SAN DIEGO COMPUTER SOCIETY

TI-SIG NEWSLETTER

FEBRUARY 1989

BY THE TIME MOST OF YOU receive this newsletter, the TI FEST WEST will be a thing of the past. Let us hope that it left a very pleasant memory! Our thanks go to all the wonderful user groups and TIers everywhere who helped.

BITS AND BYTES

Roland Anderson called my attention to an article in the Nov. 1988 "BUG BYTES" from the Brisbane User Group. No author was given, but the title was "USING THE SCREEN AS A FILE." Here is a short program using some of the ideas from the article. This program allows you to PRINT to either the screen or printer.

```
100 REM SCREEN OR PRINT. W.A
WILSON, TI-SIG, FEB 1989
110 CALL CLEAR
120 LINPUT "ENTER A SHORT SE
NTENCE: ":A$
130 CALL CLEAR
140 DISPLAY AT(3,3):"PRESS 1
) FOR SCREEN DISPLAY       2
) FOR PRINTED COPY"
150 CALL KEY(3,K,S):: IF S<1
 THEN 150 :: IF K<49 OR K>50
 THEN 150
160 CALL CLEAR
170 OPEN #1:"PIO"
180 IF K=49 THEN @=0
190 IF K=50 THEN @=1
200 PRINT #@:A$
210 DISPLAY AT(12,3):"PRESS
ANY KEY.  ""Q""=QUIT"
220 CALL KEY(3,K,S):: IF S<1
 THEN 220 :: CLOSE #1 :: IF
K=81 THEN END ELSE 110
```

The key to the entire program is in line 200. If you print to "0" you go to the screen, but print to the same file number as in line 170 and you will go to the printer. If you are working on a program that stores data to a file, you can save paper by printing to the screen provided your material is of the DISPLAY type. It may be difficult to interpret the display if file format is INTERNAL.

==========

HERE IS A SHORT PROGRAM to compute the sales tax that is due when the tax is included in the selling price.

```
100 ! SALES TAX COMPUTATION
W. WILSON, TI-SIG, FEB 1989
"SUB ROUND" FROM J. PETERSON
110 CALL CLEAR
120 DISPLAY AT(3,3):"SALES T
AX COMPUTATION WHEN  SELLING
 PRICE INCLUDES 7%   SALES T
AX"
130 INPUT "SELLING PRICE ":S
P
140 CALL CLEAR :: COST=C ::C
=SP/1.07 :: R=2 :: CALL ROUN
D(C,R)
150 IMAGE SELLING PRICE IS
 $###.##  TAX IS
   ##.##  COST IS
  ###.##
160 PRINT USING 150:SP,SP-C,
C :: PRINT :: GOTO 120
170 SUB ROUND(N,R):: X=INT((
((10 R)*N)+.5))/(10 R):: N=X
 :: SUBEND
```

==========

REPORT FOR THE SDCS

Once again only six members attended the January 17th meeting. This time we can blame the flu for the absentees.

Prior to the business meeting, Woody demoed Ray Kazmer's new program, "MAZE OF GROG". This was Ray's 1989 Valentine Day Card to the TI world. (Another Ray Kazmer winner!)

Woody reported that the treasury balance was $268.94. The group then voted to donate $105.00 to the TI FEST WEST to be used to defray vendor's room expenses.

Waldo Hamilton gave a demonstration of how to inspect printed circuit boards for likely sources of trouble. He used Mike Beach's RS232 card.

Woody then demoed Miller Graphics' "EXPLORER" program using his "$3.99" TI console. Group was impressed with how the program was slowed down about a 1000 to 1 ratio so every step of the program could be viewed.

"XB BUG" by J. Peter Hoddie then was shown and Woody told how he had used it in debugging his "CATDISK" program which has over 2500 lines.

The problem with the newsletter distribution was discussed. Woody and Roland Anderson are to bring in recommendations at the next meeting.

Roland is to bring the system for the February meeting.

THIS MONTH WE CONTINUE with the third article by Tony McGovern on how to write assembly language programs that work properly with FUNNELWEB.

==========

LIVING WITH SPIDERS

Tony McGovern

PART III

The previous installments have been on "dual-mode" programs which have to run with or without FWB. Now we will look at a few aspects of writing programs to work only with FWB.

The first thing is access to GPL routines. If the FWB E/A utilities are loaded then a normal BLWP call to GPLLNK works as usual. The various XML address possibilities should be repect- ed, particularly the >F0 XML at >8300 which is used by the TI-Writer and and Mini-mem modules, or from console GROMs. If the E/A utilities are not loaded a briefer than normal form of GPLLNK may be used. The following routine is used to call the beeps and bloops in DPatch.

```
*  FWB system equates

SETGRM EQU  >FF28
MODFL  EQU  >FF5A

*  GPLLNK BLWP vector

GPLLNK DATA GPWS,GPLK

GPLK   EQU  $
       MOV  @MODFL,R0
       LI   R1,GPLRT
       MOV  R1,*R0
       MOV  *R14+,@>83EC
       LWPI GPLWS
       B    @>0060      Go to GPL

GPLRT  LWPI GPWS
       MOV  @SETGRM,R1
       BL   *R1
       RTWP
```

Here the XML vector has already been determined by the FWB entry code and its target address stored at MODFL (>FF5A). If the E/A GROM is present MODFL will contain the value >2002 for XML >21 as example. The re-entry address is GPLRT is written into the XML address, without bothering to save the existing contents. If you do use >8300 you had better save and restore it though. The GPLLNK data item is written to R6 of the GPL workspace, and then transfer made to GPL. When the linkable GPL routine has finished it returns to GPLRT. SETGRM (>FF28) contains a pointer to a BL routine in FWB which resets all the GPL items. This routine uses only R0. One subtlety is that the GPL stack pointer at >8373 is reset to >80 by the SETGRM routine. The FWB cartridge loader from UL sets it to >7E to help with key-unit problems apparent in some programs. There is no reason why it has to be a LWP routine and the code could be recast in other forms. FWB contains a similar routine to handle cassette loading, but it is in-line code and not externally accessible.

FWB does however contain an externally accessible DSRLNK with BLWP vector at >FFD4. This routine takes no following data item, and returns with the status (equals) bit set if not found and it is otherwise up to you to dig out any error from the PAB or GPL status byte. This means that it is basically set up for file type access only. As an example here are the load and resave routines from the CF/CG configuration program,

```
FILSVE EQU  $
       MOVB @SAVEOP,R1
       JMP  FILOPS

FILOAD EQU  $
       MOVB @LOADOP,R1

*  Load-save routine

FILOPS EQU  $
       LI   R0,PAB
       BLWP @VSBW
       LI   R0,PAB+9
       MOV  R0,@>6356
       SB   @GPLST,@GPLST
       BLWP @DSRLNK
       JEQ  FAILS

       BLWP @VSBRD
       DATA PAB+1
       SRL  R0,13
       JNE  FAILS

       MOVB @GPLST,R0
       COC  @MASK,R0
       JEQ  FAILS

       INCT R11
FAILS  RT

MASK   DATA >2000
LOADOP BYTE 5
SAVEOP BYTE 6
```

Alternate entry points allow for save or load operations. It is assumed that the PAB and VDP buffers are already set up as needed. I'm never quite sure if the clearing the GPL status byte is strictly necessary but it can't hurt. The exit after the DSRLNK call is taken if the DSR is not found. This particular program uses a data VSBR to fetch the PAB error byte into R0 from fixed address in VDP. A normal VSBR would allow a more flexible routine. If no error is indicated here after the first 3 bits have been isolated, the GPL status byte is then checked. The failure exit steps over the word following the BL call. Normally this would be a JMP to an error handler.

Suppose your program needs to do a sector access via a DSRLNK. You could of course use the E/A utility routine which takes a following data item. Just to be different we'll work through the sector reading routine used in QD. The E/A routine is not available so the FWB DSRLNK has to be modified to suit. This is done by temporarily rewriting an internal item from >8 to >A on entry and restoring it on exit. The routine also illustrates the FWB method of coping with high-CRU Horizon RAMdisks. The problem here is that the sector read/write routine has of necessity the same name >10 in all disk DSRs and if the drive number doesn't match an error is returned. This means that a normal DSRLNK won't get past the usual disk controller at CRU >1100 (the Myarc RAMdisk at CRU >1000 handles this transparently for normal disk controller accesses). Some programmers solve this problem by using a DSRLNK which starts its CRU search at >1200 which finally loops around to check >1000 and >1100. Atrax Robustus doesn't care for this method as it puts a permanent slug on normal disk access, and insisted on doing it another way which will become apparent.

In the code to follow, extracted from the QD source, only sector reads are performed and any non-null data byte will do for READ. The style of coding reflects a situation where registers are in short supply and length is critical. The drive number is at DR#, the VDP sector buffer address at PBF, and the sector number in R7. The code uses routines available in QD which in turn were written to match TI-Writer routines used by SD. You could do things differently if you chose. The offset values >56, >5C, and >7A refer to particular address offsets into the FWB DSRLNK routine. These will be maintained at their present values, even though I am continually resisting temptation to shave some bytes out of the DSRLNK.

```
DSRLNK EQU  >FFD4

FAC    EQU  >834A
SCNAME EQU  >8356

PN     BYTE >02,>01,>10
BOF    BYTE >0F
BIF    BYTE >0F
CTRLC  BYTE >B3
       EVEN

SECRD  EQU  $
       MOV  R11,R10
       MOV  @DSRLNK+2,R11
       INCT @>7A(R11)
SECRN  LI   R9,PAB
       MOV  R9,@SCNAME
       LI   R8,PN
       BL   @VSTRW
       LI   R11,FAC+2
       MOVB @DR#,*R11+
       MOVB @READ,*R11+
       MOV  @PBF,*R11+
       MOV  R7,*R11
       BLWP @DSRLNK
       JEQ  DSERR

       MOVB *R11,R12
       JEQ  DSREX
```

(Continued on page 3)

(Continued from page 2)

```
DSERT   MOV   @DSRLNK+2,R11
        A     @>5C(R11),@>56(R11)
        CB    @>56(R11),@B1F
        JL    SECRN
DSERR   EQU   $
        BL    @VSTRU
        DATA  ERMSC
        DATA  >2E6
HOLD    BL    @KSCAN
        CB    R12,@CTRLC
        JEQ   HLD10
        CB    R12,@B0F
        JNE   HOLD

HLD10   LI    R10,EXIT
DSREX   MOV   @DSRLNK+2,R11
        DECT  @>7A(R11)
        MOVB  @B0F,@>56(R11)
        B     *R10
```

On entry the return is saved in R10, and R11 used as a working register thereafter. The DSRLNK is then fetched from the BLWP vector and the word offset >7A into the routine is INCTed. This makes it equivalent to a DATA >A call to an E/A DSRLNK. We must then be careful to restore the value at every possible exit from the routine. VSTRW is a BL routine that writes a string with leading length byte from CPU to VDP. While at it we load the PAB pointer at >8356. The next few lines load the FAC area with all the necessary data for a sector read call and leave R11 pointing at >8350 where errors are returned. If the DSRLNK search finally fails, or has found the drive number but can't read the disk, the JEQ sends it to the exit error routine.

The DSR sector routine error byte is then checked. If this is null the sector read has been successful and a normal exit is taken. If it is not null then the sector routine has been executed but the drive number wasn't found at that CRU base and the special error handler at DSERT is entered. This resets R11 to the start of the DSRLNK code and adds the normal increment of CRU base search at offset >5C to the start address for the CRU search. The sector read routine is then re-entered and the whole process started again, but this time from the next CRU slot, so that the last error isn't repeated. Before this repeat is done the CRU base is checked to see if it has reached the end of the line. If so the normal error exit is taken. The virtue of this method is that is the error path that cops the penalties and not your Myarc RAMdisk at >1000 CRU base. I have also heard that it works better with multiple Horizons wit ROM DSRs.

The error exit writes up an error message using an in-line data entry point to the VSTRW routine and then waits for either <fctn-9> or <ctrl-C> to be pressed. The final return address is re-written in R10 as needed. All exits pass through DSREX where the CRU base search start is re-initialized and the DSR type reset for normal file access. In some applications it might be preferable not to reset the CRU starting base immediately but this leads to messier code overall. A good example is right here in in this very disk directory application where the same CRU base could be used until a whole directory worth of file header sectors had been read. In fact this is done in the Editor's SD which uses generally similar code but it really only affects high-CRU Horizons.

Similar problems exist for volume name access to files where DSK. is also a DSR routine name common to all disk type DSRs. As yet the FWB main program code does not contain the special error handling routines as in FWB sector access code as there appears to have been no specific demand for volume-name access to high-CRU Horizon type RAMdisks. Most (the TI-Workshop cartridge is an exception) file-loader DSRLNKs don't support it either. Incidentally you can observe the other type of enhanced DSRLNK in action if you have high-CRU Horizons installed. Watch the lights when DSKU (or TI-Workshop or Ottawa issue DM-1000) is accessing low CRU devices, say a Myarc RAMdisk, or regular disk controller. The high CRU Horizon lights will be flashing, but not with similar access by a FWB system program. In this same vein if you catalog a high-CRU Horizon with QD the regular disk controller light will flash with QD but not with SD after the first sector has been read.

NOTES FROM WOODY' DESK:

IN THE JANUARY 1989 newsletter of the LA99ers User Group we were reminded that Texas Instruments has dropped its TI-CARES phone number and now furnishes into via the following numbers:

800 747-1882 General Information
800 747-2662 Technical Assistance
800 747-2265 Dealer Parts
-2268 " "

==========

PAGES 4 AND 5 HAVE A list and order form for Jim Peterson's new TI-PD (public domain) software. The nine page catalog for $1 is a bargain even if it were not refundable with your first order. Let's order a lot and show our appreciation!

TI-PD public domain software for the TI-99/4A computer, $1.50 per disk postpaid (minimum 8 disks please); number of sectors filled is indicated in parentheses). For a 9-page catalog listing all titles and authors, send $1 which is deductable from first order. (specify TI-PD catalog) Offered as a copying service only, without warranty other than that copies are equal to the original. Make checks payable to Tigercub Software (no credit card orders). Send to Tigercub Software, 156 Collingwood Ave., Columbus OH 43213.

600. Sam Moore Jr. Music #1 (341)
601. Sam Moore Jr. Music #2 (343)
602. Sam Moore Jr. Music #3 (348)
603. Sam Moore Jr. Music #4 (337)
604. Bill Knecht Hymns (334)
605. Christmas Music (318)
606. Holiday Music (339)
607. Great Songs by Bill Knecht (351)
608. Music by Bill Knecht (295)
609. March Music (329)
610. Tigercub Country Music (356)
611. Christmas Sing-Along (351)
612. J. Stephen Foster Music #1 (332)
613. J. Stephen Foster Music #2 (317)
614. Bach Music Programs (338)
615. Sing-Along Music (351)
616. Some of the Best Music (343)
617. Classical Music (340)
618. Assorted Music (346)
619. Chopin's Polonaise (280)
620. Assorted Music #2 (351)
621. Hamilton UB Music Package #1 (346)
622. Assorted Music #2 (349)
623. A Diskfull of J.S. Bach (345)
624. Assorted Music #4 (350)
625. Assorted Music #5 (347)
626. J.S. Bach Music (340)
627. Assorted Music #6 (354)
628. Some of the Very Best (349)
629. Ollie Hebert's Music (340)
630. Gregory Rashall Music Master (287)
631. Assorted Music #7 (352)
632. Sonata for Pianoforte (222)
633. Sonata for Pianoforte DS/SD
634. Strange Music (337)
635. Chuck Berry Tunes (218)
636. Christmas Songs w/Graphics (310)
637. Assorted Music #9 (337)
638. Classical Music #2 (338)
639. Assorted Music #10 (347)
640. Marches and College Songs (336)
641. Another Sing-Along (345)
642. Sing-Along Music #2 (236)
643. Christmas Music #2 (351)
644. Christmas Sing-Along #2 (340)
645. Classical Music #3 (352)
646. Assorted Music #11 (350)
647. Music Doodlers and Tinytunes (302)
648. Rhapsodie in Blue (287)
649. Assorted Music #8 (354)
650. Christmas Music w/Graphics 2 (357)
651. Sorgan II (145)
652. Christmas Music w/Graphics 3 (352)
653. Pop Demo V1.1 (225)
654. Christmas Music w/Graphics 4 (255)
655. Assorted Music #12 (349)
701. Musical Education (350)
702. Musical Education #2 (318)
703. Musical Education #3 (188)
710. American Flags (360)
711. Flags of the World (345)
712. Geography - U.S. States (341)
713. Geography - U.S. States #2 (212)
714. World Geography (179)
730. American History (48)
750. Alphabet w/Speech (343)
751. Children's Programs w/speech (357)
752. Alphabet for Preschool (329)
753. Children's Prog. w/Speech #2 (335)
755. Shapes, Colors, Directions (173)
760. Spelling (324)
770. Vocabulary and Reading (293)
780. Preschool Math (341)
790. Elementary Addition, Subtract(257)
791. Addition & Subtraction (337)
796. Multiplication, Division (348)
797. Multiplication, etc. (224)
800. Higher Math (355)
801. Higher Math #2 (228)
810. Typing Practice (223)
815. Morse Code Teacher (155)
820. Health (354)
821. Health #2 (145)
830. Physics (111)
840. Nature (277)
850. Chemistry (277)
860. Astronomy (342)
861. Astronomy #2 (304)
870. Religion (346)
871. Religion #2 (42)
890. Teacher's Helpers (203)
900. Home Utilities (351)
901. Home Utilities #2 (342)
902. Home Utilities #3 (350)
907. Screen Drawing, Doodling (160)
909. High-Resolution Drawing (287)
910. Charts & Graphs (178)
912. Calculators & Converters (345)
913. Calculators & Convert. #2(147)
915. Financial Math (339)
916. Financial Programs (356)
918. Checkbook Programs (203)
920. Business Programs (146)
950. Genealogy
970. Astrology, Numerology etc. (171)
980. Radio Utilities (220)
990. Sports Programs (329)
1100. Character & Sprite Editors(254)
1101. Programmer's Utilities (346)
1102. Sorts, Scrambles, Searches (228)
1105. Auto-loaders (217)
1106. Disk Catalogers (268)
1107. Character Sets etc. (353)
1110. Assembly Utilities (357)
1111. Assembly Utilities, Routines(328)
1112. New Horizon Assembly Util. (269)
1119. Hardware Utilities (169)
1120. Sound Effects (197)
1130. Disk Labels & Jackets (284)
1131. Gemini Printer Utilities (224)
1132. Word Processing Utilities (182)
1133. Banners, Graphs, etc. (203)
1135. Speech Utilities & Demos (355)
1140. Music Composers (288)
1141. Assembly Music Compiler (265)
1145. Telecommunications Aids (342)
1150. Programming Tutorials (348)
1160. Assembly Tutorials #1 (231)
1161. Assembly Tutorials #2 (289)
1162. Assembly Tutorials #3 (357)
1163. Assembly Tutorials #4 (358)
1164. Assembly Tutorials #5 (340)
1300. Mathematical Games (133)
1301. Brain Games #1 (344)
1302. Brain Games #2 (345)
1303. Brain Games #3 (352)
1304. Brain Games #4 (352)
1305. Two-Player Brain Games (335)
1306. Brain Games #5 (345)
1307. Master Mind (322)
1310. Memory Games (235)
1315. Sargon Chess (155)
1320. Mazes #1 (342)
1321. Maze Games #2 (346)
1322. Maze Games #3 (338)
1330. Hangman Games (335)
1331. Wheel of Fortune #1 (249)
1332. Wheel of Fortune #2 (248)
1333. Word Games (310)
1340. Games by Roland Trueman (333)
1350. Card Games #1 (352)
1351. Card Games #2 (340)
1352. Card Games #3 (94)
1356. Dice Games (354)
1360. Board Games (321)
1361. Bingo (73)
1362. Checkers (238)
1363. Board Games #2 (287)
1367. Gambling Games (237)
1381. Bowling (289)
1382. Golf (138)
1383. Billiards, Boxing, etc. (250)
1400. Adventure Disk #1 (360)
1401. Adventure Disk #2 (306)
1402. Adventure Disk #3 (329)
1403. Adventure Disk #4 (324)
1415. Hamaurabi Games (268)
1416. Text Games #1 (313)
1417. Text Adventures (340)
1425. Graphics/Text Adventures (354)
1426. Graphics/Text Adv. #2 (322)
1427. Graphics/Text Adv. #3 (325)
1430. Road Race Games (356)
1431. Keyboard Maneuvering (349)
1432. Road Crossing Games (344)
1433. Road Crossing Games #2 (175)
1434 Keyboard Games (347)
1435. Keyboard Maneuvering #2 (354)
1436. Slot Machines (343)
1437. Keyboard Games #2 (353)
1438. Keyboard Games #3 (343)
1440. Q*Bert Games (288)
1445. King Kong Type Games (351)
1455. Assembly Games (231)
1456. Assembly Games #2 (346)
1460. Children's Programs (345)
1461. Fun Games for Kids (353)
1462. Easy Games for Kids (346)
1470. Great Games (342)
1471. Assorted Games #1 (348)
1472. Assorted Games #2 (343)
1473. Texas Games Medley w/speech(346)
1474. Sea Battle Games (329)
1475. Joystick Games (342)
1476. Joystick Games #2 (355)
1477. Joystick Games #3 (346)
1478. Joystick Games #4 (338)
1479. Two-Player Joystick Games (353)
1480. Two-Player Keyboard Games (353)
1481. Joystick Games #5 (345)
1500. Kaleidoscopes & Displays (262)
1501. Sprite Displays (200)
1505. Poetry, Prose & Nonsense (128)
Ñ

SAN DIEGO COMPUTER SOCIETY

TI-SIG

NEWSLETTER FEBRUARY 1989

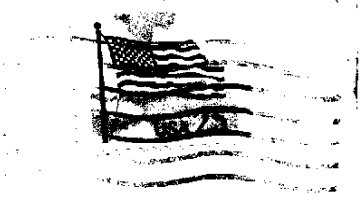
TI-SIG is a special interest group of the San Diego Computer Society. It is not sponsored by or affiliated with Texas Instruments, Inc. We acknowledge all copyrights and other legal rights of the companies/products mentioned herein. Such references imply neither an endorsement nor condemnation of these companies or their products, but are the opinions expressed by the authors and do not necessarily agree with those of the TI-SIG or its officers. Articles from this newsletter may be reprinted with proper credit to the author or TI-SIG.

Meetings: 3rd Tuesday of each month at 7 P.M.
Crafts Room, North Park Rec. Center, 4044 Idaho Street, San Diego

1988/89 OFFICERS

President: John Johnson 279-9740	Vice Pres: Woody Wilson 264-6515
Treasurer: Woody Wilson 264-6515	Secretary: Lutz Winkler 277-4437

TI-SIG
San Diego Computer Society
P.O. Box 5263
San Diego, CA 92105



DALLAS TI-HOME COMPUTER GROUP
P.O. BOX 29863
DALLAS TX 75229