B.ENTER START ADDR. OF DATA TO BE PROGRAMED.
;            C.ENTER START & END ADDR. OF WHERE DATA IS
;              STORED IN EPROM.
;            D.CHECK IF BLANK, PROGRAM, AND VERIFY EPROM.
;            E.REMOVE EPROM (PROG LED MUST BE OFF)
;
;       ** REMEMBER! 3V 2716=2716 AND 1V 2716=2516 **


;
;
;AIM SUBROUTINES
;
;
  MON   =     $E182   ;AIM MONITOR
  OUTPT =     $E97A   ;OUTPUTS ASCII CHAR IN A TO D/P
  READ  =     $E93C   ;INPUTS ASCII CHAR FROM KB TO A
  CRLOW =     $EA13   ;OUTPUTS CR AND LF
  NUMA  =     $EA46   ;CONV. 2 HEX NOS. IN A FROM BIN. TO ASCII
  ADDIN =     $EAAE   ;GET 4 BYTE ADDR. AND STORE AT ADDR.
  ADDR  =     $A41C   ;4 BYTE ADDR. (LOW BYTE, HIGH BYTE)
  FROM  =     $E7A3   ;OUTPUTS "FROM =" TO D/P & ENTERS ADDR.
  TO    =     $E7A7   ;OUTPUTS "TO =" TO D/P & ENTERS ADDR.
  SPACE =     $E83B   ;OUTPUTS TWO SPACES TO D/P
  POINT =     $A415   ;DISPLAY POINTER
  PRINT =     $A416   ;PRINTER POINTER
;
;
;ZERO PAGE SCRATCHPAD
;
;
  DEL1  =     $00     ;DELAY SUBRTN
  DEL2  =     $01     ;DELAY SUBRTN
  PROM  =     $02     ;EPROM PART NO.
  PRST  =     $04     ;PROG. START ADDR.
  EPST  =     $06     ;EPROM START ADDR.
  EPND  =     $08     ;EPROM END ADDR.
  LPCNT =     $0A     ;NO. OF PROG. CYCLES, IN HEX
  TYPE  =     $0C     ;TYPE OF EPROM CODE
  COUNT =     $0D     ;TEMP. EPROM ADDR.
  LNG   =     $0F     ;TEMP. MESSAGE LENGTH
  MESG  =     $10     ;TEMP. MESSAGE ADDR.
  HEX   =     $12     ;TEMP. HEX TO DEC. SUBRTN.
  DECL  =     $13     ;NO. OF PROG. CYCLES, IN DECIMAL
  FLAG  =     $14     ;USED IN PROG AND VERIFY SECTIONS OF PROG.
  PTEMP =     $15     ;TEMP. PROG. ADDR.
;
;
;VIA REGISTERS
;
;
  DRA   =     $A001   ;AIM USER VIA
  DRB   =     $A000
  DDRA  =     $A003
  DDRB  =     $A002
  PCR   =     $A00C
;
  DRA2  =     $A480   ;AIM KEYBOARD RIOT
  DRB2  =     $A482   ; "      "      "
;
;
;MAIN PROGRAM
;
                .OR  $0A00
;
;
0A00  4CE40A        JMP  START
;
;
0A03  4550524F4D20504152542045554D424552
            MESS1 .BT  'EPROM PART NUMBER'
;
;
0A14  43484B2E20535749544348204F534954494F4E
            MESS2 .BT  'CHK. SWITCH POSITION'
;
;
0A28  50524F4752414D20535441525420414444524553353
            MESS3 .BT  'PROGRAM START ADDRESS'
;
;
0A3D  4550524F4D20414444524553353
            MESS4 .BT  'EPROM ADDRESS'
;
;
0A4A  57524F4E472050415254204E554D424552
            MESS5 .BT  'WRONG PART NUMBER'
;
;
0A5B  43484B2049462042414E4B3F20592F4E
            MESS6 .BT  'CHK IF BLANK? Y/N'
;
;
0A6C  52454144
            MESS7 .BT  'READ'
;
;
0A70  4E4F5420424C414E4B
            MESS8 .BT  'NOT BLANK'
;
;
0A79  50524F4752414D3F20592F4E
            MESS9 .BT  'PROGRAM? Y/N'
;
;
0A85  50524F4752414D
            MESSA .BT  'PROGRAM'
;
;
0A8C  5645524946593F20592F4E
            MESSB .BT  'VERIFY? Y/N'
;
;
0A97  564552494659
            MESSC .BT  'VERIFY'
;
;
0A9D  5645524946592045525F52
            MESSD .BT  'VERIFY ERROR'
;
;
0AA9  5645524946592043F4D504C455445


            MESSE .BT  'VERIFY COMPLETE'
;
;
0AB8  1114150D11110409 0C070B060C0F
            LNGTH .BT  17,20,21,13,17,17,4,9,12,7,11,6,12,15
;
;
0AC6  030A    ADDS  .WD  MESS1
0AC8  140A          .WD  MESS2
0ACA  280A          .WD  MESS3
0ACC  3D0A          .WD  MESS4
0ACE  4A0A          .WD  MESS5
0AD0  5B0A          .WD  MESS6
0AD2  6C0A          .WD  MESS7
0AD4  700A          .WD  MESS8
0AD6  790A          .WD  MESS9
0AD8  850A          .WD  MESSA
0ADA  8C0A          .WD  MESSB
0ADC  970A          .WD  MESSC
0ADE  9D0A          .WD  MESSD
0AE0  A90A          .WD  MESSE
;
;
0AE2  1601
                TAB   .BT  $16,1

;
;
0AE4  A900    START LDA  #0
0AE6  8D03A0        STA  DDRA       ;PA0-PA7=INPUTS
0AE9  A99F          LDA  #$9F
0AEB  8D02A0        STA  DDRB       ;PB0-PB4,PB7=OUT PB5=PB6=INPUTS
0AEE  A98A          LDA  #$8A
0AF0  8D00A0        STA  DRB        ;PB1=PB3=PB7=1 PB0=PB2=PB4=0
0AF3  A9EE          LDA  #$EE
0AF5  8D0CA0        STA  PCR        ;CA2=CB2=1
;
;
0AF8  A200    REDO  LDX  #0
0AFA  20260E        JSR  DISPL      ;DISPLAY "EPROM PART NUMBER"
0AFD  2013EA        JSR  CRLOW      ;DO A CR
;
;
0B00  20AEEA        JSR  ADDIN      ;GET, STORE & DISPL. EPROM PART NO.
0B03  AD1CA4        LDA  ADDR
0B06  8503          STA  PROM+1     ;STORE LAST 2 DIGITS OF EPROM PART NO.
0B08  AD1DA4        LDA  ADDR+1
0B0B  8502          STA  PROM       ;STORE 1ST 2 DIGITS OF EPROM PART NO.
0B0D  2013EA        JSR  CRLOW      ;DO A CR
;
;
;
;     ***** CHECK EPROM PART NUMBER THEN ASSION CODE *****
;
;
0B10  A502          LDA  PROM
0B12  C927          CMP  #$27       ;EPROM PART NO.=27XX?
0B14  F022          BEQ  VAL4       ;IF SO
0B16  C925          CMP  #$25       ;EPROM PART NO.=25XX?
0B18  F003          BEQ  VAL1       ;IF SO
0B1A  4C530B        JMP  VAL7
0B1D  A503    VAL1  LDA  PROM+1
0B1F  C916          CMP  #$16       ;EPROM PART NO.=2516?
0B21  F00E          BEQ  VAL3       ;IF SO
0B23  C932          CMP  #$32       ;EPROM PART NO.=2532?
0B25  F003          BEQ  VAL2       ;IF SO
0B27  4C530B        JMP  VAL7
0B2A  A903    VAL2  LDA  #3
0B2C  850C          STA  TYPE       ;TYPE=3 FOR 2532
0B2E  4C5E0B        JMP  CHECK
0B31  A902    VAL3  LDA  #2
0B33  850C          STA  TYPE       ;TYPE=2 FOR 2516
0B35  4C5E0B        JMP  CHECK
```

```
0B38  A503    VAL4  LDA  PROM+1
0B3A  C908          CMP  #8          ;EPROM PART NO.=2708?
0B3C  F00E          BEQ  VAL6        ;IF SO
0B3E  C916          CMP  #$16        ;EPROM PART NO.=2716?
0B40  F003          BEQ  VAL5        ;IF SO
0B42  4C530B        JMP  VAL7
0B45  A901    VAL5  LDA  #1
0B47  850C          STA  TYPE        ;TYPE=1 FOR 2716
0B49  4C5E0B        JMP  CHECK
0B4C  A900    VAL6  LDA  #0
0B4E  850C          STA  TYPE        ;TYPE=0 FOR 2708
0B50  4C5E0B        JMP  CHECK
0B53  A204    VAL7  LDX  #4
0B55  20260E        JSR  DISPL       ;DISPLAY "WRONG PART NUMBER"
0B58  2013EA        JSR  CRLOW       ;DO A CR
0B5B  4CF80A        JMP  REDO        ;TRY IT AGAIN

;
;
;     **** CHK PROGRAMER SWITCH AGAINST EPROM PART NUMBER ****
;
;
0B5E  A50C    CHECK LDA  TYPE        ;GET TYPE OF EPROM
0B60  C900          CMP  #0          ;EPROM=2708?
0B62  D00A          BNE  CK1         ;IF NOT
0B64  AD00A0        LDA  DRB
0B67  2960          AND  #$60        ;PB5=PB6=0?
0B69  F037          BEQ  CK5         ;IF SO
0B6B  4C970B        JMP  CK4         ;IF NOT
0B6E  C901    CK1   CMP  #1          ;EPROM=2716?
0B70  D00C          BNE  CK2         ;IF NOT
0B72  AD00A0        LDA  DRB
0B75  2960          AND  #$60
0B77  C940          CMP  #$40        ;PB5=0 PB6=1?
0B79  F027          BEQ  CK5         ;IF SO
0B7B  4C970B        JMP  CK4         ;IF NOT
0B7E  C902    CK2   CMP  #2          ;EPROM=2516?
0B80  D00C          BNE  CK3         ;IF NOT
0B82  AD00A0        LDA  DRB
0B85  2960          AND  #$60
0B87  C920          CMP  #$20        ;PB5=1 PB6=0?
0B89  F017          BEQ  CK5         ;IF SO
0B8B  4C970B        JMP  CK4         ;IF NOT
0B8E  AD00A0  CK3   LDA  DRB
0B91  2960          AND  #$60
0B93  C960          CMP  #$60        ;PB5=PB6=1?
0B95  F00B          BEQ  CK5         ;IF SO
0B97  A201    CK4   LDX  #1
0B99  20260E        JSR  DISPL       ;DISPLAY "CHK. SWITCH POSITION"
0B9C  2013EA        JSR  CRLOW       ;DO A CR
0B9F  4CF80A        JMP  REDO        ;TRY AGAIN

;
;
;     **** GET AND STORE PROGRAM AND EPROM ADDRESSES *****
;
;
0BA2  A202    CK5   LDX  #2
0BA4  20260E        JSR  DISPL       ;DISPLAY "PROGRAM START ADDRESS"
0BA7  2013EA        JSR  CRLOW       ;DO A CR
0BAA  20AEEA        JSR  ADDIN       ;GET, STORE & DISPL. PROG. START ADDR.
0BAD  AD1CA4        LDA  ADDR
0BB0  8504          STA  PRST        ;STORE PROG. START ADDR. LOW BYTE
0BB2  AD1DA4        LDA  ADDR+1
0BB5  8505          STA  PRST+1      ;STORE PROG. START ADDR. HI BYTE
0BB7  2013EA        JSR  CRLOW       ;DO A CR
0BBA  A203          LDX  #3
0BBC  20260E        JSR  DISPL       ;DISPLAY "EPROM ADDRESS"
0BBF  2013EA        JSR  CRLOW       ;DO A CR
0BC2  20A3E7        JSR  FROM        ;DISPLAY "FROM =" THEN ENTER ADDR.
0BC5  AD1CA4        LDA  ADDR
0BC8  8506          STA  EPST        ;STORE EPROM START ADDR. LOW BYTE
0BCA  AD1DA4        LDA  ADDR+1
0BCD  8507          STA  EPST+1      ;STORE EPROM START ADDR. HI BYTE
0BCF  20A7E7        JSR  TO          ;DISPLAY "TO =" THEN ENTER ADDR.
0BD2  AD1CA4        LDA  ADDR
0BD5  8508          STA  EPND        ;STORE EPROM END ADDR. LOW BYTE
0BD7  AD1DA4        LDA  ADDR+1
0BDA  8509          STA  EPND+1      ;STORE EPROM END ADDR. HI BYTE
0BDC  2013EA        JSR  CRLOW       ;DO A CR

;
;
;          ***** CHECK IF EPROM IS BLANK *****
;
;
0BDF  A205          LDX  #5
0BE1  20260E        JSR  DISPL       ;DISPL. "CHK IF BLANK? Y/N"
0BE4  2038E8        JSR  SPACE       ;ADD TWO SPACES
0BE7  203CE9        JSR  READ        ;READ RESPONSE
0BEA  207AE9        JSR  OUTPT       ;DISPL. RESPONSE
0BED  2013EA        JSR  CRLOW       ;DO A CR
0BF0  C959          CMP  #'Y'        ;RESPONSE=Y?
0BF2  F003          BEQ  BLANK       ;IF SO
0BF4  4C630C        JMP  PROG
0BF7  A9CE    BLANK LDA  #$CE
0BF9  8D0CA0        STA  PCR         ;APPLY POWER
0BFC  A064          LDY  #100
0BFE  20170E        JSR  DELAY       ;WAIT 100MSEC
0C01  A206          LDX  #6
0C03  20260E        JSR  DISPL       ;DISPL. "READ"
0C06  A9CC          LDA  #$CC
0C08  8D0CA0        STA  PCR         ;ENABLE EPROM ADDR. COUNTER
0C0B  20740E        JSR  INCR        ;INCR EPROM ADDR CNTR TO EPROM START ADDR
0C0E  A900          LDA  #0
0C10  8514          STA  FLAG        ;RESET FLAG
0C12  A001    BLNK1 LDY  #1
0C14  20170E        JSR  DELAY       ;WAIT 1MSEC
0C17  AD01A0        LDA  DRA
0C1A  C9FF          CMP  #$FF        ;DOES EPROM BYTE=FF?
0C1C  F018          BEQ  BLNK2       ;IF SO
0C1E  2013EA        JSR  CRLOW       ;DO A CR
0C21  A207          LDX  #7
0C23  20260E        JSR  DISPL       ;DISPL. "NOT BLANK"
0C26  203BE8        JSR  SPACE       ;ADD 2 SPACES
0C29  20A30E        JSR  PUT         ;DISPL. EPROM ADDR. & BYTE
0C2C  A901          LDA  #1
0C2E  8514          STA  FLAG        ;SET FLAG
0C30  20F10E        JSR  CHEK        ;IF ESC IS PUSHED TURN PWR OFF & EXIT
0C33  4C460C        JMP  BLNK3       ;IF NOT
```

12 TARGET SEP/OCT 1980

```
0C36  A514    BLNK2 LDA  FLAG        ;IS FLAG SET?
0C38  F00C          BEQ  BLNK3       ;IF NOT
0C3A  2013EA        JSR  CRLOW       ;DO A CR
0C3D  A206          LDX  #6
0C3F  20260E        JSR  DISPL       ;DISPL. "READ"
0C42  A900          LDA  #0
0C44  8514          STA  FLAG        ;RESET FLAG
0C46  A50D    BLNK3 LDA  COUNT
0C48  C508          CMP  EPND        ;HAVE WE REACHED EPROM END ADDR. LO BYT?
0C4A  D011          BNE  BLNK4       ;IF NOT
0C4C  A50E          LDA  COUNT+1
0C4E  C509          CMP  EPND+1      ;HAVE WE REACHED EPROM END ADDR. HI BYT?
0C50  D00B          BNE  BLNK4       ;IF NOT
0C52  A9CE          LDA  #$CE
0C54  8D0CA0        STA  PCR         ;RESET EPROM ADDR. COUNTER
0C57  2013EA        JSR  CRLOW       ;DO A CR
0C5A  4C630C        JMP  PROG
0C5D  20940E  BLNK4 JSR  BUMP        ;INCR. EPROM ADDR. COUNTER
0C60  4C120C        JMP  BLNK1       ;GO BACK

;
;
;          ***** PROGRAMING THE EPROM *****
;
;
0C63  A208    PROG  LDX  #8
0C65  20260E        JSR  DISPL       ;DISPL. "PROGRAM? Y/N"
0C68  203BE8        JSR  SPACE       ;ADD 2 SPACES
0C6B  203CE9        JSR  READ        ;READ RESPONSE
0C6E  207AE9        JSR  OUTPT       ;DISPL. RESPONSE
0C71  2013EA        JSR  CRLOW       ;DO A CR
0C74  C959          CMP  #'Y'        ;RESPONSE=Y?
0C76  F003          BEQ  PROG1       ;IF SO
0C78  4C730D        JMP  VERFY
0C7B  A50C    PROG1 LDA  TYPE        ;GET CODE FOR TYPE OF EPROM
0C7D  C900          CMP  #0          ;IS EPROM A 2708?
0C7F  D008          BNE  PROG2       ;IF NOT
0C81  A99A          LDA  #$9A
0C83  8D00A0        STA  DRB         ;SET UP TO PROGRAM 2708'S
0C86  4CA60C        JMP  PROG5
0C89  C901    PROG2 CMP  #1          ;IS EPROM A 2716?
0C8B  D008          BNE  PROG3       ;IF NOT
0C8D  A90A          LDA  #$0A
0C8F  8D00A0        STA  DRB         ;SET UP TO PROGRAM 2716'S
0C92  4CA60C        JMP  PROG5
0C95  C902    PROG3 CMP  #2          ;IS EPROM A 2516?
0C97  D008          BNE  PROG4       ;IF NOT
0C99  A902          LDA  #2
0C9B  8D00A0        STA  DRB         ;SET UP TO PROGRAM 2516'S
0C9E  4CA60C        JMP  PROG5
0CA1  A908    PROG4 LDA  #8
0CA3  8D00A0        STA  DRB         ;SET UP TO PROGRAM 2532'S
0CA6  A9CE    PROG5 LDA  #$CE
0CA8  8D0CA0        STA  PCR         ;APPLY POWER
0CAB  A064          LDY  #100
0CAD  20170E        JSR  DELAY       ;WAIT 100MSEC
0CB0  A900          LDA  #0
0CB2  850A          STA  LPCNT       ;START CYCLE COUNT (HEX) AT 0
0CB4  8513          STA  DECL        ;START CYCLE COUNT (DEC) AT 0
0CB6  A9FF          LDA  #$FF
0CB8  8D03A0        STA  DDRA        ;PA0-PA7=OUTPUTS
0CBB  A209          LDX  #9
0CBD  20260E        JSR  DISPL       ;DISPL. "PROGRAM"
0CC0  203BE8        JSR  SPACE       ;ADD 2 SPACES
0CC3  A900          LDA  #0
0CC5  2046EA        JSR  NUMA        ;CONV. 0 TO ASCII THEN DISPL. IT
0CC8  A9CC    PROG6 LDA  #$CC
0CCA  8D0CA0        STA  PCR         ;ENABLE EPROM ADDR. COUNTER
0CCD  20740E        JSR  INCR        ;INCR. EPROM ADDR CNTR TO EPROM START ADDR
0CD0  206B0E        JSR  GET1        ;TEMP. STORE PROG. START ADDR.
0CD3  A001          LDY  #1
0CD5  20170E        JSR  DELAY       ;WAIT 1MSEC
0CD8  A000    PROG7 LDY  #0
0CDA  B115          LDA  (PTEMP),Y
0CDC  8D01A0        STA  DRA         ;PUT PROG. BYTE IN PORT A
0CDF  A001          LDY  #1
0CE1  20170E        JSR  DELAY       ;WAIT 1MSEC
0CE4  A50C          LDA  TYPE        ;GET CODE FOR TYPE OF EPROM
0CE6  C900          CMP  #0          ;IS EPROM A 2708?
0CE8  F007          BEQ  PROG8       ;IF SO
0CEA  C901          CMP  #1          ;IS EPROM A 2716?
0CEC  F003          BEQ  PROG8       ;IF SO
0CEE  4C090D        JMP  PROG9       ;EPROM MUST BE A 25XX

0CF1  AD00A0  PROG8 LDA  DRB         ;GET READY TO TOGGLE PB2
0CF4  4904          EOR  #4
0CF6  8D00A0        STA  DRB         ;TOGGLE PB2 HI
0CF9  A001          LDY  #1
0CFB  20170E        JSR  DELAY       ;WAIT 1MSEC
0CFE  AD00A0        LDA  DRB
0D01  4904          EOR  #4
0D03  8D00A0        STA  DRB         ;TOGGLE PB2 LO
0D06  4C1E0D        JMP  PROGA
0D09  AD00A0  PROG9 LDA  DRB         ;GET READY TO TOGGLE PB1
0D0C  4902          EOR  #2
0D0E  8D00A0        STA  DRB         ;TOGGLE PB1 LO FOR 2516 / HI FOR 2532
0D11  A032          LDY  #50
0D13  20170E        JSR  DELAY       ;WAIT 50MSEC
0D16  AD00A0        LDA  DRB
0D19  4902          EOR  #2
0D1B  8D00A0        STA  DRB         ;TOGGLE PB1 HIGH FOR 2516 / LO FOR 2532
0D1E  A50D    PROGA LDA  COUNT
0D20  C508          CMP  EPND        ;HAVE WE REACHED EPROM END ADDR. LO BYT?
0D22  D039          BNE  PROGB       ;IF NOT
0D24  A50E          LDA  COUNT+1
0D26  C509          CMP  EPND+1      ;HAVE WE REACHED EPROM END ADDR. HI BYT?
0D28  D033          BNE  PROGB       ;IF NOT
0D2A  E60A          INC  LPCNT       ;ADD 1 TO NO. OF PROG. CYCLES
0D2C  20C50E        JSR  HEXDC       ;CONV. NO. FROM HEX TO DEC. & STORE AT DECI
0D2F  CE15A4        DEC  POINT       ;BACK UP DISPLAY POINTER
0D32  CE15A4        DEC  POINT
0D35  CE16A4        DEC  PRINT       ;BACK UP PRINTER POINTER
0D38  CE16A4        DEC  PRINT
0D3B  A513          LDA  DECL        ;GET NO. OF PROG. CYCLES (DEC)
0D3D  2046EA        JSR  NUMA        ;CONV. NO. TO ASCII & DISPL. IT
0D40  A9CE          LDA  #$CE
0D42  8D0CA0        STA  PCR         ;RESET EPROM ADDR. COUNTER
0D45  A50A          LDA  LPCNT
0D47  C963          CMP  #99         ;PROGRAM CYCLES=99?
```

```
0D49  F01B          BEQ  PROGC      ;IF SO
0D4B  A50C          LDA  TYPE       ;GET CODE FOR TYPE OF EPROM
0D4D  C902          CMP  #2         ;IS EPROM A 2516?
0D4F  F015          BEQ  PROGC      ;IF SO
0D51  C903          CMP  #3         ;IS EPROM A 2532?
0D53  F011          BEQ  PROGC      ;IF SO
0D55  A001          LDY  #1         ;EPROM MUST BE A 27XX
0D57  20170E        JSR  DELAY      ;WAIT 1MSEC
0D5A  4CC80C        JMP  PROG6      ;GO BACK AND RE-PROGRAM
0D5D  20B70E  PROGB JSR  UP1        ;ADD 1 TO PROG. ADDR.
0D60  20940E        JSR  BUMP       ;ADD 1 TO EPROM ADDR. & INCR ADDR. CNTR.
0D63  4CD80C        JMP  PROG7      ;GO BACK & PROG. NEXT BYTE
0D66  2013EA  PROGC JSR  CRLOW      ;DO A CR
0D69  A900          LDA  #0
0D6B  8D03A0        STA  DDRA       ;PA0-PA7=INPUTS
0D6E  A98A          LDA  #$8A
0D70  8D00A0        STA  DRB        SET UP TO READ MODE

;
;
;                ***** VERIFY THE EPROM *****
;
;
0D73  A20A    VERFY LDX  #10
0D75  20260E        JSR  DISPL      ;DISPL. "VERIFY? Y/N"
0D78  203BE8        JSR  SPACE      ;SKIP 2 SPACES
0D7B  203CE9        JSR  READ       ;GET RESPONSE
0D7E  207AE9        JSR  OUTPT      ;DISPL. RESPONSE
0D81  2013EA        JSR  CRLOW      ;DO A CR
0D84  C959          CMP  #'Y'       ;RESPONSE=Y?
0D86  F008          BEQ  VER1       ;IF SO
0D88  A9EE          LDA  #$EE
0D8A  8D0CA0        STA  PCR        ;REMOVE PWR.
0D8D  4C82E1        JMP  MON        ;EXIT PROGRAM
0D90  A9CE    VER1  LDA  #$CE
0D92  8D0CA0        STA  PCR        ;APPLY PWR.
0D95  A064          LDY  #100
0D97  20170E        JSR  DELAY      ;WAIT 100MSEC
0D9A  A20B          LDX  #11
0D9C  20260E        JSR  DISPL      ;DISPL. "VERIFY"
0D9F  A9CC          LDA  #$CC
0DA1  8D0CA0        STA  PCR        ;ENABL. EPROM ADDR. COUNTER
0DA4  20740E        JSR  INCR       ;INCR. EPROM ADDR CNTR TO EPROM START ADDRL
0DA7  A900          LDA  #0
0DA9  8514          STA  FLAG       ;RESET FLAG
0DAB  206B0E        JSR  GET1       ;TEMP. STORE PROG. START ADDR.
0DAE  A001    VER2  LDY  #1
0DB0  20170E        JSR  DELAY      ;WAIT 1MSEC
0DB3  A000          LDY  #0
0DB5  B115          LDA  (PTEMP),Y  ;GET PROGRAM BYTE
0DB7  CD01A0        CMP  DRA        ;PROG. BYTE=EPROM BYTE?
0DBA  D035          BNE  VER5       ;IF NOT
0DBC  A514          LDA  FLAG       ;IS FLAG SET?
0DBE  F00C          BEQ  VER3       ;IF NOT
0DC0  2013EA        JSR  CRLOW      ;DO A CR
0DC3  A20B          LDX  #11
0DC5  20260E        JSR  DISPL      ;DISPL. "VERIFY"
0DC8  A900          LDA  #0
0DCA  8514          STA  FLAG       ;RESET FLAG
0DCC  A50D    VER3  LDA  COUNT
0DCE  C508          CMP  EPND       ;HAVE WE REACHED EPROM END ADDR. LO BYT?
0DD0  D016          BNE  VER4       ;IF NOT
0DD2  A50E          LDA  COUNT+1
0DD4  C509          CMP  EPND+1     ;HAVE WE REACHED EPROM END ADDR. HI BYT?
0DD6  D010          BNE  VER4       ;IF NOT
0DD8  2013EA        JSR  CRLOW      ;DO A CR
0DDB  A20D          LDX  #13
0DDD  20260E        JSR  DISPL      ;DISPL. "VERIFY COMPLETE"
0DE0  A9EE          LDA  #$EE
0DE2  8D0CA0        STA  PCR        ;REMOVE PWR.
0DE5  4C82E1        JMP  MON        ;EXIT PROGRAM


0DE8  20B70E  VER4  JSR  UP1        ;ADD 1 TO PROG. ADDR.
0DEB  20940E        JSR  BUMP       ;INCR. EPROM ADDR. COUNTER
0DEE  4CAE0D        JMP  VER2       ;GO VERIFY NEXT BYTE
0DF1  A514    VER5  LDA  FLAG       ;IS FLAG SET?
0DF3  D008          BNE  VER6       ;IF SO
0DF5  2013EA        JSR  CRLOW      ;DO A CR
0DF8  A20C          LDX  #12
0DFA  20260E        JSR  DISPL      ;DISPL. "VERIFY ERROR"
0DFD  2013EA  VER6  JSR  CRLOW      ;DO A CR
0E00  20A30E        JSR  PUT        ;DISPL. EPROM ADDR. AND BYTE
0E03  203BE8        JSR  SPACE      ;SKIP 2 SPACES
0E06  A000          LDY  #0
0E08  B115          LDA  (PTEMP),Y  ;GET PROG. BYTE
0E0A  2046EA        JSR  NUMA       ;CONV. HEX TO ASCII & DISPL
0E0D  A901          LDA  #1
0E0F  8514          STA  FLAG       ;SET FLAG
0E11  20F10E        JSR  CHEK       ;IF ESC IS PUSHED TURN PWR. OFF & EXIT
0E14  4CCC0D        JMP  VER3       ;GO BACK FOR MORE


;
;
;SUBROUTINES
;
;
;                ***** 1MSEC DELAY *****
;
0E17  8400    DELAY STY  DEL1
0E19  A07B    DE1   LDY  #$7B
0E1B  8401          STY  DEL2
0E1D  C601    DE2   DEC  DEL2
0E1F  D0FC          BNE  DE2
0E21  C600          DEC  DEL1
0E23  D0F4          BNE  DE1
0E25  60            RTS
;
;
;                ***** DISPLAY MESSAGE *****
;
;
0E26  BDB80A  DISPL LDA  LNGTH,X    ;GET LENGTH OF MESSAGE
0E29  850F          STA  LNG
0E2B  8A            TXA
0E2C  18            CLC
0E2D  2A            ROL  A          ;MULTIPLY BY 2
0E2E  AA            TAX
0E2F  BDC60A        LDA  ADDS,X
0E32  8510          STA  MESG       ;STORE MESSAGE ADDR. LO BYT
0E34  BDC70A        LDA  ADDS+1,X
0E37  8511          STA  MESG+1     ;STORE MESSAGE ADDR. HI BYT
0E39  A000          LDY  #0
0E3B  B110    DIS1  LDA  (MESG),Y   ;GET MESSAGE CHAR.
0E3D  207AE9        JSR  OUTPT      ;DISPL. CHAR
0E40  C8            INY
0E41  C40F          CPY  LNG        ;ALL OF MESSAGE DISPL?
0E43  D0F6          BNE  DIS1       ;IF NOT
0E45  60            RTS
;
;
;           ***** ADD ONE TO EPROM ADDRESS *****
;
;
0E46  18      ADD   CLC
0E47  A50D          LDA  COUNT
0E49  6901          ADC  #1
0E4B  850D          STA  COUNT
0E4D  A50E          LDA  COUNT+1
0E4F  6900          ADC  #0
0E51  850E          STA  COUNT+1
0E53  60            RTS


;
;
;           ***** SUBTRACT ONE FROM EPROM ADDRESS *****
;
;
0E54  38      SUB   SEC
0E55  A50D          LDA  COUNT
0E57  E901          SBC  #1
0E59  850D          STA  COUNT
0E5B  A50E          LDA  COUNT+1
0E5D  E900          SBC  #0
0E5F  850E          STA  COUNT+1
0E61  60            RTS
;
;
;           ***** TEMP. STORE EPROM START ADDRESS *****
;
;
0E62  A506    GET   LDA  EPST
0E64  850D          STA  COUNT      ;STORE LO BYTE
0E66  A507          LDA  EPST+1
0E68  850E          STA  COUNT+1    ;STORE HI BYTE
0E6A  60            RTS
;
;
;           ***** TEMP. STORE PROGRAM START ADDRESS *****
;
;
0E6B  A504    GET1  LDA  PRST
0E6D  8515          STA  PTEMP      ;STORE LO BYTE
0E6F  A505          LDA  PRST+1
0E71  8516          STA  PTEMP+1    ;STORE HI BYTE
0E73  60            RTS
;
;
;      ***** INCR. EPROM ADDR. COUNTER TO EPROM START ADDR. *****L
;
;
0E74  20620E  INCR  JSR  GET        ;STORE EPROM START ADDR. AT COUNT
0E77  A50D    IN1   LDA  COUNT      ;EPROM START ADDR. LO BYT=0?
0E79  D004          BNE  IN2        ;IF NOT
0E7B  A50E          LDA  COUNT+1    ;EPROM START ADDR. HI BYT=0?
0E7D  F011          BEQ  IN3        ;IF SO
0E7F  20540E  IN2   JSR  SUB        ;SUB. 1 FROM EPROM START ADDR.
0E82  EE00A0        INC  DRB        ;TOGGLE PB0 HI
0E85  A001          LDY  #1
0E87  20170E        JSR  DELAY      ;WAIT 1MSEC
0E8A  CE00A0        DEC  DRB        ;TOGGLE PB0 LO
0E8D  4C770E        JMP  IN1
0E90  20620E  IN3   JSR  GET        ;RE-STORE EPROM START ADDR.
0E93  60            RTS
;
;      ***** ADD ONE TO EPROM ADDR. & INCR. ADDR. COUNTER *****
;
;
0E94  20460E  BUMP  JSR  ADD        ;ADD 1 TO EPROM ADDR.
0E97  EE00A0        INC  DRB        ;TOGGLE PB0 HI
0E9A  A001          LDY  #1
0E9C  20170E        JSR  DELAY      ;WAIT 1MSEC
0E9F  CE00A0        DEC  DRB        ;TOGGLE PB0 LO
0EA2  60            RTS
;
;
;           ***** DISPLAY EPROM ADDR. AND BYTE *****
;
;
0EA3  A50E    PUT   LDA  COUNT+1
0EA5  2046EA        JSR  NUMA       ;CONV. HI BYT ADDR. TO ASCII & DISPL.
0EA8  A50D          LDA  COUNT
0EAA  2046EA        JSR  NUMA       ;CONV. LO BYT ADDR. TO ASCII & DISPL.
0EAD  203BE8        JSR  SPACE      ;SKIP 2 SPACES
0EB0  AD01A0        LDA  DRA
0EB3  2046EA        JSR  NUMA       ;CONV. EPROM BYT TO ASCII & DISPL.
0EB6  60            RTS
;
;
;           ***** ADD ONE TO PROGRAM ADDRESS *****
;
;
0EB7  18      UP1   CLC
0EB8  A515          LDA  PTEMP
0EBA  6901          ADC  #1
0EBC  8515          STA  PTEMP
0EBE  A516          LDA  PTEMP+1
0EC0  6900          ADC  #0
0EC2  8516          STA  PTEMP+1
0EC4  60            RTS
```

SEP/OCT 1980 TARGET 13

```
;
;
;       ***** CONVERT HEX TO DECIMAL AND STORE AT DECL *****
;
;
0EC5  A900     HEXDC LDA  #0
0EC7  8513           STA  DECL
0EC9  A50A           LDA  LPCNT      ;GET NO. OF PROG. CYCLES (HEX)
0ECB  8512           STA  HEX
0ECD  A201           LDX  #1
0ECF  A512     HEX1  LDA  HEX        ;GET NO. OF PROG. CYCLES (HEX)
0ED1  290F           AND  #$0F       ;LOW BYTE=0?
0ED3  F00D           BEQ  HEX3       ;IF SO
0ED5  A8             TAY             ;Y REG.=NO. OF PROG. CYCLES (LO BYT)
0ED6  18       HEX2  CLC
0ED7  F8             SED
0ED8  A513           LDA  DECL       ;GET DEC. NO.
0EDA  7DE20A         ADC  TAB,X      ;ADD 1 OR 16
0EDD  8513           STA  DECL       ;STORE DEC. NO.
0EDF  88             DEY             ;LO BYT=0?
0EE0  D0F4           BNE  HEX2       ;IF NOT
0EE2  CA       HEX3  DEX
0EE3  300A           BMI  HEX5       ;BRNCH. IF X<0
0EE5  A004           LDY  #4         ;X MUST BE=0
0EE7  4612     HEX4  LSR  HEX        ;DIVIDE BY 2
0EE9  88             DEY             ;DIVIDED BY 16?
0EEA  D0FB           BNE  HEX4       ;IF NOT
0EEC  4CCF0E         JMP  HEX1       ;GO BACK AND ADD 16
0EEF  D8       HEX5  CLD
0EF0  60             RTS
;
;
;           ***** CHECK TO SEE IF ESC KEY IS PUSHED *****
;
;
0EF1  A9FB     CHEK  LDA  #$FB
0EF3  8D80A4         STA  DRA2       ;SEND 0 TO KB ON PA2
0EF6  AD82A4         LDA  DRB2
0EF9  2980           AND  #$80       ;DOES KB PB7=0?
0EFB  D008           BNE  EXIT       ;IF NOT
0EFD  A9EE           LDA  #$EE
0EFF  8D0CA0         STA  PCR        ;TURN PWR. OFF
0F02  4C82E1         JMP  MON        ;EXIT PROGRAM
0F05  60       EXIT  RTS
;
;
                     .EN
00 ERRORS
```

```
SYMBOL TABLE
ADD     0E46  01
ADDIN   EAAE  02
ADDR    A41C  08
ADDS    0AC6  02
BLANK   0BF7  01
BLNK1   0C12  01
BLNK2   0C36  01
BLNK3   0C46  02
BLNK4   0C5D  02
BUMP    0E94  03
CHECK   0B5E  04
CHEK    0EF1  02
CK1     0B6E  01
CK2     0B7E  01
CK3     0B8E  01
CK4     0B97  03
CK5     0BA2  04
COUNT   000D  20
CRLOW   EA13  19
DDRA    A003  03
DDRB    A002  01
DE1     0E19  01
DE2     0E1D  01
DECL    0013  05
DEL1    0000  02
DEL2    0001  02
DELAY   0E17  12
DIS1    0E3B  01
DISPL   0E26  16
DRA     A001  04
DRA2    A480  01
DRB     A000  22
DRB2    A482  01
EPND    0008  08
EPST    0006  04
EXIT    0F05  01
FLAG    0014  09
FROM    E7A3  01
GET     0E62  02
GET1    0E6B  02
HEX     0012  03
HEX1    0ECF  01
HEX2    0ED6  01
HEX3    0EE2  01
HEX4    0EE7  01
HEX5    0EEF  01
HEXDC   0EC5  01
IN1     0E77  01
IN2     0E7F  01
IN3     0E90  01
INCR    0E74  03
LNG     000F  02
LNGTH   0A88  01
LPCNT   000A  04
MESG    0010  03
MESS1   0A03  01
MESS2   0A14  01
MESS3   0A28  01
MESS4   0A3D  01
MESS5   0A4A  01
MESS6   0A5B  01
MESS7   0A6C  01
MESS8   0A70  01
MESS9   0A79  01
MESSA   0A85  01
MESSB   0A8C  01
MESSC   0A97  01
MESSD   0A9D  01
MESSE   0AA9  01
MON     E182  03
NUMA    EA46  06
OUTPT   E97A  04
PCR     A00C  12
POINT   A415  02
PRINT   A416  02
PROG    0C63  02
PROG1   0C7B  01
PROG2   0C89  01
PROG3   0C95  01
PROG4   0CA1  01
PROG5   0CA6  03
PROG6   0CC8  01
PROG7   0CD8  01
PROG8   0CF1  02
PROG9   0D09  01
PROGA   0D1E  01
PROGB   0D5D  02
PROGC   0D66  03
PROM    0002  05
PRST    0004  04
PTEMP   0015  09
PUT     0EA3  02
READ    E93C  03
REDO    0AF8  02
SPACE   E83B  07
START   0AE4  01
SUB     0E54  01
TAB     0AE2  01
TO      E7A7  01
TYPE    000C  08
UP1     0EB7  02
VAL1    0B1D  01
VAL2    0B2A  01
VAL3    0B31  01
VAL4    0B38  01
VAL5    0B45  01
VAL6    0B4C  01
VAL7    0B53  03
VER1    0D90  01
VER2    0DAE  01
VER3    0DCC  02
VER4    0DE8  02
VER5    0DF1  01
VER6    0DFD  01
VERFY   0D73  01
:
<
```

New Product

NEW PRODUCT

A version of the FORTH programming system is now available for the 6502 based KIM-1, SYM-1, and AIM 65 microcomputer systems. A version for the APPLE II will be forthcoming.

FORTH is a high-level vocabulary based language which is structured and extendable. It's finding its way to increased usage in microcomputer applications such as graphics, robotics and process control and telecommunications due to its memory efficiency, fast operating speed (when compared to most other high-level languages) and ease of interfacing to assembly language.

But, perhaps the greatest advantage of using FORTH is that everytime you write a program, a new application language is created because FORTH actually BECOMES the application! A typical FORTH application language/program could consist of words like DELAY, READ-A/D, TESTVALVE, TURNON and SCREENCLEAR.

Non-programming engineers can be taught to write programs in this new applications language in less time than it would take to teach them a conventional high-level language, like BASIC.

This particular version of FORTH contains a built-in 6502 assembler (this enables the user to "drop-in" to assembly language at any time), a text editor (to manipulate FORTH source programs), and a cassette file management system. Information on interfacing FORTH to a floppy disk is provided as well as several extensions to the language.

6502 FORTH sells for $90.00 (plus $4.00 for S&H) and includes a user manual, a well-commented source listing, and a cassette containing the object code.

For ordering or additional information, contact: Eric C. Rehnke, Tech Services, 1067 Jadestone Lane, Corona CA 91720.

```
>?
<D>EPROM PART NUMBER ◄── Start Program here
=2708
PROGRAM START ADDRESS
=0200
EPROM ADDRESS
FROM=000 TO=00F
CHK IF BLANK? Y/N  Y
READ
PROGRAM? Y/N  Y                          Program Steps
                                         ↓
PROGRAM  000102030405060708091011121314151617181920212223242526272829303132333435
536373839404142434445464748495051525354555657585960616263646566676869707172737475
767778798081828384858687888990919293949596979899
VERIFY? Y/N  Y
VERIFY
VERIFY COMPLETE
<>?
<>?  ◄──────────────────── End program
<>?
<M>=0200 41 44 44 20
< > 0204 20 46 84 01
< > 0208 41 44 44 49
< > 020C 4E AE EA 02      Data that was programmed
< > 0210 41 44 44 52
<>?
<>?
<D>EPROM PART NUMBER ◄── Re-entered program
=2708
PROGRAM START ADDRESS
=0200
EPROM ADDRESS
FROM=000 TO=012
CHK IF BLANK? Y/N  Y
READ
NOT BLANK  0000  41
NOT BLANK  0001  44
NOT BLANK  0002  44
NOT BLANK  0003  20
NOT BLANK  0004  20
NOT BLANK  0005  46      Data in EPROM
NOT BLANK  0006  84
NOT BLANK  0007  01
NOT BLANK  0008  41
NOT BLANK  0009  44
NOT BLANK  000A  44
NOT BLANK  000B  49
NOT BLANK  000C  4E
NOT BLANK  000D  AE
NOT BLANK  000E  EA
NOT BLANK  000F  02
READ
PROGRAM? Y/N  N
VERIFY? Y/N  Y
VERIFY
VERIFY ERROR
0010  FF  41
0011  FF  44      These addresses were not programmed
0012  FF  44
VERIFY COMPLETE
<
```

GENERAL INFORMATION

Article contributions are always welcome. Program listings may or may not be retyped. When submitting information on AIM thermal paper adjust the darkness control to its darkest setting. Artwork will not be redrawn so please submit your best work. Artwork may be oversize if necessary and will be reduced to proper size.

Text should accompany articles to explain what is being done, how it is done, and how it may be modified to suit the user.

Please submit a self addressed stamped envelope for any replies that you desire.

Back Issues- A consolidated 1979 issue is available for $6.00 ($12). In addition 1980 issues are available beginning with the January/ February and at subsequent two month intervals. Individual 1980 issues are $1.00 (US and CAN, $2.00 elsewhere).

Time to Renew- The mailing label contains the last issue that you will receive. If no date appears you have at least two issues left on you subscription.

Target- an AIM 65 newsletter is published bimonthly with an annual subscription rate of $6.00 in the US and Canada, $12.00 elsewhere. Contact Donald Clem, RR# 2, Spencerville,OH 45887.

THE TARGET
c/o DONALD CLEM
R.R. #2, CONANT RD.
SPENCERVILLE, OHIO 45887