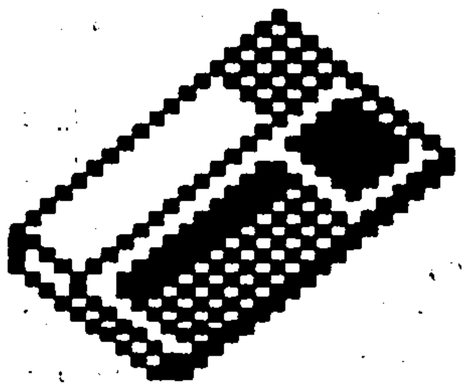


HUG

HOUSTON USERS' GROUP

NOVEMBER
1985

PROP. of HUG
SET "A"
c/o R.Lumpkin
Houston Texas
713-469-5089

MEETING SCHEDULE

FIRST SUNDAY OF EVERY MONTH
(2nd SUNDAY IF 1st SUNDAY
is on a holiday weekend)

HUG TIBBS - (713) 477-8909
24 -hour BULLETIN BOARD

****

THE NEXT MEETING IS

SUNDAY, NOVEMBER 3, 1985 2:00 PM

St. John's School - 2401 Clairmont

****

IN THIS ISSUE

****

PRESIDENT'S REPORT

As it is just a little over a month until we will be electing new officers, I would like to take the opportunity to announce the nominating committee. The committee consists of GARY CURRY 453-0142, BILL RISTER 537-8596 or the Phoenix, and Cecil Crowder Box 5310 Pasadena 77508-5310. If you would like to serve HUG next year by being an officer, contact one of these men and tell them of your interest. Below is a list of the offices and what is needed for each.

PRESIDENT - no special system requirements, but need to be able to speak in front of a group. Telephone helpful.

VICE-PRESIDENT MEMBERSHIP - Requires disk, printer, E/A cart. In charge of maintaining membership list, collecting dues and sending out membership information for new members.

VICE-PRESIDENT SPECIAL INTEREST GROUPS - Disk, printer helpful. In charge of setting up special educational programs or seminars for small groups.

VICE-PRESIDENT PROGRAMS - no special equipment required. Should have some knowledge of what people want to see at the meetings each month and is in charge of planning the programs.

SECRETARY - Requires word processor or typewriter - In charge of keeping records of the group, giving notices of meetings and general correspondence.

TREASURER - Requires disk, Multiplan, printer - In charge of the finances of the group. Knowledge of Multiplan a must.

LIBRARIAN - Requires 2 DSDD disk drives, DSDD Disk Controller, printer, Multiplan, TI-Writer, P-code card. Accept & check programs submitted by members, fill orders each month, keep catalog updated, keep records as to members' credit.

NEWSLETTER EDITOR - Requires disk, TI-Writer, printer. Put out newsletter each month, including collecting information, formatting, taking to printer, assembling and mailing. Requires at least 1 weekend a month.

SysOp - Requires knowledge of programming and Disk Repair, daily maintanance of 1 to 2 hours if no problems arise, space for club's system. Most time-consuming office we have. Understanding family helpful.

This is your group. If there is some way you can serve, tell the nominating comittee.

Bill W. Knecht

HOLIDAY CONTEST

This is to announce a programming contest for the HUG Newsletter. The only hardware and software requirements are that you produce an original progra on the TI-99/4A COMPUTER and that you use either basic or extended basic.

The prize will be either Miller Graphics Advanced Diagnostics or Explorer. The prize will be awarded at the December Meeting of HUG as the first order of business.

The Judges will be Rogers G. Mills Jr., HUG Editor; Cecil Crowder. HUG Sysop; and Bill Knecht, HUG President. The decision of the judges will be final and all judging authority will rest with them.

The following rules will apply to the programing :

1 : The progra must be written in basic or extended basic and not be protected.
2 : The graphics must be original programing.
3 : The music can be a popular tune or tunes provided that the progra reflects the name of the tune either in the first screen visible in the progra or in the first screen of the program listing.
4 : The contestant must submit the progra to the judges no later than 3 PM (one hour after the meeting begins) on the day of the November Meeting of HUG. This should allow for more than sufficient time for anyone who might arrive late.
5 : The judges will complete authority and their decision will be final. The judges, the staff of the Newsletter and their families are not eligible for the contest.
6 : Announcement of the winner will be at the December meeting and the program will be published in the December issue of the HUG.
7 : The progra must be submitted on either cassette or disk medium in a runable form. Print-outs are not required. The progra and all its contents will be placed in the Library and will be deemed public domain.

DESCRIPTIVE FILE FOR USING DISKFIXER
TO REPAIR BLOWN DISKS. D/L'D
FROM THE "SPIRIT OF 99" 11885. DALE RINDLEY. SYSOP.
HOW TO FIX DISKS
BY Mirai N. Onan Mike Dallman

This article originally appeared in the Spirit Of 99, the Newsletter Bulletin Board of The Central Ohio Ninety Niner's

The Board Number is 614-451-0880 Did you ever try to catalog a disk and find out the Disk Controller thinks the disk is NOT initialized? But you know better! What do you usually do with the blown disk? Most people delete the file giving them the problem. Usually that does correct the problem, but it also gets rid of that file forever. The ultimate solution is to use DISK FIXER by Navarone Industries.

The DISK FIXER enables one to examine and change the contents of any disk on a a sector- by- sector basis. I think it is worth its forty-dollar list price. It is available from some TI retailers or directly from Navarone Industries.

Here is the process to fix a blown-up disk...

First acquire a DISK FIXER from a freind buy one, they're worth it. Get a hard- copy catalog of the blown disk, or even better, get a complete (old) catalog of what should be on the disk. If a comple te catalog is not available try to remem ber what should be on the disk and write those names down on paper. Once you have a catalog of the disk, you are ready to start using DISK FIXER.

Insert the DISK FIXER cartridge and se lect option 2 from the Title Screen. Upon doing so you should see the DISK FIXER menu. Do the following if the most recent catalog of the bad disk tells you there are more sectors used/free than is logically posssible: 358 for single side d 718 for double sided disks. For ex ample, IF the catalog lists 500 sectors used/free on a single-sided disk THEN do the following ELSE GOTO the paragraph on "SECTOR ONE".

This part tells you how to fix up Sector 0; which is the sector containing the in formation concerning the disk name and number of sectors used/free on the disk. If the disk catalog tells you the used/ free sector information is in error then Sector 0 needs to be fixed. The easiest way to do this is to copy a good Sector 0 from another disk to the blown disk. Here is how to do that:

1) Insert a good disk in drive
2) Read Sector 0 of that disk:
   R 0,1 [ENTER]
3) Put the blown disk in drive
4) Write good Sector 0 to disk:
   W 0,1 [ENTER]

If you catalog the bad disk, you will see that the diskname and the used/free information is the same as the good disk But do not let that alarm you. We did that to fool the Disk Controller into thinking the bad disk is at least partially restored to normalcy. Now we need to fix up the blown disk as much as we can This is done by changing Sector 1. Here is how to fix Sector 1. First, get the most complete catalog and the most recent catalog of the bad disk in front of you. Then compare the two catalogs to see which filenames are missing. Next, compile an alphabetical list of all the filenames which are and should be in the catalog.

Then you need to find the corresponding sector for each filename. This is done by using the Find String function of the DISK FIXER

1) Put the bad disk in drive
2) Find a filename by:
   F 0,2D0,1 [ENTER]
   type in the filename [ENTER]
3) Ignore the "ERROR IN SECTOR" message
4) Write down the sector number for that filename
5) If that filename could not be found make sure you typed it in correctly and try again; otherwise that file does not exist on the disk.

6) Repeat the process from step two for all of the filenames

You should now have an alphabetical list consisting of two columns: filenames and sectors. With that information in hand you are ready to begin fixing up the bad disk. This is done by modifying Sector 1 of the blown disk. First you have to read Sector 1 from the bad disk by doing this:

1) Put the bad disk in drive
2) Read Sector 1 of disk by:
   R 1,1 [ENTER]

Then you want to alter the contents of sector 1. This is done by using the alter function of the DISK FIXER. This process is best learned by observing a con crete example.

Lets say the blown disk has 14 files (filenames) on it. Thus there should be 14 entries on sector 1; one entry for each file. The rest of the sector should be all zeros. Lets alter Sector 1:

1) Keep the bad disk in drive
2) Enter the Alter function:
   A 0 [ENTER]
3) Type in the following just as shown, including the spaces:
   1 2 3 4 5 6 7 8 9 A B C D E
4) Do not press [ENTER] yet!
5) If you saw a non-zero entry after the E entry in the first column then type in [0] and a [SPACE] and repeat until the first column shows a zero.
6) Press [ENTER]

7) Write the revised Sector 1 to the bad disk:
W 1,1 [ENTER]

You have just entered a table of pointers to the files on the disk. The table points to the corresponding sector for each file name. This is the table that is updated and sorted if you add/delete files to the disk.

Leave the DISK FIXER by typing [Q] for QUIT and press [ENTER]. Then catalog the disk. Lets call this new catalog the mixed catalog. You will see the reason once the disk has been cataloged. Notice how the catalog is NOT in alphabetical order It does however contain all of the file names that you hoped and prayed would be on the disk! The next step is to alphabetize the catalog. This is done by first alphabetizing the catalog on paper and carrying along the appropriate sector number of each filename. Here is an example of a Mixed Catalog.

MIXED CATALOG		SORTED CATALOG	
FILENAME	SECTOR	FILENAME	SECTOR
CAT	1	APPLE	E
SCREEN	5	CAT	1
VOTE	2	DEMO	7
FIRE	6	FIRE	6
APPLE	E	HELLO	9
HELLO	9	JUSTIFY	D
SCROLL	C	LOAD	3
LOAD	3	LOGO	A
TIME	8	PLOT	B
DEMO	7	QUICK	4
QUICK	4	SCREEN	5
JUSTIFY	D	SCROLL	C
PLOT	B	TIME	8
LOGO	A	VOTE	2

The above example shows how you should alphabetize the filenames and the corresponding sector numbers on paper. If you are unsure when dealing with funny characters, the system alphabetizes by lower to higher ASCII values. These values can be found on your TI Basic reference card Once you have done this you are ready to enter this information into Sector 1. You do not have to enter the filenames, just the sector numbers.

Here is how to do that:

1) Put the blown disk in drive
2) Read Sector 1 by entering:
R 1,1 [ENTER]
3) Enter the Alter function:
A 0 [ENTER]
4) Type in the sector numbers in the order as shown for the above sorted example catalog. Seperate each number by a space:

E 1 7 6 9 D 3 A B 4 5 C 8 2

5) Then press [ENTER]

6) Write revised sector to disk:
W 1,1 [ENTER]
7) Put a Write-Protect tab on the disk!

You have now fixed up the disk. For verification quit the DISK FIXER program and catalog the disk. You should have no problems during the cataloging process. But you are not completely done yet! DO NOT add/delete any files or programs to this disk!

Get a fresh disk and initialize it to the same configuration as the blown disk Then backup the blown disk to the fresh disk. Then catalog the fresh disk and you will see that the used/free sector information is now correct. Thus, the fresh disk is now your working disk and the blown disk is now a disk for your archives.

Keep the blown disk in a safe place just in case you remember a file that was not previously recovered from the blown disk Go through the above procedures to recover that new-but-old file.

If you have any questions on how to fix fix up blown disks please leave private mail to MIKE WALLMANN (that's TWO N's)

Happy fixing!

AN ASSEMBLY TUTORIAL ON DEVICE

SERVICE ROUTINES

BY: Jim Kice

No matter what device it is you may want to use, be it a disk drive, printer, modem, or whatever other device you may have hooked up to your computer, it can be accessed through assembly language. All peripherals except for the cassette recorder are accessed through routines called Device Service Routines. I'll be calling them DSR's from herein. (It's much easier to type!)

DSR's can be tricky to set up. That's why I've written this article, to clear up the incomprehensible lore of the infamously confusing Editor/ Assembler manual. I hope what follows will help you to understand how DSR's work. Once you understand how they work, they're easy to use.

When accessing any device in assembly language, the first step is to put a reference into your REF/DEF table telling the computer that you will be using a DSR. All you do is include the following at the start of your program:

```
REF DSRLNK
```

I will also discuss another routine called the LOADER. The format for the LOADER is exactly like that of the DSRLNK routine. To include the LOADER routine in your program enter the following at the start of your program:

```
REF LOADER
```

That's easy enough! The next step is more complicated, but shouldn't be any problem either. This next step is setting up a Peripheral Access Block(PAB). The format for a PAB is the same no matter what peripheral you wish to access. The format is also the same for both the DSRLNK and the LOADER routines. There are ten(0-9) bytes in a PAB plus however many characters there are in the device name of the device you wish to access. For example, a PAB for the printer using the device name of "RS232/1" would be 17 bytes long. Ten for the PAB data and seven for the device name

```
"RS232/1."
 1234567
```

The first nine bytes of any PAB must contain certain information. What each byte must contain is as follows:

BYTE	BIT	CONTENTS	MEANING
0	All	I/O Opcode	>00=open
			>01=close
			>02=read
			>03=write
			>04=restore/ rewind
			>05=load
			>06=save
			>07=delete
			>08=scratch record
			>09=status
1	All	Flag/Status	
	0-2	Error code	000=bad device name
			001=device is write protected
			010=bad open attribute eg. incorrect file type, length etc.
			011=illegal operation eg. operation conflicting with open attributes
			100=out of butter space
			101=attempt to read past end of file
			110=device error eg. damaged disk, wrong parity for RS232, etc.
			111=file error. eg. file type mismatch, reading from unopened file, etc.
	3	Record Type	0=fixed 1=variable

	4	Data Type	0=display 1=internal
	5,6	Operation Mode	00=update 01=output 10=input 11=append
	7	File Type	0=sequential 1=relative
2,3	All	Data Buffer Address	VDP RAM address of data buffer to be read from or written to.
4	All	Logical Record Length	Logical length in fixed, max. length in variable.
5	All	Character Count	Number of characters to read from or write to a file at one time.
6,7	All	Record Number	Only for relative files. Which record # current I/O operation is being done to.
8	All	Screen Offset	Set to 0 for all peripherals except cassette.
9	All	Name Length	Number of characters in device name.
10+	All	File Descriptor	Device name eg. "RS232" or "DSK1.F1"

Byte #0 is fairly self- explanatory. Example:Your data you would have byte #0 as >00 to open the file. then change it to >03 to write a record to the file, then >01 to close the file.

You will always open a file with the first 3 bits of Byte #1 equal to zero so you don't get an error. Bits #4-#7 are set depending upon which setting you choose.

Bytes #2-#3 should be set the the address in VDP RAM where you want the data you read or write to be stored. >1000 is usually the best place to start your buffer at because it is unused by any RUM or BRUM routines.

Byte #4 should be set to the record length you wish to use. It's the "80" in "variable 80" to help clarify what I mean. For a logical record length of 80, you would convert it to hexadecimal and enter >50.

Byte #5 should be set to the number of bytes you wish to read or write to or from a record. A value of >00 sets this to the standard value(the value in Byte #4.)

Bytes #6-#7 are set to >0000 unless relative files are used in which case, the record number you wish to read or write from is entered in hexadecimal. You may still read/write from the first record by entering >0000 even if you are using relative files.

Byte #8 should be set to >00.

Byte #9 should equal the number of characters in the device name. For example, if your device name is "DSK1.F" then Byte #9 would contain >06 since "DSK1.F" is 6 characters long.

Bytes 10+ contain the device name such as "DSK1.F". Bytes #0-#9 use DATA directives. Bytes #10+ use a TEXT directive.

Here is an example of a PAB:

```
EDATA  DATA >0005,>1000,>5000,>0000
       DATA >000C
       TEXT 'DSK1.EXAMPLE'
```

Byte #0 gives the open opcode. Byte #1 says the file is a fixed length, display, input mode, and relative file. Bytes #2-#3 say the buffer address to read from/write to is >1000. Byte #4 specifies 80 as the logical record length. Byte #5 defaults(because it is >00) to 80 as the number of bytes to read/write at one time. Bytes #6-#7 set the record number to 0. Byte #8 is set to >00 as always. Byte #9 is set to 12, the number of bytes in the device name given in bytes 10+. The device name given in bytes 10+ is "DSK1.EXAMPLE". NOTICE THE TEXT DIRECTIVE USES SINGLE QUOTES!

The next step after setting up your PAB is to decide where to put your PAB in memory. Since the PAB is stored in VDP RAM, I recommend placing it anywhere between >0B00 AND >0FCF. Placing your PAB's before >0B00 will overwrite the character set and after >0FCF will overwrite your read/write buffer. Store the address you decide upon in an EQU directive with a label like this:

```
PABADD EQU >FB0
```

From then on, PABADD equals the address of the first byte of your PAB. This makes it easier to change a byte in your PAB. For example, if you wanted to change the number of characters your file will read/write at one time (Byte #5), you would change it like this:

```
        LI R0,PABADD+5(for byte
                      #5)
        LI R1,NEW VALUE
             TO ENTER
        BLWP @VSBW
```

One other thing you must do when setting up a PAB is load a copy of the length of the device name into >8356. All you would do is this:

```
      LI R3,PABADD+9
      MOV R3,@>8356.
```

Two examples of file access, one using DSKLNK and one using LOADER follow.

This is an example of writing to a file. This example writes to the device "RS232/2". You may change the device name to fit your RS232 specifications. Remember to change byte #9 of PDATA to the new number of characters in the device name if you change it!

```
      REF DSRLNK,VMBW,VMBR,VSBW  * REFERENCES TO ROUTINES
                                 * USED IN PROGRAM
      DEF RUN      * PROGRAM NAME
PABBUF EQU >1000   * FILE BUFFER STARTING ADDRESS
PABADD EQU >F80    * PAB STARTING ADDRESS
STATUS EQU >837C   * STATUS BYTE ADDRESS
PNTR   EQU >8356   * ADDRESS OF POINTER TO LENGTH OF
                   * DEVICE NAME IN PAB
SAVRTN DATA 0      * BYTE TO SAVE RETURN ADDRESS
PRINT  TEXT 'This is a test for the TI-99/4A DSR routine.'
                   * MESSAGE ABOVE WRITTEN TO FILE
PDATA  DATA >0012,PABBUF,>0050,>0000,>0007   * PAB DATA
       TEXT 'RS232/2'                        "
       EVEN              * SETS ADDRESS BACK TO EVEN NUMBER
WRITE  BYTE >03  * REPRESENTS BYTE TO LOAD IN BYTE #0 OF
CLOSE  BYTE >01  * PAB TO WRITE TO FILE
MYREG  BSS >20   * BUFFER FOR MY OWN WORKSPACE REGISTER
BUFFER BSS 80    * BUFFER TO READ/WRITE DATA FROM/TO FILE
RUN    MOV R11,@SAVRTN  * SAVE RETURN ADDRESS
       LWPI MYREG       * LOAD MY OWN WORKSPACE REGISTERS
       LI R0,PABADD     *
       LI R1,PDATA      *
       LI R2,>20        *  LOAD PAB INTO PAB ADDRESS >F80
       BLWP @VMBW       *
       LI R6,PAB+9
       MOV R6,@PNTR     * LOAD POINTER WITH DEVICE NAME LENGTH
       BLWP @DSKLNK     * OPEN FILE
       DATA 8           *    "
       MOVB @WRITE,R1   *
       LI R0,PAB        * SET UP FILE WITH WRITE I/O OPCODE
       BLWP @VSBW       *
       LI R0,PABBUF     *
       LI R1,PRINT      *
       LI R2,44         * LOAD DATA TO WRITE TO FILE
       BLWP @VMBW       *
       MOV R6,@PNTR     *
       BLWP @DSKLNK     * WRITE TEXT TO FILE
       DATA 8           *
       MOVB @CLOSE,R1   *
       LI R0,PABADD     * LOAD DATA TO CLOSE FILE
       BLWP @VSBW       *
       MOV R6,@PNTR     *
       BLWP @DSKLNK     * CLOSE FILE
       DATA 8           *
       CLR R0
       MOVB R0,@STATUS  * CLEAR STATUS BYTE
       MOV @SAVRTN,R11  * RELOAD ADDRESS TO RETURN TO
       RT               * RETURN TO E/A
       END
```

This is an example of using the LOADER routine. It asks you for which of two programs you wish to have loaded into the Editor/Assembler. It will then load your choice. You will have to modify this to correspond to files that you have. NOTE: If your program doesn't automatically run, you will press enter after selecting your program, choose option #4(RUN) and enter your program name. The file will already be in memory. You may modify this to accomodate more than a two file selection by adding more PAB's similar to those in the sample program, adding more key values to R6-R10 and R12-R15 and using a CB R1,R(whichever) and a JEQ (to your label) Your label should start a section with instructions exactly like those of the ALOAD or BLOAD sections of the program except for the PAB data labelled, ADATA or BDATA.)Change that to the PAB label corresponding to the program name you have chosen.

```
      REF DSKLNK,VMBW,VSBW,LOADER,KSCAN     *
      DEF RUN                               *
PABBUF EQU >1000                            *
PABADD EQU >F80                             *
STATUS EQU >837C                            *
PNTR   EQU >8356                            *
SAVRTN DATA 0                               *
ADATA  DATA >0005,PABBUF,>5000,>0000,>0000  * BASICALLY
       TEXT 'DSK1.EXAMPLE1'                 * SAME AS
       EVEN                                 * FIRST
BDATA  DATA >0005,PABBUF,>5000,>0000,>0000  * EXAMPLE
       TEXT 'DSK1.EXAMPLE2'                 *
       EVEN                                 *
CLOSE  BYTE >01                             *
MYREG  BSS >20                              *
EXAM1  TEXT '1. EXAMPLE 1'  * TEXT FOR CHOICE #1
EXAM2  TEXT '2. EXAMPLE 2'  * TEXT FOR CHOICE #2
RUN    LI R0,34     *
       LI R1,EXAM1  *
       LI R2,12     * WRITE TEXT FOR CHOICE #1 TO SCREEN
       BLWP @VMBW   *
       LI R0,98     *
       LI R1,EXAM2  *
       LI R2,12     * WRITE TEXT FOR CHOICE #2 TO SCREEN
       BLWP @VMBW   *
KPREP  CLR R0
       MOVB R0,@>8374  * SET KEYBOARD SCAN TO ENTIRE KEYBOARD
       LI R3,>2000  * DATA FOR CHARACTER MASK
       LI R4,>3100  * ASCII VALUE FOR "1"
       LI R2,>3200  * ASCII VALUE FOR "2"
KCHECK CLR R1
       BLWP @KSCAN  * SCAN KEYBOARD
```

```
      MOVB @STATUS,R5 ;
      COC R3,R5       ; CHECK IF KEY HAS BEEN PRESSED
      JNE KCHECK      ;
      MOVB @>8375,R1  ; LOAD ASCII VALUE OF KEY
                      ; PRESSED INTO R1
      CB R1,R4        ; COMPARE KEY PRESSED TO "1"
      JEQ ALOAD       ; IF KEY PRESSED="1" THEN GO TO
                      ; ROUTINES TO LOAD PROG. #11
      CB R1,R2        ; COMPARE KEY PRESSED TO "2"
      JEQ BLOAD       ; IF KEY PRESSED="2" THEN GO TO
                      ; ROUTINE TO LOAD PROG. #2
      JMP KPREP       ; IF NEITHER KEY WAS PRESSED,
                      ; GO BACK AND CHECK AGAIN
ALOAD MOV R11,@SAVRTN ; LOAD ADDRESS TO RETURN TO,
                      ; INTO SAVRTN
      LWPI MYREG      ; LOAD OWN WORKSPACE REGISTERS
      LI R0,PABADD    ;
      LI R1,ADATA     ;
      LI R2,>20       ; LOAD PAB INTO >#80
      BLWP @VMBW      ;
      LI R6,PAB+9     ;
      MOV R6,@PNTR    ; LOAD POINTER ADDRESS WITH
                      ; LENGTH OF DEVICE NAME
      BLWP @LOADER    ; LOAD FILE
      JMP CLOSEF      ; GO TO CLOSE FILE ROUTINE
BLOAD MOV R11,@SAVRTN ;
      LWPI MYREG      ;
      LI R0,PABADD    ;
      LI R1,BDATA     ;
      LI R2,>20       ; SAME AS ALOAD EXPLANATION
      BLWP @VMBW      ;
      LI R6,PAB+9     ;
      MOV R6,@PNTR    ;
      BLWP @LOADER    ;
CLOSEF MOV R6,@PNTR   ;
      MOVB @CLOSE,R1  ;
      LI R0,PAB       ; LOAD CLOSE FILE DATA
      BLWP @VSBW      ;
      MOV R6,@PNTR
      BLWP @DSRLNK    ; CLOSE FILE
DATA 8                ; CLOSE FILE
CLR R0
```

```
      MOVB R0,@STATUS ; CLEAR STATUS BYTE
      MOV @SAVRTN,R11 ; RELOAD SAVED RETURN ADDRESS
      RT              ; RETURN TO E/A
      END
```

You may have noticed that after every BLWP @DSRLNK, there is a DATA 8 directive. This is necessary and should be after every BLWP @DSRLNK in your program.

Although these two programs are examples, they will both work for you with only slight modifications. All you must do to the first example is change the PAB byte #9 to the length of the device name you wish to use and change the TEXT directive right after it to the device name you want to use. The second example needs the same changes plus changing the TEXT directives with the program choices in it to the names of the programs you wish to run. Also, remember to change the two occurances of R2 as 12 right after the LI R1,EXAM1 and LI R1,EXAM2 statements to the length of the new values you have changed EXAM1 and EXAM2 to. For example, if EXAM1 is changed to TEXT '1. DEVICENAME', THEN you would change the R2 right after the LI R1,EXAM1 to LI R2,13 because EXAM1 is now 13 characters long.

It may be easier to delete the text around the examples and save the examples modified the way you want them to disk using the E/A editor. The program name for both examples is "RUN".

I hope you have learned something from this tutorial. Good luck with assembly language and Happy Computing!!!

Jim Rice

```
00120       A0000B0000B546BB6973B2069B7320B6120B7465B7374B20667F334F    0001
A0012B6F72B2074B6865B2054B492DB3939B2F34B4120B4453B5220B726F7F2BAF    0002
A0028B7574B696EB652EB0012B1000B0050B0000B0007B5253B3233B322F7F2FEF    0003
A003EB3200B0301A0042A0062A00B2BC80BC0000B02E0C0042B0200B0FB07F306F    0004
A00BEB0201C002EB0202B0020B0420B0000B0206B0FB9BCB06BB356B04207F2FBF    0005
A00D4B0000B0008BD060C0040B0200B0FB0B0420B0000B0200B1000B02017F347F    0006
A00EAC0002B0202B002CB0420C00C8BC806BB356B0420C00B4B0008BB0607F2D6F    0007
A0100C0041B0200B0FB0B0420C00E2BC806BB356B0420C00FAB0008B04C07F2E9F    0008
A0116BD8008B37CBC2E0C0000B0458B7F92FF                                 0009
30110DSRLNK300F2VMBW  40000VMBR  30108VSBW  5008#2RUN   7F338F      0010
:       99/4 AS                                                       0011
```

D.C.ELECTRONICS
5206 KINGSMILL
FRIENDSWOOD, TEXAS
(713) 482-0186

LEGEND 808 100CPS DOT MATRIX PRINTER.....$169.95

DATABIOTICS "4A/TALKS".....$17.95

MILLER GRAPHICS "EXPLORER".....$21.95

DATABIOTICS "MINI-WRITTER III".....$79.95
(minimum requirements console & T.V.)
(16K to 48K word processor)
(with built in printer interface)

T.I. LOGO II.....$69.95

NAVARONE DATABASE MANAGER.....$54.95

DOUBLE DENSITY DISKETTES 10(PER PACK).....$7.00

HUG LIBRARY CATALOG ADDENDUM
NOVEMBER 1985

4121 CHECKBOOK MANAGEMENT **XB Disk * Memory Exp. required. AnalizeYour spending habits with this checkbook and budget manager program by John Taylor.Optional printed reports or screen display. Complete with 11 page document file. Requires a dedicated disk. 357 sectors

4122 DISK ENVELOPE DESIGNER **XB Printeer required. Prints disk catalog with a cut-and-fold lines to custom make your sleeves.Space provided to include coment lines. Freeware program by Trio+ Software. 117 sectors

4123 ARTICLES of ASSOCIATION DV/80 file Printer required. Prints out the Articles of Association recently adopded by the HUG members. Use TI-Writer. 70 sectors.

4124 TI-WRITER CODES Printer required. File that will print out, with TI-WRITER, the control code list necessary to shange the print styles for the Gemini and epson printers. 10 sectors.

4125 GAS MILLAGE **XB A nice program by Jim Hutchison which will figure gas millage. 3 sectors.

4126 THE DIRECTOR **XB printer optional. Freeware program by Ron Rutledge that catalogs and keeps track of all of your disks. Search routine to look for filenames. Also Prints disk directories in 3 columns either virticle or horizontal. Very usefull program. Documentation is printed out by the program. 134 sectors.

4127 NEAT LIST **XB printer required. Freeware program by Danny Michaels. An excellent programming tool that will print out all variables used in a program. 360 sectors, DEDICATED DISK REQUIRED.

5222 CHORD_SONG **MUSIC MAKER CARTRIDGE REQUIRED. Cute music program by Michael Lowe, composed to be used with th Music Maker Cartridge. 59 sectors.

5223 NICE_MUSIC **MUSIC MAKER CARTRIDGE REQUIRED. Nice music program by Michael Lowe, composed to be used with the Music Maker Cartridge. 59 sectors.

IMPORTANT NOTICE: The Library will once again be offering a holiday gift to any member who brings an INITIALIZED BLANK DISK to the November Meeting. We will copy our selections onto it and return it to you at the December meting. Please pl;ace a label with your name on the disk. If you do not have a disk system, bring a cassette.

M & S COMPUTER SYSTEMS
15918 CAVENDISH DRIVE
HOUSTON, TEXAS 77059
(713) 486-0224

NOVEMBER/DECEMBER 1985 SPECIALS

DISKS/DISK STORAGE

BONUS DISKETTES, LIFETIME WARRANTY. SSDD, 11/BX.....$ 9.95/BX OR 2 FOR 19.00
FLIPPY FLOPPY DISKETTES, DSDD, 10 PK, W/O PLASTIC LIBRARY CASE..... 11.95
WITH " " " 15.95
VERBATIM DATALIFE, LIFETIME WARRANTY, 12/BX.....SSDD>>>> 16.95 DSDD>>>> 23.95
DISK BANK MEDIA MATE 5, HOLDS 50($1.00 OFF W/ PURCHASE OF BOX OF DISKS)..... 11.49
ROLLTOP 100, HOLDS 120 DISKS..... 26.95

PRINTERS & ACCESSORIES

STAR SG-10 NEAR LETTER QUALITY DOT MATRIX PRINTER, 120 CPS..... 249.95
(INCLUDES FREE TI-99/4A ADDENDUM & DEMO SOFTWARE)
PARALLEL PRINTER CABLE(W/ PRINTER PURCHASE... 24.95)..... 29.95
PRINTER STAND WITH PAPER BASKET..... 18.95

DISK DRIVES

TANDON TM65-2L DSDD HALF HIGH 5-1/4" DRIVE..... 122.95
TEAC F-55B DSDD HALF HIGH 5-1/4" DRIVE..... 122.95
MOUNTING KIT FOR 2 HALF HIGH DRIVES FOR PERIPHERAL EXPANSION BOX..... 29.95
EXTERNAL DRIVE CASE W/ POWER SUPPLY FOR 5-1/4" DRIVE(EXT. DRIVE CABLE..15.00). 60.00
PERIPHERAL BOX EXTENSION CABLE, 32"LG(CONNECTS BETWEEN COMPUTER & PEB CABLE).. 22.95
YOUR CHOICE OF 2 HALF HIGH DRIVES PLUS MOUNTING KIT..... 274.95

HARDWARE

CORCOMP "TRIPLE TECH" CARD W/ CLOCK/CALENDAR, 114.95
64K PRINTER BUFFER & SPEECH SYNTHESIZER CONNECTION
CORCOMP RS-232 CARD(FOR PEB)..... 84.95
"SUPER SKETCH" GRAPHICS PAD,REQUIRES CONSOLE ONLY 39.95
AXIOM PARALLAX TI DIRECT CONNECT PRINTER INTERFACE 77.95
SIGNALMAN MARK III, 300 BAUD, DIRECT CONNECT MODEM 73.95
SIGNALMAN EXPRESS,300/1200 BAUD,DIRECT CONNECT 279.95

SOFTWARE

BITMAC(DATABIOTICS)XB,E/A,OR M/M,32K,JS,DRIVE..... 31.95
ADVANCED DIAGNOSTICS(MILLER GRAPHICS)XB OR E/A,32K 17.95
99-WRITER II,TI WRITER CLONE(TEX-COMP)XB,32K,DRIVE 19.95

CLOSE OUT ON TI SOFTWARE
LIMITED TO STOCK-ON-HAND

JAWBREAKER II.....$ 5.95
HOPPER..... 5.95
MOON MINE..... 5.95
TREASURE ISLAND..... 5.95
SUPER DEMON ATTACK.... 5.95
MUNCH MOBILE..... 5.95
SLYMOIDS..... 5.95
MINER 2049'ER..... 18.95
ESPIAL..... 18.95
Q-BERT,FROGGER,POPEYE 19.95
MOON PATROL..... 16.95
DEFENDER..... 16.95
PROTECTOR II..... 16.95
RABBIT TRAIL..... 6.00
HEN HOUSE..... 6.00
ADVENTURE MODULE(W/CAS) 6.50
VOODOO CASTLE,THE COUNT 6.50

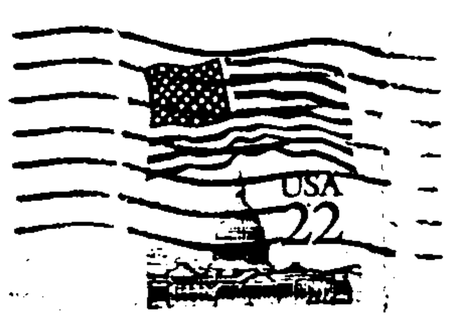
CORCOMP 9900 STAND ALONE CLOCK/CALENDAR..... 73.95
CORCOMP DSDD CONTROLLER CARD(FOR PEB)..... 164.95
MYARC DSDD CONTROLLER CARD(FOR PEB)..... 169.95
MBX SYSTEM(CARTRIDGES ONLY $9.95)..... 54.95
TI-99/4A BLACK & SILVER CONSOLE W/ 1 YR. WARRANTY 75.00
VOLKS 300 MODEM W/TI CABLE,300 BAUD,DIRECT CONNECT 68.95
COMPUSERVE STARTER KIT, 5 HOURS FREE CONNECT TIME 21.95

4A/TALK(DATABIOTICS) TERMINAL EMULATOR,110-9600B 17.75
EXPLORER(MILLER GRAPHICS)XB,E/A,~~M/M~~, 32K, DRIVE.. 21.95
ADVENTURE EDITOR(TEX-COMPIE/A,32K,DRIVE..... 26.95

NEW SCOTT ADAMS ADVENTURES: SPIDERMAN, HULK, BUCKAROO BANZAI, CLAYMORE CASTLE(D OR C)REQUIRES ~~ADVENTURE~~ MODULE. 17.95EA
BUY ANY 3 NEW ADVENTURES & RECEIVE THE 4TH FREE ALONG WITH NEW THE ADVENTURE HINT BOOK & TI ADVENTURE MODULE
WE SHIP UPS, TEX-PACK OR US MAIL. ORDERS CAN BE PREPAID OR SHIPPED COLLECT. FREE SHIPPING ON SOFTWARE ORDERS OVER $100

Bill W. Knecht
815 Yorkshire
Pasadena, Tx. 77503

U.S. POSTAL SERVICE OCT 29 TX 77



0186 Ruth/Sanford Herman
6219 Sanford Road
Houston, Tx 77096