THE TARGET

March/April 1980 -- an AIM65 newsletter

HERE, THERE and EVERYWHERE

Users of the Aim 65 are continuing to snowball support. One area is Rockwell's soon to be released newsletter for the Aim 65. The newsletter will cost only $5.00 and provide low cost support. Target will continue as an independent newsletter so Aim users should really be happy. In addition a new magazine is being formed to support Single Board Computers based on the 6502. The first issue of Compute II will be available shortly.

Hudson Digital Electronics and RNB Enterprises have announced floppy disk interfaces for the Aim. RNB is 8 inch while HDE may be either 8 or 5¼ inch. Cyberdyne has cassette software (see this issue). Hopefully this is just the tip of the iceberg to come.

Scitronics Inc and Connecticut Micro-Computer have or will have shortly controllers that interface with the AC control modules sold by Radio Shack, etc. These controllers and the remote modules combined with your computer allow remote operation of lights, appliances or whatever is connected to normal house wiring.

No discrete wiring is required between the computer and the remote location. All communication is over the house wiring. Even if you move the remote module it will still respond in its new location. One remote module (lamp) will turn a light off or on and may be used to control the brightness. Since these modules require a minimum of wiring a user can eliminate costly house rewiring and if you move they can go with you. This idea of remote control is not new but to have the remotes available through Radio Shack or Sears is.

GRAPEVINE

The grapevine has some changes to zero page usage in Basic. Ø6= A delimit character, Ø7=another delimiting char, Ø8=a general counter, ØF=for determining sign of tangent, 11=position of terminal carriage, 96=pointer used in function selection, 98=pointer to a string descriptor.

Pyramid Data Systems has a cassette based operating system in the works.

Thanks grapevine!!

Ron Riley has labels for Short Cut so if you would like to have a set, send a business size SASE and $1.00 to Ron.

Have you had any problems with your Aim that you were able to fix? Why not share your answers with the rest of us! Along the same lines, would you like to be a Good Guy and offer your services to others? Just send your name and your field of expertise and we'll put out the word.

Short Cut-AUTONumber

Steve Sliber
5815 Southminster
Houston, TX 77035

This modified version of Don Clem's original Short Cut program provides for automatic line numbering in addition to automatic typing of the longer Basic statements. The first table at $1DD0 to $1E5D is the ASCII strings for the various Basic statements. The main difference is the addition of a five byte line number string at M1 ($1E0E) following the carriage return.

Table two at $1E5E to $1E77 is just the vectors to table one, in order to be indexed by the ASCII control codes $01-$1B.

The main program at $1E78 to $1EF0 operates in a manner essentially identical to the original program. The only signicant difference is that after a control Z a second character is input and decoded. Control O toggles the auto-line number ON/OFF, control R resets the line number and increment, and on my system control L is echoed to provide a form feed to the line printer. Any other input results in a control Z being sent back to clear the user input function. Additional code could be added here to provide additional functions that may be desired to customize the system further.

At $1EF1 to $1F17 is the toggle routine. The autonumber function is disabled at two points: a zero is stored at M1 to inhibit the printing of line numbers, and the ONFLG is reset to inhibit incrementing line numbers.

The reset routine at $1F18 to $1F7F uses the GET routine at $1F51 to input a new starting line number and increment. Up to five ASCII characters may be input and right justified. No testing is done to verify the legality of the input. A space or carriage return will terminate the input and cause the input data to be right justified.


```
PASS 1

PASS 2

--------------------
SHORT CUT

==0000
;THIS IS A MODIFICATION OF DON CLEM'S SHORT CUT
;PROGRAM THAT PROVIDES FOR AUTO LINE NUMBERING
;MODIFIED BY STEVE SILBER, HOUSTON, TEXAS
;  JANUARY 1, 1980
;
;CONTROL Z(↑Z) IS USED AS A PREFIX FOR
;    CONTROL FUNCTIONS:
;     ↑Z-↑O -TOGGLE AUTO LINE NUMBER ON/OFF
;      -↑R -RESET FIRST NUMBER & INCREMENT
;      -↑Z -EXIT SHORT CUT
;      -ANY OTHER IS ECHOED
;
--------------------

--------------------
MONITOR EQUATES

==0000 ON=$E6FA
==0000 OFF=$E6F1
==0000 GETCH=$FE83
==0000 OUTPUT=$E97A
==0000 RDRUB=$E95F
==0000 CLR=$EB44
--------------------

;
;USER INPUT LINKAGE
;
==0000
        *=$100
==0100
781E  .  .WORD INPUT
;
;FIRST THE DATA TABLES
;
==010A
        *=$1DD0
==1DD0 A1
4153    .BYTE 'ASC(',00
00
==1DD5 B
52554E  .BYTE 'RUN',00
00
==1DD9 C
434F    .BYTE 'CONT',00
00
==1DDE D
4441    .BYTE 'DATA',00
00
==1DE3 E


454E44  .BYTE 'END',00
00
==1DE? F
464F52  .BYTE 'FOR',00
00
==1DEB G
474F    .BYTE 'GOTO',00
00
==1DF0 H
4C45    .BYTE 'LEFT$(',00
00
==1DF7 I
494E    .BYTE 'INPUT',00
00
==1DFD J
5245    .BYTE 'READ',00
00
==1E02 K
4348    .BYTE 'CHR$(',00
00
==1E08 L
4C49    .BYTE 'LIST',00
00
==1E0D M
0D      .BYTE $0D
==1E0E M1
3030    .BYTE '00100',00
00
==1E14 N
4E45    .BYTE 'NEXT',00
00
==1E19 O
504F    .BYTE 'POKE',00
00
==1E1E P1
5045    .BYTE 'PEEK(',00
00
==1E24 Q
5249    .BYTE 'RIGHT$(',00
00
==1E2C R
5245    .BYTE 'RETURN',00
00
==1E33 S1
5354    .BYTE 'STR$(',00
00
==1E39 T
5448    .BYTE 'THEN',00
00
==1E3E U
5553    .BYTE 'USR(',00
00
==1E43 V
5641    .BYTE 'VAL(',00
00
==1E48 W
4D49    .BYTE 'MID$(',00
00
==1E4E X1
5245    .BYTE 'RESTORE',00
```

```
00
==1E5B  Y1
474F    .BYTE 'GOSUB',00
00
==1E5C  Z
1A      .BYTE $1A,00
00
;
;
==1E5E  TAB2
00      .BYTE 00
05      .BYTE B-A1,C-A1,D-A1,E-A1,F-A1,G-A1,H-A1,I-A1,J-A1
09
0E
13
17
18
20
27
2D
32      .BYTE K-A1,L-A1,M-A1,N-A1,O-A1,P1-A1,Q-A1,R-A1,S1-A1
38
3D
44
49
4E
==1E6E
54
5C
63
69      .BYTE T-A1,U-A1,V-A1,W-A1,X1-A1,Y1-A1,Z-A1
6E
73
78
7E
86
8C
```

CONTINUED

```
0000 48 PHA
0001 A9 LDA #40
0003 8D STA A00B
0006 A9 LDA #00
0008 8D STA A004
000B A9 LDA #F0
000D 8D STA A005
0010 2C BIT A00D
0013 50 BVC 0010
0015 68 PLA
0016 4C JMP EF05
0019 EA NOP
001A 48 PHA
001B A9 LDA #00
001D 8D STA A406
0020 8D STA A407
0023 68 PLA
0024 60 RTS
0025 A9 LDA #05
0027 8D STA A406
002A A9 LDA #EF
002C 8D STA A407
002F 60 RTS

(K)*=10C
/03
 010C 4C JMP 001A
 010F 4C JMP 0025
 0112 4C JMP 0025
```

Slow Display

SLOW

Steve Bresson
1302 Strawberry Lane
Hanover, MD 21076

This program slows down the rate at which characters are displayed on the Aim-65 display. This is done through the display link vector. It is changed so that a new subroutine is executed each time a character is output to the display.

This subroutine produces a delay before jumping to $EF05, which is the default value in DILINK. This subroutine uses the T1 timer in the user VIA. It does not affect the printer operation, and is not itself affected by a reset.

The F1 key is used to change the DILINK so that "SLOW is used. F2 changes it back to the default value.

The subroutines are located at $0000-002F. To see it in action, load it. Then disassemble memory from $F000. Goes sort of fast, right? Now hit F1 and again disassemble memory. The change is noticeable. The amount of delay is determined by the data loaded at $06 and $0B.

This may look trivial at first, but wait till you are trying to trace a long program that doesn't work, and you don't want to use the printer to slow the display down.

Note: Lest you find that I am wasteful, because the VIA is set up each time through, try moving the setup portion over to the F1 subroutine, so that it is only set up once. Put a load $A004 to clear the interrupt each time through. Works fine!! Now hit reset.

```
MAIN PROGRAM

==1E78 INPUT
98      TYA
48      PHA
B003    BCS GET
68      PLA
;NO INITIALIZATION
A8      TAY
60      RTS
;
;FIRST TEST IF DONE WITH LAST INPUT
==1E7F GET
ADB01F LDA NEXT
F01C    BEQ IN
ACAB1F LDY HLDY
C8      INY
==1E88 OUTLP
C8      INY
B9D01D LDA A1,Y

8DB01F STA NEXT
88      DEY
8CAB1F STY HLDY
B9D01D LDA A1,Y
==1E96 OUT
8DAC1F STA HOLDA
68      PLA
A8      TAY
ADAC1F LDA HOLDA
38      SEC
60      RTS
;
;
;INPUT NEW DATA
==1EA0 IN
ADAE1F LDA ENFLG
2DAF1F AND ONFLG
C90D    CMP #$0D
D01A    BNE IN1
;
;IF END FLAG(ENFLG) AND ON FLAG (ONFLG) ARE SET, THEN
;INCREMENT LINE NUMBER
A900    LDA #0
8DAE1F STA ENFLG
;RESET FLG
A004    LDY #4
==1EB1
18      CLC
==1EB2 ADDLP
B9A51F LDA INCR,Y
790E1E ADC M1,Y
C93A    CMP #$3A
3002    BCC SKIP
E90A    SBC #10
==1EBE SKIP
990E1E STA M1,Y
88      DEY
10EE    BPL ADDLP
==1EC4 IN1
2083FE JSR GETCH
C91B    CMP #$1B
B0CB    BCS OUT
C90D    CMP #$0D
D005    BNE ELSE
8DAE1F STA ENFLG
F015    BEQ IN2
==1ED4
;
;IF <CR>THEN SET ENFLG
;IF ↑Z THEN TEST IF CONTROL FUNCTION
;
==1ED4 ELSE
C91A    CMP #$1A
D011    BNE IN2
2083FE JSR GETCH
C90F    CMP #$0F
F012    BEQ TOGGLE
C912    CMP #$12
F035    BEQ RESET
C90C    CMP #$0C
```

```
==1EE5
F0AF    BEQ OUT
A91A    LDA #$1A
==1EE9 IN2
A8      TAY
B95D1E LDA TAB2-1,Y
A8      TAY
4C881E JMP OUTLP
;
;END OF MAIN LOOP
;EXPANSION POSSIBLE BEFORE IN2 TO DECODE OTHER CNTL FCNS
;
;
==1EF1 TOGGLE
AD0E1E LDA M1
F010    BEQ ON1
8DAD1F STA FIRST
;SAVE FIRST DIGIT AND REPLACE W/0
20F1E6 JSR OFF
A900    LDA #0
8DAF1F STA ONFLG
==1F01
8D0E1E STA M1
F090    BEQ OUT
;
;TURN BACK ON
;
==1F06 ON1
ADAD1F LDA FIRST
8D0E1E STA M1
A90D    LDA #$0D
8DAF1F STA ONFLG
20FAE6 JSR ON
A900    LDA #0
==1F16
F0EC    BEQ ON1-2
;
;RESET LINE # & INCREMENT
;
==1F18 RESET
8EAA1F STX HLDX
A200    LDX #0
20001F JSR MPRINT
20511F JSR GET2
A004    LDY #4
==1F25 MOVLP
B9A51F LDA INCR,Y
990E1E STA M1,Y
88      DEY
10F7    BPL MOVLP
A20B    LDX #MES2-MES1
20001F JSR MPRINT
20511F JSR GET2
==1F36
A004    LDY #4
==1F38 DECLP
38      SEC
B9A51F LDA INCR,Y
E930    SBC #$30
99A51F STA INCR,Y
```

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. This is the only method that is used to determine if your subscription is running out.

The Target- an Aim 65 newsletter is published bimonthly with an annual subscription rate of $5.00 in the US and CAN. $12.00 elsewhere(US Funds). First Class and Air Mail respectively. Contact Donald Clem RR#2 Spencerville, OH 45887.

```
88      DEY
10F4    BPL DECLP
ADAA1F  LDA HLDX
A90D    LDA #$0D
==1F49
8DAF1F  STA ONFLG
A900    LDA #0
4C961E  JMP OUT
;
;GET2 INPUTS UP TO 5 CHAR. INTO INCR BUFFER
;AND ALLOWS FOR DEL.  IT WILL RIGHT JUSTIFY LESS THAN 5 CHAR
;
==1F51  GET2
A000    LDY #0
==1F53  GETLP
205FE9  JSR RDRUB
C90D    CMP #$0D
F00E    BEQ EXIT
C920    CMP #$20
F00A    BEQ EXIT
99A51F  STA INCR,Y
C8      INY
C005    CPY #05
==1F64
D0ED    BNE GETLP
F015    BEQ CLEAR
==1F68  EXIT
88      DEY
A204    LDX #4
==1F6B  SHFLP
B9A51F  LDA INCR,Y
9DA51F  STA INCR,X
CA      DEX
88      DEY
10F6    BPL SHFLP
A930    LDA #$30
==1F77  FILLP
9DA51F  STA INCR,X
CA      DEX
10FA    BPL FILLP
==1F7D  CLEAR
4C44EB  JMP CLR
;
;
;SIMPLE MESSAGE PRINTER
==1F80  MPRINT
BD901F  LDA MES1,X
3006    BMI EX
207AE9  JSR OUTPUT
E8      INX
D0F5    BNE MPRINT
==1F8B  EX
297F    AND #$7F
4C7AE9  JMP OUTPUT
;
==1F90  MES1
4649    .BYTE 'FIRST LINE',$A0
A0
==1F9B  MES2
494E    .BYTE 'INCREMENT',$A0

A0
==1FA5  INCR
00      .BYTE 0,0,0,1,0
00
00
01
00
==1FAA  HLDX
        *=*+1
==1FAB  HLDY
        *=*+1
==1FAC  HOLDA
        *=*+1
==1FAD  FIRST
30      .BYTE $30
==1FAE  ENFLG
00      .BYTE 0
==1FAF  ONFLG
0D      .BYTE $0D
==1FB0  NEXT
00      .BYTE 0
        .END
 ERRORS= 0000

00035
```

6502 BOOKS

If you are in need of a book on programming the 6502 here is a partial listing.

6502 Assembly Language Programming
Osborne/McGraw-Hill $12.50
6502 Games
Sybex $12.95
6502 Programming and Interfacing the,
Sam's $13.95
6502 Software Design
Sam's $10.50
6502 Programming the,
Sybex $12.95
6502 Applications Book
Sybex $12.95
Microprocessor Systems Engineering
Matrix $16.00
6502 Software Cookbook
Scelbi $10.95

Shipping and handling are approximately $1.50 (UPS) for domestic and $4.00 for foreign.

Osborne/McGraw-Hill
630 Bancroft Way
Berkeley,CA 94710

Sybex
2344 Sixth St.
Berkeley, CA 94710

Howard W. Sams & Co,Inc
4300 West 62nd St.
PO Box 7092
Indianapolis, IND 46206

Matrix Publishers, Inc
207 Kenyon Rd.
Champaign, IL 61820

Scelbi Publications
20 Hurlbut St.
Elmwood, CT 06110

Assembler

Michael R. Corder
Vice President
Compas Microsystems
224 S.E. 16th Street
Ames, IA 50010

In the Jan/Feb 1980 issue a reader was interested in a comparison of the AIM and ARESCO assemblers. Since our company was indirectly responsible for both these products, I thought I would respond.

We started working with MOS/Technology as a software support group in Jan. 1975 when the 6502 was still a dream. One of the products we produced for them was the software for their MDT 650 developement system. This was a very fine, two processor system which never saw wide acceptance due to financial problems at MOS/Technology which are commonly known. Incidentally, the MDT hardware was produced by COM LOG in Phoenix. The MDT assembler served as the basis for the ARESCO product.

When it became evident that the MDT was going nowhere, we started developement of a simpler system. Rockwell then became another source of the 6500 and was looking for a developement system. We eventually licensed them to build our system which they sell under the name SYSTEM 65. The SYSTEM 65 features an improved version of the MDT assembler. The SYSTEM 65 software was used as the basis as the AIM. Users who have had a chance to use both the AIM and SYSTEM 65 will note that they are identical from a command standpoint. The SYSTEM 65 assembler requires about 5.25K of memory. ROCKWELL took this product and squeezed it into 4K so it would fit into a single ROM. In doing so, they had to reduce its functionality as well as alter its output format to fit the 20 column AIM printer. The other major differences are that most of the .OPT options no longer exist and the sorted symbol table is not printed. Hence, the major difference most users will notice is in the listing format.

To continue our history lesson one step further. Our firm took the basic mini-floppy disk controller design and software from the SYSTEM 65 and turned it into the DAIM disk system for the AIM. We also took the assembler and turned it into A/65.

A/65 is designed to be loaded from disk into RAM and has all the features of the SYSTEM 65 assembler. Thus it is designed

Kim-4

George Sellers
1033 Bishop Walsh Rd
Cumberland, MD 21502

I have just finished adding a Kim-4 motherboard and 8K of RAM from Hudson Digital Electronics, Inc. (HDE-DM816-M8). I bought my Kim-4 and the 8K memory board from Falk-Baker Associates. One week delivery from the time I mailed my order.

The Kim-4 has to be modified somewhat and the Rockwell document No R6500N11 which covers this interfacing is incorrect in several spots.

The corrections which were found by Vern Wolodkin are: on page 1 figure 2 add "cut these two traces" to two traces at top between U7 and U8. On page 2 in paragraph 1 add "cut the two traces on the top end between U8 and U7". These traces connect; U7-1 to Expansion Module Connector Pin J(AB6) and U7-4 to Expansion Module Connector Pin H (AB5).

In paragraph "2." reverse the position of L and M. Also jumper U8-5 to Expansion Connector H and U8-3 to Expansion Con. J.

To connect Tape Control Lines;

1. Cut trace (BRDY) coming from Pin E of Kim-4 Application connector.
2. Jumper;

Aim Appl	Kim-4 Appl
E	E
F	F
H	H
J	J

On page 3 in the Signal Table the following signals are redefined under Kim-4 Application Conn. (column 4).

17-21 are now PB6,CB1,CB2,CA1,CA2
22 is not used
E -J are now Tape 1B rtn, tape 1B, Tape 2B rtn, tape 2B
K the last is not connected.

---

to be used with a terminal and give a full width listing and offers the other features described.

There is one more SYSTEM 65 product which has converted to the AIM that may be of interest to your readers. Our group offers a high level language for the SYSTEM 65 (or PDP-11) that is sold by us as CSL/65 and by ROCKWELL as PL/65. We have developed a subset of this language that will shortly be available from ROCKWELL as a two chip ROM set that plugs into the locations occupied by BASIC ($B000 to $CFFF).

An AIM 65 Scanning Subroutine—SCAN by Jim Clark

The AIM 65 has some very useful programs within its Monitor for reading the keyboard. However, this reading process requires your program to stop and wait for you to press a key. You may prefer to just "scan" the keyboard. The scanning process checks to see if a key is pressed and if it is, then gets the key and responds to it. If, however, no key is pressed, the scanning process is terminated. This scanning process is given in the program below as the SCAN Subroutine. SCAN makes use of the Monitor Subroutine ONEKEY ($EDØ5). ONEKEY checks for a key depression. If a key is depressed, the Y Register will return with the row value (1-8). If no key was depressed, the Y Register will contain a 0. Upon returning from ONEKEY, the Y Register is decremented. If no key was depressed, the contents of the Y Register will be negative and execution will return to the calling program. If a key was depressed, the contents of the Y Register will be a 0 or plus (both considered positive) and the key pressed will be decoded by Monitor Subroutine GETK2 ($ED2C). Actually, the first line in GETK2 is skipped since it is a debounce delay. In the SCAN Subroutine shown in the listings at the right only the A, B, and ESC keys are checked. If A is pressed, pseudo Program A is executed (CRLF); if B was depressed Pseudo Program B; and if ESC is pressed the program uses the Q command from the Editor STOP Subroutine ($F870) to neatly return to the Monitor Program. If none of the keys are pressed, execution returns to the calling pro-gram. A sample calling program is included so something interesting is happening while SCAN is looking for a key closure. The calling program (CPROG) outputs the contents of TIMER 1 LOW (A004) as well as going to the SCAN Subroutine. In the example program the only keys that will cause the scanning process to hang up and execute improperly are the PRINT and the LF keys. You may wish to change the keys which are checked by SCAN or expand to additional keys. Both of these changes may be done quite simply and easily by changing or expanding the instructions shown in the program listings. To run the SCAN Program as shown, load the program as shown in the disassembly (K) listing or, if you are using the Assembler ROM, the source listing provided. The SCAN Subroutine begins at $0200; the Calling Program (CPROG) at $0250; and the called Program A at $0227 and Program B at $022B. The F1 Users Key (010C) is used to start the execution of the calling program.

This program is one which I've found very handy in many applications. It can be used in many game programs which require a specific key closure as a player response to some portion of the game. Its applications in the many industrial uses of the AIM 65 are too numerous to mention. I hope that you find many uses for it and that it will save you programming time and solve some problems for you. I would appreciate hearing some of the uses which you may find for it.

```
(K)*=10C
/01
010C 4C JMP 0250
(K)*=200
/24
0200 48 PHA
0201 20 JSR EB9E
0204 20 JSR ED05
0207 88 DEY
0208 30 BMI 021B
020A A2 LDX #00
020C 20 JSR EC85
020F C9 CMP #41
0211 F0 BEQ 0227
0213 C9 CMP #42
0215 F0 BEQ 022B
0217 C9 CMP #1B
0219 F0 BEQ 0220
021B 20 JSR EBAC
021E 68 PLA
021F 60 RTS
0220 20 JSR EBAC
0223 68 PLA
0224 4C JMP F870
0227 20 JSR E9F0
022A 00 BRK
022B 20 JSR E9F0
022E 00 BRK
022F EA NOP

(K)*=250
/04
0250 20 JSR 0200
0253 AD LDA A004
0256 20 JSR E97A
0259 4C JMP 0250
```

```
PHXY=$EB9E
PLXY=$EBAC
ONEKEY=$ED05
GETK2=$EC82
ESC=$1B
STOP=$F870
CRLF=$E9F0
UT1L=$A004
OUTPUT=$E97A
;
*=$200
SCAN PHA
JSR PHXY
JSR ONEKEY ;KEY?
DEY ; IF MINUS, NO
BMI RETURN
LDX #0
JSR GETK2+3
CMP #'A
BEQ PROGA
CMP #'B
BEQ PROGB
CMP #ESC
BEQ LEAVE
RETURN JSR PLXY
PLA
RTS ; TO CALLING PROG
LEAVE JSR PLXY
PLA
JMP STOP ; TO MONITOR
PROGA JSR CRLF
BRK
PROGB JSR CRLF
BRK
NOP
;
;CALLING PROGRAM
;
*=$10C
JMP CPROG ; F1 KEY
*=$250
CPROG JSR SCAN ; CHK
LDA UT1L
JSR OUTPUT
JMP CPROG
```

EXTERMINATING SOME INVISIBLE BUGS (They can crawl down the power lines). by Gary Peterson

A perfectly good program bombs out; data stored in memory changes quite unexpectedly; one bit of a memory, buffer, or select chip quits working. Hours are spent trying to determine why. All of these problems and more can many times be traced back to high voltage "glitches" being passed thru our power supplies. There are usually many other devices attached to the same power line as our power supply. Many of these can and do produce high voltage "spikes" which are not fully filtered out by our power supply circuits. This is especially true of the hobbyist or non-industrial power supplies to which most of the AIM's I've seen are attached. After experiencing a costly bout of 2114 RAM failures I decided to solve this problem. Using a high speed oscilloscope to watch these "transients" zip right thru my supplies, I applied some new solid state "bug traps" and got rid of them once and forever. The solution was easy and inexpensive. You will need to get a GE-MOV Varistor, Model V130LA10, from your local General Electric components dealer, for each power supply. This two lead device, which looks like a large red disc capacitor, should be soldered directly across the 117 volt power input line where it enters your power supply, right inside the case. If you are using 220 volt power (overseas) use a model V250LA20. At this same point should be connected, in parallel, a 0.001 mfd. 1500 volt disc ceramic capacitor. If your supply has a fuse in the input circuit, then the above connections sould be made so that the fuse would blow if the Varistor should short circuit. The next step involves a tiny but very potent "glitch eater." One of these devices, in parallel with a 0.001 mfd. 1000 volt disc ceramic cap., should be soldered directly across the output leads of each of the low voltage supplies. This device is polarized so you must be careful to insert it into the circuit properly. The end of the device with a band of color around it should be connected to the positive terminal. These devices are made by UNITRODE Corp. of Watertown, Mass. (580 Pleasant Ave.) 02172. For the 5 volt supply you should use a Type UZS306 and for the 24 volt supply a Type UZS330, if your 24 volt supply is regulated, and a Type UZS336, if it is not regulated. If you are using Dynamic memories or A/D or D/A converters which use ±12 or ±15 volts, use type UZS314 on the 12 volt supplies and UZS318 on the 15 volt supplies. These devices present a near infinite impedance until thier breakdown voltage (last two digits of type number) is exceeded. When this voltage is exceeded, they change, in one pico second, to a nearly dead short. When the voltage returns to normal, they revert to their prior state. 8