West Jax 99er News

Dedicated to the TI-99/4A Users

AUGUST 1988

The WEST JAX 99'ERS is a non-profit computer users group for the TI-99/4A Home Computer. NOT affiliated in any way with Texas Instruments. The club's mailing address is PO BOX 176 Orange Park Florida 32067.

MEETINGS are held on the Second and Fourth Tuesday of each Month in the auditorium of the Webb Library. It is located two lights west of Blanding Boulevard on 103rd Street. The first meeting of the month is the Business meeting with workshop time after adjournment. The second meeting is strictly workshop time.

OFFICERS - President Rick Felzien (904) 772-9162
Treasurer Thomas LeMay (904) 282-5220
Secretary Ralph Glattli (904) 751-1308
Librarian Zach Ziegler (904) 389-2194
EXCALIBUR IBM/PC BBS Jim Hutchison (904) 751-3970
(Sysop)
Richard Barton
(TIUS SYSOP)

For newsletter suggestions and submissions, contact Rick Felzien.

This month we have another of the Basic Assembler articles by Steve Peacock.
I have included a combination of two of my previous articles on the Advanced Diagnostics package on the disk system and file recovery.

T H E B A S I C A S S E M B L E R #3 By Steve Peacock

CHANGE SCREEN COLOR AND REDEFINE CHARACTERS

This month we will learn how to change the character definition of each character. In order to do this we must write a some data to the proper address. This month I will also show you how to change the color of the screen. This uses the VWTR command. Below you will see a table that shows all of the addresses for the 16 standard character sets, the CURSOR and EDGE CHARACTER. This table is used when redefining a character. Please note, if your program is called from BASIC or EXTENDED BASIC the address is not the same.

	ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC
CURSOR			EDGE								
30	>08F0	>03F0	31	>08F8	>03F8						
SET	ONE		SET	FIVE		SET	NINE		SET	THIRTEEN	
32	>0900	>0400	64	>0A00	>0500	96	>0B00	>0600	128	>0C00	>0700
33	>0908	>0408	65	>0A08	>0508	97	>0B08	>0608	129	>0C08	>0708
34	>0910	>0410	66	>0A10	>0510	98	>0B10	>0610	130	>0C10	>0710
35	>0918	>0418	67	>0A18	>0518	99	>0B18	>0618	131	>0C18	>0718
36	>0920	>0420	68	>0A20	>0520	100	>0B20	>0620	132	>0C20	>0720
37	>0928	>0428	69	>0A28	>0528	101	>0B28	>0628	133	>0C28	>0728
38	>0930	>0430	70	>0A30	>0530	102	>0B30	>0630	134	>0C30	>0730
39	>0938	>0438	71	>0A38	>0538	103	>0B38	>0638	135	>0C38	>0738
SET	TWO		SET	SIX		SET	TEN		SET	FOURTEEN	
40	>0940	>0440	72	>0A40	>0540	104	>0B40	>0640	136	>0C40	>0740
41	>0948	>0448	73	>0A48	>0548	105	>0B48	>0648	137	>0C48	>0748
42	>0950	>0450	74	>0A50	>0550	106	>0B50	>0650	138	>0C50	>0750
43	>0958	>0458	75	>0A58	>0558	107	>0B58	>0658	139	>0C58	>0758
44	>0960	>0460	76	>0A60	>0560	108	>0B60	>0660	140	>0C60	>0760
45	>0968	>0468	77	>0A68	>0568	109	>0B68	>0668	141	>0C68	>0768
46	>0970	>0470	78	>0A70	>0570	110	>0B70	>0670	142	>0C70	>0770
47	>0978	>0478	79	>0A78	>0578	111	>0B78	>0678	143	>0C78	>0778
SET	THREE		SET	SEVEN		SET	ELEVEN		SET	FIFTEEN	
48	>0980	>0480	80	>0A80	>0580	112	>0B80	>0680	144	>0C80	>0780
49	>0988	>0488	81	>0A88	>0588	113	>0B88	>0688	145	>0C88	>0788
50	>0990	>0490	82	>0A90	>0590	114	>0B90	>0690	146	>0C90	>0790
51	>0998	>0498	83	>0A98	>0598	115	>0B98	>0698	147	>0C98	>0798
52	>09A0	>04A0	84	>0AA0	>05A0	116	>0BA0	>06A0	148	>0CA0	>07A0
53	>09A8	>04A8	85	>0AA8	>05A8	117	>0BA8	>06A8	149	>0CA8	>07A8
54	>09B0	>04B0	86	>0AB0	>05B0	118	>0BB0	>06B0	150	>0CB0	>07B0
55	>09B8	>04B8	87	>0AB8	>05B8	119	>0BB8	>06B8	151	>0CB8	>07B8
SET	FOUR		SET	EIGHT		SET	TWELVE		SET	SIXTEEN	
56	>09C0	>04C0	88	>0AC0	>05C0	120	>0BC0	>06C0	152	>0CC0	>07C0
57	>09C8	>04C8	89	>0AC8	>05C8	121	>0BC8	>06C8	153	>0CC8	>07C8
58	>09D0	>04D0	90	>0AD0	>05D0	122	>0BD0	>06D0	154	>0CD0	>07D0
59	>09D8	>04D8	91	>0AD8	>05D8	123	>0BD8	>06D8	155	>0CD8	>07D8
60	>09E0	>04E0	92	>0AE0	>05E0	124	>0BE0	>06E0	156	>0CE0	>07E0
61	>09E8	>04E8	93	>0AE8	>05E8	125	>0BE8	>06E8	157	>0CE8	>07E8
62	>09F0	>04F0	94	>0AF0	>05F0	126	>0BF0	>06F0	158	>0CF0	>07F0
63	>09F8	>04F8	95	>0AF8	>05F8	127	>0BF8	>06F8	159	>0CF8	>07F8

West Jax 99er News

Dedicated to the TI-99/4A users

AUGUST 1988

The WEST JAX 99'ERS is a non-profit computer users group for the TI-99/4A Home Computer. NOT affiliated in any way with Texas Instruments. The club's mailing address is PO BOX 176 Orange Park Florida 32067.

MEETINGS are held on the Second and Fourth Tuesday of each Month in the auditorium of the Webb Library. It is located two lights west of Blanding Boulevard on 103rd Street. The first meeting of the month is the Business meeting with workshop time after adjournment. The second meeting is strictly workshop time.

OFFICERS - President Rick Felzien (904) 772-9162
Treasurer Thomas LeMay (904) 282-5220
Secretary Ralph Glattli (904) 751-1308
Librarian Zach Ziegler (904) 389-2194
EXCALIBUR IBM/PC BBS Jim Hutchison (904) 751-3970
(Sysop)
Richard Barton
(TIUS SYSOP)

For newsletter suggestions and submissions, contact Rick Felzien.

This month we have another of the Basic Assembler articles by Steve Peacock.
I have included a combination of two of my previous articles on the Advanced Diagnostics package on the disk system and file recovery.

T H E B A S I C A S S E M B L E R #3 By Steve Peacock

CHANGE SCREEN COLOR AND REDEFINE CHARACTERS

This month we will learn how to change the character definition of each character. In order to do this we must write a some data to the proper address. This month I will also show you how to change the color of the screen. This uses the VWTR command. Below you will see a table that shows all of the addresses for the 16 standard character sets, the CURSOR and EDGE CHARACTER. This table is used when redefining a character. Please note, if your program is called from BASIC or EXTENDED BASIC the address is not the same.

	ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC		ASSEMBLY	CALL FROM BASIC
CURSOR			EDGE								
30	>08F0	>03F0	31	>08F8	>03F8						
SET	ONE		SET	FIVE		SET	NINE		SET	THIRTEEN	
32	>0900	>0400	64	>0A00	>0500	96	>0B00	>0600	128	>0C00	>0700
33	>0908	>0408	65	>0A08	>0508	97	>0B08	>0608	129	>0C08	>0708
34	>0910	>0410	66	>0A10	>0510	98	>0B10	>0610	130	>0C10	>0710
35	>0918	>0418	67	>0A18	>0518	99	>0B18	>0618	131	>0C18	>0718
36	>0920	>0420	68	>0A20	>0520	100	>0B20	>0620	132	>0C20	>0720
37	>0928	>0428	69	>0A28	>0528	101	>0B28	>0628	133	>0C28	>0728
38	>0930	>0430	70	>0A30	>0530	102	>0B30	>0630	134	>0C30	>0730
39	>0938	>0438	71	>0A38	>0538	103	>0B38	>0638	135	>0C38	>0738
SET	TWO		SET	SIX		SET	TEN		SET	FOURTEEN	
40	>0940	>0440	72	>0A40	>0540	104	>0B40	>0640	136	>0C40	>0740
41	>0948	>0448	73	>0A48	>0548	105	>0B48	>0648	137	>0C48	>0748
42	>0950	>0450	74	>0A50	>0550	106	>0B50	>0650	138	>0C50	>0750
43	>0958	>0458	75	>0A58	>0558	107	>0B58	>0658	139	>0C58	>0758
44	>0960	>0460	76	>0A60	>0560	108	>0B60	>0660	140	>0C60	>0760
45	>0968	>0468	77	>0A68	>0568	109	>0B68	>0668	141	>0C68	>0768
46	>0970	>0470	78	>0A70	>0570	110	>0B70	>0670	142	>0C70	>0770
47	>0978	>0478	79	>0A78	>0578	111	>0B78	>0678	143	>0C78	>0778
SET	THREE		SET	SEVEN		SET	ELEVEN		SET	FIFTEEN	
48	>0980	>0480	80	>0A80	>0580	112	>0B80	>0680	144	>0C80	>0780
49	>0988	>0488	81	>0A88	>0588	113	>0B88	>0688	145	>0C88	>0788
50	>0990	>0490	82	>0A90	>0590	114	>0B90	>0690	146	>0C90	>0790
51	>0998	>0498	83	>0A98	>0598	115	>0B98	>0698	147	>0C98	>0798
52	>09A0	>04A0	84	>0AA0	>05A0	116	>0BA0	>06A0	148	>0CA0	>07A0
53	>09A8	>04A8	85	>0AA8	>05A8	117	>0BA8	>06A8	149	>0CA8	>07A8
54	>09B0	>04B0	86	>0AB0	>05B0	118	>0BB0	>06B0	150	>0CB0	>07B0
55	>09B8	>04B8	87	>0AB8	>05B8	119	>0BB8	>06B8	151	>0CB8	>07B8
SET	FOUR		SET	EIGHT		SET	TWELVE		SET	SIXTEEN	
56	>09C0	>04C0	88	>0AC0	>05C0	120	>0BC0	>06C0	152	>0CC0	>07C0
57	>09C8	>04C8	89	>0AC8	>05C8	121	>0BC8	>06C8	153	>0CC8	>07C8
58	>09D0	>04D0	90	>0AD0	>05D0	122	>0BD0	>06D0	154	>0CD0	>07D0
59	>09D8	>04D8	91	>0AD8	>05D8	123	>0BD8	>06D8	155	>0CD8	>07D8
60	>09E0	>04E0	92	>0AE0	>05E0	124	>0BE0	>06E0	156	>0CE0	>07E0
61	>09E8	>04E8	93	>0AE8	>05E8	125	>0BE8	>06E8	157	>0CE8	>07E8
62	>09F0	>04F0	94	>0AF0	>05F0	126	>0BF0	>06F0	158	>0CF0	>07F0
63	>09F8	>04F8	95	>0AF8	>05F8	127	>0BF8	>06F8	159	>0CF8	>07F8

```
*****************************************************************************
*
*PROGRAM BA3A==>Basic Assembler #3 Assembly Version
*CHANGE SCREEN COLOR AND REDEFINE CHARACTERS
*(C)1985 S. PEACOCK
*
*****************************************************************************
       REF  VMBW,VWTR     *REFERENCE TO MULT. BYTE WRITE AND WRITE TO REG.
       REF  VSBW          *REFERENCE TO SINGLE BYTE WRITE
       DEF  START         *DEFINE START OF PROGRAM (MAY BE ANY WORD)
*************************
START  LI   R0,>0B80      *LOAD THE ADDRESS OF CHAR. 112 IN REG. ZERO
       LI   R1,DEF1       *LOAD THE DEFINITON TO ASSIGN TO CHAR 112
       LI   R2,8          *LOAD THE NUMBER OF BYTES TO WRITE
       BLWP @VMBW         *WRITE THE BYTES
       LI   R0,>0B88      ****THIS SECTION COULD BE WRITTEN:
       LI   R1,DEF2       ****LI   R0,>0B80       (SINCE THE THREE LETTERS
       LI   R2,8          ****LI   R0,DEF1         ARE IN SEQUENCE.  LEAVE OUT
       BLWP @VMBW         ****LI   R2,24           THE LABEL 'DEF2' AND 'DEF3'
       LI   R0,>0B90      ****BLWP @VMBW           BUT NOT THE CODE)
       LI   R1,DEF3
       LI   R2,8
       BLWP @VMBW
*************************
DEF1   DATA >FFB1,>B1B1,>B1B1,>B1FF   *THE HEX CODE FOR THE REDEFINITION
DEF2   DATA >3C42,>A581,>A589,>423C   *
DEF3   DATA >9254,>38FE,>3854,>9200   *
*************************
       LI   R0,>07FF      *WRITE TO REG. 7 THE COLOR WHITE ON WHITE
       BLWP @VWTR         *WRITE TO REG. THIS CHANGES THE TOP AND BOTTOM
*************************BORDER OF THE SCREEN TO WHITE.
       LI   R0,>03B4      *ADDRESS OF CHAR. SET ONE
       LI   R1,>FF00      *CHANGE SET ONE'S COLOR TO WHITE ON WHITE
       BLWP @VSBW         *WRITE NEW COLOR TO SET ONE
*************************
       LI   R0,0          *PRINTS A 'SPACE' TO ALL SCREEN POSITIONS
       LI   R1,>2000      *THUS CHANGING THE SCREEN TO WHITE
LOOP1  BLWP @VSBW
       INC  R0            *INCREASE THE VALUE OF REG. 0
       CI   R0,767        *COMPARE REG. 0 TO 767
       JLE  LOOP1         *IF LESS OR EQUAL TO 767, PRINT ANOTHER 'SPACE'
*************************
       LI   R0,>038E      *ADDRESS OF CHAR. SET 11
       LI   R1,>1000      *CHANGE SET 11'S COLOR TO BLACK ON TRANSPARENT
       BLWP @VSBW
*************************
MES1   TEXT 'p    q    r'   *THE MESSAGE TO PRINT (HAVE BEEN REDEFINED)
       LI   R0,361        *THE STARTING SCREEN POSITION TO PRINT
       LI   R1,MES1       *MESSAGE TO PRINT
       LI   R2,11         *NUMBER OF BYTES TO PRINT
       BLWP @VMBW         *BRANCH TO MULT. BYTE WRITE
*************************
       JMP  $             *JUMP TO SELF (AN ENDLESS LOOP)
       END
```

USING ADVANCED DIAGNOSTICS

By Rick Felzien
West Jax 99'ERS

As I was doing my research in preparation to write a series on the use of the Advanced Diagnostics Program set from Miller's Graphics a thought occurred to me. I realized that perhaps not everyone has had the opportunity to become as familiar with the TI-DOS(Texas Instruments Disk Operating System) as I have. To use the Advanced Diagnostics Package(henceforth Diags.) effectively, one must understand a good bit. about what is on the disk and what that data means to the system.

We shall begin by examining sector zero. This is where the computer tells whether the disk is initialized and keeps track of such pertinent information as sectors used/available, how many sectors are formatted, whether the disk is single or double sided, and whether it is formatted as single or double density. Let us take a look at sector zero.

```
Drive  : 1     Track  : 0
Side   : 1     Sector : 0
Byte   : 0     Display: Hex

0449535420202020202005A01244
534B202B02020000000000000000
0000000000000000000000000000
0000000000000000000000000000
0300000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF
```

This is what you would see on your screen if you used the Edit Sector (ES) command. The information is used in either byte size or word (2 byte) sized units to keep track of the info.

Bytes >0->9 are for the disk name If it is less than 10 characters long the trailing characters are blanks. (To keep things consistant hexedecimal numbers, which is how all data is used and displayed will appear with a preceeding greater than character (>10)).

Bytes >0A->0B denote the number of formatted sectors, >0168=SS/SD, >02D0=DS/SD or SS/DD, and >05A0=DS/DD.

Byte >0C is the number of sectors per track, >09=9, and >12=18(for DD).

Bytes >0D->0F are the letters DSK which the controller look for to see that the disk is initialized by a TI compatible controller.

Byte >10 is used for the protected/not-protected code which was used by TI to protect some of their disks in the beginning. >20 (space char.)=Unprotected, >50 (letter P)=Protected.

Byte >11 denotes tracks per side >23=35, >28=40.

Byte >12 designates the number of sides formatted either >01 or >02.

Byte >13 tells us the formatted density either >01=SD or >02=DD.

bytes >14->55 reserved-not used-all zero's.

Bytes >38->FF are the allocation bit map. This is what tells the disk controller which sectors are used and which are free. Each byte controls 8 sectors and this is controled by the fact that each bit of the particular byte is either a 1 for used or a zero for unused. On this particular disk there are 3 sectors used.

```
*****************************************************************************
*
*PROGRAM BASA==>Basic Assembler #3 Assembly Version
*CHANGE SCREEN COLOR AND REDEFINE CHARACTERS
*(C)1985 S. PEACOCK
*
*****************************************************************************
       REF  VMBW,VWTR     *REFERENCE TO MULT. BYTE WRITE AND WRITE TO REG.
       REF  VSBW          *REFERENCE TO SINGLE BYTE WRITE
       DEF  START         *DEFINE START OF PROGRAM (MAY BE ANY WORD)
**************************
START  LI   R0,>0B80      *LOAD THE ADDRESS OF CHAR. 112 IN REG. ZERO
       LI   R1,DEF1       *LOAD THE DEFINITON TO ASSIGN TO CHAR 112
       LI   R2,8          *LOAD THE NUMBER OF BYTES TO WRITE
       BLWP @VMBW         *WRITE THE BYTES
       LI   R0,>0B88      ****THIS SECTION COULD BE WRITTEN:
       LI   R1,DEF2       ****LI   R0,>0B80       (SINCE THE THREE LETTERS
       LI   R2,8          ****LI   R0,DEF1         ARE IN SEQUENCE.  LEAVE OUT
       BLWP @VMBW         ****LI   R2,24           THE LABEL 'DEF2' AND 'DEF3'
       LI   R0,>0B90      ****BLWP @VMBW           BUT NOT THE CODE)
       LI   R1,DEF3
       LI   R2,8
       BLWP @VMBW
**************************
DEF1   DATA >FF81,>8181,>8181,>81FF   *THE HEX CODE FOR THE REDEFINITION
DEF2   DATA >3C42,>A581,>A589,>423C   *
DEF3   DATA >9254,>38FE,>3854,>9200   *
**************************
       LI   R0,>07FF      *WRITE TO REG. 7 THE COLOR WHITE ON WHITE
       BLWP @VWTR         *WRITE TO REG. THIS CHANGES THE TOP AND BOTTOM
**************************BORDER OF THE SCREEN TO WHITE.
       LI   R0,>0384      *ADDRESS OF CHAR. SET ONE
       LI   R1,>FF00      *CHANGE SET ONE'S COLOR TO WHITE ON WHITE
       BLWP @VSBW         *WRITE NEW COLOR TO SET ONE
**************************
       LI   R0,0          *PRINTS A 'SPACE' TO ALL SCREEN POSITIONS
       LI   R1,>2000      *THUS CHANGING THE SCREEN TO WHITE
LOOP1  BLWP @VSBW
       INC  R0            *INCREASE THE VALUE OF REG. 0
       CI   R0,767        *COMPARE REG. 0 TO 767
       JLE  LOOP1         *IF LESS OR EQUAL TO 767, PRINT ANOTHER 'SPACE'
**************************
       LI   R0,>03BE      *ADDRESS OF CHAR. SET 11
       LI   R1,>1000      *CHANGE SET 11'S COLOR TO BLACK ON TRANSPARENT
       BLWP @VSBW
**************************
MES1   TEXT 'p    q    r'  *THE MESSAGE TO PRINT (HAVE BEEN REDEFINED)
       LI   R0,361        *THE STARTING SCREEN POSITION TO PRINT
       LI   R1,MES1       *MESSAGE TO PRINT
       LI   R2,11         *NUMBER OF BYTES TO PRINT
       BLWP @VMBW         *BRANCH TO MULT. BYTE WRITE
**************************
       JMP  $             *JUMP TO SELF (AN ENDLESS LOOP)
       END
```

USING ADVANCED DIAGNOSTICS

By Rick Felzien

West Jax 99'ERS

As I was doing my research in preparation to write a series on the use of the Advanced Diagnostics Program set from Miller's Graphics a thought occurred to me. I realized that perhaps not everyone has had the opportunity to become as familiar with the TI-DOS(Texas Instruments Disk Operating System) as I have. To use the Advanced Diagnostics Package(henceforth Diags.) effectively, one must understand a good bit about what is on the disk and what that data means to the system.

We shall begin by examining sector zero. This is where the computer tells whether the disk is initialized and keeps track of such pertinent information as sectors used/available, how many sectors are formatted, whether the disk is single or double sided, and whether it is formatted as single or double density. Let us take a look at sector zero.

```
Drive   : 1     Track   : 0
Side    : 1     Sector  : 0
Byte    : 0     Display: Hex

5445535420202020202005A01244
534B2028020200000000000000000
0000000000000000000000000000
0000000000000000000000000000
0300000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
000000000000000000000000FFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF
```

This is what you would see on your screen if you used the Edit Sector (ES) command. The information is used in either byte size or word (2 byte) sized units to keep track of the info.

Bytes >0->9 are for the disk name If it is less than 10 characters long the trailing characters are blanks. (To keep things consistant hexedecimal numbers, which is how all data is used and displayed will appear with a preceeding greater than character (>10)).

Bytes >0A->0B denote the number of formatted sectors, >0168=SS/SD, >02D0=DS/SD or SS/DD, and >05A0=DS/DD.

Byte >0C is the number of sectors per track, >09=9, and >12=18(for DD).

Bytes >0D->0F are the letters DSK which the controller look for to see that the disk is initialized by a TI compatible controller.

Byte >10 is used for the protected/not-protected code which was used by TI to protect some of their disks in the beginning. >20 (space char.)-Unprotected, >50 (letter P)=Protected.

Byte >11 denotes tracks per side >23=35, >28=40.

Byte >12 designates the number of sides formatted either >01 or >02.

Byte >13 tells us the formatted density either >01=SD or >02=DD.

bytes >14->55 reserved-not used-all zero's.

Bytes >38->FF are the allocation bit map. This is what tells the disk controller which sectors are used and which are free. Each byte controls 8 sectors and this is controled by the fact that each bit of the particular byte is either a 1 for used or a zero for unused. On this particular disk there are 3 sectors used.

```
Drive  : 1    Track  : 0
Side   : 1    Sector : 1
Byte   : 0    Display: Hex

0013001A00030014000400050006
001F001B0015001B000700190008
00150009000A000B001D000C0016
0010000D00110002000E0012000F
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
00000000
```

Sector 1 is the directory link and tells the disk drive where to look for the directory sectors for the files.

Although the files are placed on the disk in the order that they are saved, the link numbers are shuffled to give correct positions for the alphabetical which shows up on a catalog operation.

This data is stored in one word or two byte blocks. The first alphabetical or A program on this particular disk has it's directory link on sector 0013 or >13, but 0013 is the first number thus denoting that it is the first in alphabetical order.

When a file is deleted, it is not actually erased. The link number is removed from this sector and the bit map on sector zero is changed, but the data is still on the origional sectors and is merely overwritten as more files are added to the disk.

```
Drive  : 1    Track  : 0
Side   : 1    Sector : 2
Byte   : 0    Display: Hex

50555A5A4C455220202000000100
000DBE0000000000000000000000
22C0000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
00000000
```

Sectors >2->22 are called the File headers and sometimes called the File Descriptor Blocks.

Bytes >0->9 make up the filename up to 10 characters.

Bytes >A->B are zero's and are not currently used for data.

Byte >C tells the controller the filetype. If the file is protected, the value of 8 is added to the unprotected code number.

Type	Unprotected	Protected
DIS/FIX	00	08
DIS/VAR	80	88
INT/FIX	02	0A
INT/VAR	82	8A
PROGRAM	01	09

Byte >D denotes the number of records per sector. This number equals the sector size (256 bytes) divided by the record length--(>100/>50 =>3 or 256/80 = 3). Program files always=0, DIS or INT/FIX 40 = >06, DIS or FIX 60 = 04, DIS or FIX 80 = 03 etc.

Bytes >0E->0F equal number of sectors in the file (not including the file descriptor). This is the cataloged length minus 1.

In the allocation bit map the format is as follows: SS/SD uses bytes >38->64, SS/DD or DS/SD use bytes>65->91, and DS/DD uses bytes >92-EB, and >EC->FF are unused and are formatted to all F's. or used with all bits on. On initialization the actual bit map bytes are set with all bits off or zero's.

For instance byte >38 controls sectors 1-8 as follows:

bit nos -	7	6	5	4	3	2	1	0
sector nos -	7	6	5	4	3	2	1	0

Byte >39 would be:

bit nos -	7	6	5	4	3	2	1	0
sector nos -	15	14	13	12	11	10	9	8

And so on.

To do the mathematics for finding which sector is controlled by which bit of which byte it is easiest to convert to decimal, do the math and then convert back to hex.

byte no - 56 = start of 8 sector group
>38 - >38 = 0 = sectors 0-7
56 - 56 - 0 - " "

sector no / 8 + 56 = byte no remainder * 8 = bit no
137 / 8 + 56 = 73.125 .125 * 8 = bit 1
34 / 8 + 56 = 4.25 .25 * 8 = bit 2 and so on.

When I started using Diags. and had gotten fairly familiar with the DOS, I often wondered how the Check Disk (CD) command kept track of sectors that are mapped bad. This is easy when the disk is freshly initialized as they are designated as being used in the bit map. But, I said to myself how is this done after several files are put on the disk and a good bit of the bit map is used up. How did it keep the bad sectors separated from those that were used. I looked at both a good disk and one with several bad sectors and could see no special coding, even on the track header data. Being curious I called M.G. and the people there were very courteous and helpful. When they told me how it is done I felt like an idiot for not thinking of it in the first place, so what's new. What happens is that after there are files on the disk, and if there are some bad sectors, the Diags. programming does a compare type operation. It checks the allocation bit map, and then checks the file directories for the sectors that the files occupy, and any sectors that are designated as used that are not included in the file directory information are considered to be mapped bad. neat huh?

In case any of you wondered how I got the neat Diags. display into the page without a lot of extra counting and typing, here is how. With the Output Device (OD) command you can select DSKn.filename and Diags. will dump the screen to a Display/Variable 80 disk file that can be merged into the TI-Writer file.

On several occasions people have heard me say that I have been using Diags, a good bit and that I have had great success with saving D/V 80 files that have developed bad sectors and the like. Some have asked that I do an article on this procedure, so here is a run-down on the procedures that I use.

The recovery of a text file is fairly easy because of two primary facts about the way the TI-99/4A stores data on a disk and that the text files are easier to decifer the data on the disk. The TI stores program files in reverse order, in other words, the beginning sector of the file block is actually the last sector of the program, with the last sector being the beginning of the program. This is due to the stack and first in is last out and all that. This will be explained at a later date. Text files and program files that have been saved with SAVE "DSK1.filename",MERGE format are not saved in the same format as regular program files. These are stored in normal sequence, first sector being the beginning and so on.

The recovery process is a bit involved, but becomes easier with practice and doing this and retyping eight or nine lines of text is much better than having to retype a five or six page file. For our sample file we will use the file from the TI-WRITER disk called PRACTICE1, as it is short and yet long enough for our purposes. Since Diags. uses decimal numbers to keep track of sectors and bytes, I will use the same format in this procedure to make it less complicated.

The first thing to do in preparation is to initialize a disk to SS/SD and then put the file PRACTICE1 on this disk. Next fire up TI-writer and select the Text Editor and type in the following data. Save this as Filename FILLER. This will almost exactly fill a sector.

```
NOW IS THE TIME FOR ALL GOOD MEN
**********************************
**********************************
**********************************
**********************************
THE QUICK SLY FOX JUMPED OVER THE
```

Now we can exit TI-Writer and load in Diags. After which we can do our file by-pass operation using the FILLER file as a patch or band-aid sector. Next use FIND FILE (incidentally most of the Diags. commands can be entered with two or so letter abbreviations). Type in FF PRACTICE1 and press <enter>. Your screen should look like this.

```
ADVANCED DIAGNOSTICS

          Status
Command : Find File

Drive   : 1     Track   : 0
Side    : 1     Sector  : 2
Free    : 0     H/S     : 10
Used    : 0     Bad     : 0


Start sec.  End sec.  Offs
    34         39       5
```

As you can see the file is seven sectors long, but has an offset of five. This means that there are actually five sectors of data. D Disk Manager will list it as seven because of sector 1 and the file directory sector. We also see that the file directory is on sector 2 in this case and that the file starts at sector 34 and ends on sector 39. (or sectors >22 and >27).

If you use FF to check on FILLER, you will see that it's directory is on sector three and is located on sector 40.

You will notice that there is always a difference of 7 between the sectors that are listed by cataloging and the offset.

Now just for curiosity, let's use FS 40 and take a look at our filler file.

Byte >10 is called the end of file offset. For variable length files and programs this byte lets us know the number of bytes in the last sector of the file are used. It also indicates which byte number is the EOF marker. For fixed length files this is always >00. The last byte of the last file sector is an end sentinel= AA for programs and FF for all other type files.

```
Drive  : 1      Track  : 0
Side   : 1      Sector : 6
Byte   : 0      Display: Hex

44454D4F2D312020202000000880 3
0006C2500600000000000000000
2C3000915000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
0000000000000000000000000
00000000
```

Byte >11 gives the logical record length. FIX or VAR 40=>28, 80=>50, 163=>A3, and 254=>FE.

Bytes >12->13 are the number of fixed length files or else the number of sectors in variable length files and are not used by programs. The bytes of this two byte block are reversed so that >0500 is actually >0005.

Bytes >14-1F are all zero's and are not used (reserved for future use). Does this possibly sound familiar?

Bytes >1C----These keep track of the blocks of sectors that the file actually occupies on the disk. This is done in 3 byte blocks and are not read as they appear in the block. Nybbles 4,1,&2 are the beginning sector and Nybbles 3,6,& 5 are the number of sectors occupied by that block of the file.

The following is an example of how to read these bytes in the case of a badly fractured file which is in five segments on the disk. This does not usually happen, but will hopefully show you how to read this block of data effectively.

Sector address (hex)	address contents (nybble) 12 34 56	start sector (nyb) 412	additional & prior sectors (nyb) 365	logical end sector	size of block of sectors	Subtot.
1C 1D 1E	23 30 00	023	003	026	4	4
1F 20 21	31 40 00	031	004	031	1	5
22 23 24	58 50 00	058	005	058	1	6
25 26 27	5A 10 01	05A	011	065	12	18
28 29 2A	67 60 01	067	016	068	d	23
2B 2C 2D	B4 80 01	0B4	018	0BE	2	25

Total data sectors = 25
Directory sector + 1
Cataloged sectors = 26

```
Drive   : 1      Track   : 4
Side    : 1      Sector  : 40
Byte    : 0      Display: Hex

204E4F5720495320544845205449
4D4520464F5220414C4C20474F4F
44204D454E212A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A212A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A212A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A212A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A20544520515549434B20534059
20464F58204A554D504544204F56
45522054484516809680D5BD9095
9FA953BDC7D5D5D5D5D5D5D5D58C
B6FF000000000000000000000000
0000000000000000000000000000
00000000
```

If you look at the end of the sector just before the block of zero's you will notice at byte 225 a value of FF. this is the end of file pointer. If you trace on back to byte 215, you will see a series of D5 bytes, these are unused margin and tab pointers. From there back to byte 202 is the margine and tab data for the file. Going back through the data of asterisk codes to byte 135 we see a 21 code, this is an end of line marker with no carriage return.

While doing the research for this article, I noticed that on sector 2 in byte 16, the number is always +1 greater than the offset. This is still one less than the catalog number, but adds the file descriptor to the sector count. The reason that the catalog number is +1 greater than ever this count is that it takes into consideration sector 1. Sector 0 is not counted because it is the master sector for the system and is always read.

Type in ES 37 and type in a series of random bytes. I used a series of >4A's then a few >21's and so on till the data block was almost full. Then use FCTN 9 to get back to Command Mode and type a W over the E to tell Diags. WS 40, this will simulate a trashed sector in your file.

Now let's type in ES 2 and enter to bring in sector two in order to make our changes in the file block data beginning at byte 28.

```
Drive   : 1      Track   : 0
Side    : 1      Sector  : 2
Byte    : 0      Display: Hex

505241435449434553120000008803
00064E50060000000000000000000000
222000283000265000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
00000000
```

As we see, the file started at sector >22 with an offset of 5. The three byte block should read 225000. Now we need to change the 5 to a 2. In the next two blocks type in 2B3000265000; Now the data should read thus 2220002B3000265000 as in the sample at the left, this tells that the first block of the file starts at sector 34 with an offset of 3 keeping in mind that hex numbering starts with 0. then is our filler of one sector beginning at sector 40 with an offset of 1 bringing the subtotal to 3. The final block starts at sector 38 with an offset of 2 making a total offset of 5.

Now use FCTN 9 to get back to Command Mode and type W over the E to write the sector to the disk.

We can now go to TI-Writer and load our patched file. It should load fine and you should see your filler data about the middle of the file. As you can see, you would only have to retype about 6 or 7 lines of the file.

Hopefully the information contained in this article will be of use to new and old users alike. Usually the information set forth here must be accumilated from several sources. Hopefully it will be of value due to the fact that it is all contained on a few pages here.