8703 (094)
VAST

50¢

VALLEY OF THE SUN TI 99 USERS GROUP newsletter

VOL.3 MARCH 14,1987 NO.3

****

CONTENTS

VAST 99 BBS 437-4335

---------------------------------MARCH,1987---------------------------------

VAST 99 INFORMATION

The **VAST 99 USERS' GROUP** is a support group for TI 99 Home Computer users. We meet on the second Saturday of the month at the Los Olivos Resort Motel in the "Phoenix" room at 202 E. McDowell Road (about a block East of the Library). The meetings start at 10:00 AM and continue until 11:00 AM with socializing starting at 9:00 AM. The yearly membership fee is $6.00.

All meetings are open and anyone may attend. Only dues paying members may vote in elections and obtain programs from the Users' Group library.

The current officers are:

President	
Gerry Kennedy	992-7668
Vice-President	
Doug Otten	973-7768
Secretary	
Mike Marfisi	897-8280
Treasurer	
Ike Van Kampen	934-5164
User Group Librarian	
Earl Bonneau	269-3802
Newsletter Editor/BBS SysOp	
Jim Ely	437-1796

****

A FORTH Tutorial is being conducted by Rene' LeBlanc in this newsletter. It consists of a continuing series of articles relating to his version of FORTH which is available from the User Group Library. For more information, please contact him at (602) 991-1403.

The Users' Group's BBS is now in operation 24 hours a day. Contact it at (602) 437-4335. There is a lot of interesting conversation and information available here so give it a try.

Deadline for submission of articles or advertising for the Newsletter is the last Saturday of every month. Articles may be submitted in any form, however, the preferred method is by phone transfer directly to the Editor.

****

Advertising_rates_are_as_follows:

Commercial:

Full Page $10.00
Half Page $ 7.00
Quarter Page $4.00

Personal:

Four lines,
30 Characters/line
$1.00
$.20 per line
over four.

All rates are for **ONE** issue only!

****

Programs are available from the USERS' GROUP LIBRARY at the following rates:

SS/SD Disk	$2.00
SS/DD Disk	$4.00
DS/SD Disk	$4.00
DS/DD Disk	$8.00

If copying of documentation is required, it will be at the rate of $.10 per page. If the User Group supplies the disk, please add $1.00 to the above charges. An exchange program for free programs is also in effect. Please contact the librarian for further information.

****
*   *
* VAlley of the Sun TI99 Users Group *
*   *
****

MARCH, 1987

From the Editors Desk

M I N U T E S
for
February 14, 1987

The February meeting of the VAST 99 TI User Group was held on Saturday Feb. 14 at the Los Olivos Resort Hotel. The meeting was called to order by President Gerry Kennedy at 10AM.

No formal business or voting was conducted.

Gerry updated the group on the Myarc computer and gave out some other "news" of the TI world. Stu Olson announced that PC Pursuit has started operation in the Phoenix area. PC Pursuit is a long distance phone service that allows customers to call all over the country for purposes of telecommunications. Rather than paying individual long distance charges, members are charged a flat monthly fee for the service. Both Stu and Jim Ely reported that we are starting to get out of town calls on both BBS now.

Mike Marfisi read from a letter from the RAVE Co. in response to our inquiry about special pricing for members on the new RAVE keyboard. They said that they would consider a group price on an order of over $1000. They also announced that the 2 models of the keyboard currently available would be discontinued in favor of a new and better model, the 99/105 on April 1. The company is also offering an option where one can purchase the interface only, either assembled or in kit form. This means the buyer can then choose his own IBM style keyboard from any other source.

Mike then pointed out that our newsletter exchange program is going well. We are getting almost 50 newsletters per month. They have been placed in binders and are available for members to borrow or brouse through.

The meeting ended at 11:30. The next meeting will be held, same time, same place on Saturday March 14th.

Please remember that a new slate of officers will need to be elected soon. Think about making a time commitment. Money helps but people are more important.

T h e E d i t o r s a y s :

***SYNTAX ERROR**

That's what you might get if you typed in last months FORTH Screen. It seems that a TI-Writer transliterate command got eliminated from the text of the article and the pound/number sign (#) that appeared in several places in the screen should have been a circumflex (^) (Did you know that little symbol was called that?). So, if you change all the (#)'s to (^)'s everything should be OK and work normally. Sorry for the error, Rene'.

NEWS... (Or lack thereof)

This past month has been a real sloooooow month. Last month, a lot of promises were made about a number of new hardware items for your TI. Since those announcements, the manufacturers have been sooo quiet, you could hear a pin drop! Did Myarc ship the first 9640 computers as planned? How about the Hard disk controller cards? What

CONTINUED NEXT PAGE-=>

--------------------------------MARCH,1987--------------------------------

Editors Desk Continues

about the Triton Turbo XT? Has anybody seen anything from anybody? Is anybody out there? From all the silence, I guess you could draw the conclusion that nothing happened as planned or promised. Good going, Lou. You did it to us again. The one thing that we all forgot to ask again was WHAT YEAR? Maybe we will hear something next month.

THIS ISSUE...

It started out rather slow and then everything sort of hit last week. So, on page 5 is a short article from the "Dallas 99 Interface" newsletter on an interesting concept called **TINYGRAMS**. **WHEREFORTHS** this month starts a series on a disk utility program written in **FORTH**. The complete program is presented this month and the explanation on how it works will start in next months issue. The program starts on page 6. Page 8 is Computer Tutor. This month we start a 2 part series on basic sort routines. On page 9 is part 2 of our series on TMS9900 Assembly Language comparing X-basic display speed to Assembly speed. And last, but not least, on page 11 is a **QUIK TIP** relating somewhat to Assembly Language.

There you have it!

N O M I N A T I O N S

Don't forget that at today's meeting, we will be holding nominations for new officers of our group and next month we will hold our elections. If you think you might be interested in holding a position on the group's governing board, please speak up. We could use **YOUR** input.

A SMALL REVIEW

One last thing. You will notice that the Editor's page is printed a little differently this month. I am using the new version of **CSGD** (Version 3) by **Dave Rose**. Double column printing and 6 different type styles for text printing are possible among various other options. This is an amazing software package and I hope to have a review of it in an upcoming issue. Stay tuned...

Have a nice St. Patrick's Day! See you all next month.

Jim Ely,
Newsletter Editor

```
 ***********************
<                       >
<     THE  VAST  99     >
<                       >
<         BBS           >
<                       >
<       ON  LINE        >
<                       >
<   24  HRS.  A  DAY    >
<                       >
<         FOR           >
<                       >
<     INFORMATION       >
<                       >
<    COMMUNICATION      >
<                       >
<    ENTERTAINMENT      >
<                       >
<         AND           >
<                       >
<   JUST PLAIN FUN      >
<                       >
<       C A L L         >
<                       >
< (602) - 437 - 4335    >
<                       >
 ***********************
```

-----------------------------MARCH,1987-----------------------------

ODDS AND ENDS

TINYGRAMS

Mike Stanfill writes for the DALLAS TI HOME COMPUTER GROUP and his articles appear in their newsletter, the "DALLAS 99 INTERFACE." He is the creator of a program format he calls "TINYGRAMS". The program format has only one rule: the program must not exceed 24 screen lines in length. (i.e. When "listed", the entire program does not scroll off the screen (Title and/or remark lines are not included)). Mike's main purpose is to try to get you to write compact and efficient code. If you can make a simple program that short, then, when you do more complex programs, you can still apply the same thought processes.

Jim Peterson of Tigercub Software is another master of short and consise code. He has written many "TINYGRAMS", many as merge type sub-programs. He has also done many "One Liners", programs that are only "one" line long and do wonderful things!

The following two "TINYGRAMS" are from the October, 1986, issue of the DALLAS 99 INTERFACE. If you plan to type in these programs, there is one very important "trick" you will need to know: A **normal** XB program line is only 5 screen lines long. When you reach this point, you can't enter any more characters. However, you will notice that at least one program line in each of these 2 programs is longer than 5 screen lines! (How'd he do that?) Ah, yes. The "trick"! After you have gone as far as the computer will allow you, press <ENTER>. Now do <FUNCTION 8> (REDO) and the line is relisted. Now, move the cursor to the end of the program line (using <FUNCTION D> (right arrow)). Now, just continue entering the rest of the line. Press <ENTER>. When you go back and list the program, you will find that, indeed, the program line exists at greater than 5 screen lines long! Jim Peterson uses this "trick" often in his "One Liners".

The Tinygram "TINYCAT" is a program for listing disk contents from any disk drive 1-4, including your RAM disk if it is designated as one of the four drives.

```
1 !********TINYCAT*********
   ******A TINYGRAM********
   *****BY JOHN GUION******
   ************************
2 T$(1)="Dis/Fix" :: T$(2)="
Dis/Var" :: T$(3)="Int/Fix"
:: T$(4)="Int/Var" :: T$ (5)
="Program" :: DISPLAY ERASE
ALL:"Disk? (1-4)"
3 J=J-1 :: CALL HCHAR(23,15,
30+(J=0)*-19):: CALL KEY(0,N
,S):: IF N=13 THEN N=49 ELSE
 IF N<49 OR N>52 THEN 3
4 CALL HCHAR(23,15,N):: OPEN
 #1:"DSK"&CHR$(N)&".",INPUT
,RELATIVE,INTERNAL :: INPUT
#1:A$,J,J,N :: PRINT "Diskna
me=":A$:"Available=":N:" Use
d=":J-N+2:"Filename  Size
 Type      P":RPT$("-",28)
5 INPUT #1:A$,I,J,N :: IF A$
="" THEN STOP ELSE PRINT :A$
:TAB(11):J:TAB(17):T$(ABS(I)
):: IF ABS(I)<>5 THEN PRINT
 N:
6 IF I>0 THEN 5 ELSE PRINT T
AB(28):"Y":: GOTO 5
```

The Tinygram "CAMEL" is a game based on the old saying, "The straw that broke the camel's back."

```
1 !********CAMEL********
   *****A TINYGRAM******
   **BY MIKE STANFILL***
   *********************
2 CALL CLEAR :: Q$="55767671
353235" :: K=-1 :: CALL COLO
R(10,16,7,2,11,11):: B,Q=0
3 P=2 :: W=INT(RND+9):: F
OR T=1 TO 7 :: CALL VCHAR(VA
L(SEG$(Q$,T,1))+5,T+12,42,VA
L(SEG$(Q$,T+7,1))):: NEXT T
:: FOR X=1 TO 7 :: FOR Y=15
TO 19 :: CALL SOUND(1,-5,0)
4 IF Q=0 THEN P=P+K :: K=-K
5 Z=11-X :: IF Q=0 THEN DISP
LAY AT(20,2):"GUESS?":"#1=":
R(1):"#2=":R(2):: ACCEPT AT(
20+P,8)SIZE(1)BEEP VALIDATE(
"123456789"):Q
6 F=P+K :: B=B+1 :: CALL HCH
AR(2,Y,111):: IF B>W THEN 8
7 Q=Q-1 :: NEXT Y :: NEXT X
8 DISPLAY AT(18,2):"#":F:"WI
NS!" :: R(F)=R(F)+1 :: FOR J
=5 TO 10 :: CALL HCHAR(J,15,
32,5):: CALL SOUND(599,440-1
0*J,0):: CALL HCHAR(21-J,15,
111,5):: NEXT J :: GOTO 2
```

If you enjoy these and would like to see more, let me know or better yet, make your own "TINYGRAM" or "ONE LINER" and submit it for publication here. You can send them to the Editor, VAST 99 Newsletter, 4144 E. Nancy Lane, Phoenix, AZ 85040, or you can upload them to the VAST BBS (602-437-4335) with a note in the description that it is for submission to the newsletter. All submissions will also be forwarded to Mike Stanfill to be added to his collection.

ENJOY!
J. E.

----------------------------------MARCH,1987----------------------------------

WHEREFORTHS OF FORTH

In past issues of WHEREFORTHS we have been building up a number of small utility words which are not sufficient as stand-alone programs in themselves. Rather, they are potential pieces of programs. Beginning with this issue, I will begin using some of those pieces to write a complete useful program.

Several times I have encountered the problem of wanting to make a disk copy of a Forth disk for someone when the only system available had only a single disk drive. The FORTH-COPY word in TI-Forth only works with two drives. Forth disks needing a sector-by-sector copy to ensure that the sector locations don't change make use of the TI Disk Manager a risky proposition. So, I decided a little sector-by-sector disk copy program would be a nice project for our WHEREFORTHS articles.

Unfortunately, this program is too large to completely cover all in a single issue of WHREREFORTHS. Rather than give you just a few screens of the program at a time with a complete explanation, but have only useless fragments of the entire program until a few months later, I decided to put a first pass of the entire program in the first issue, but use subsequent WHEREFORTH issues to explain the design process and the program details. This way, you can key this program all in as soon as you get home, and have an adventure trying to get it working. If you fail, you will develop lots of questions along the way, trying to debug the thing, and this will be a vital experience. Also, the next few issues of WHEREFORTHS will be vital to you because they will explain things about this to you.

I have tested the program as presented, and it works. In future issues, we can make some further enhancements to it.

```
+--------------------------------------------------------------------+
: \ Disk Copy Program SCR#1                                          :
:  TEXT                                                              :
:  : INSTRG ( -- )                                                   :
:    IOBUF 1+ 40 EXPECT PAD 40 1 DO DUP  I + C@ 0=                   :
:    IF I 1- SWAP C! BL PAD COUNT + C! LEAVE THEN LOOP ;             :
:                                                                    :
:  : (C,R)OK? ( ? -- f )                                             :
:    DEPTH 2 < IF 0 ELSE 2DUP 24 < SWAP 39 < AND THEN ;              :
:                                                                    :
:  : ACCEPT_AT  ( col row -- ) (C,R)OK? IF GOTOXY THEN INSTRG ;      :
:                                                                    :
:  : DISPLAY_AT ( $addr col row -- ) GOTOXY COUNT TYPE ;             :
:                                                                    :
:   -->                                                              :
+--------------------------------------------------------------------+
```

```
+--------------------------------------------------------------------+
: \ Disk Copy program SCR#2                                          :
:  : !$ELEM ASCII , WORD HERE C@ 1+ ALLOT ;                          :
:                                                                    :
:  : <ER1> ." Index too large" ;                                     :
:                                                                    :
:  : $ARRAY ( n -- )                                                 :
:    <BUILDS DUP C, 0 DO !$ELEM LOOP DOES>                           :
:    DUP C@ ROT DUP >R >                                             :
:    IF   1+ R> -DUP  IF 0 DO COUNT + LOOP THEN                      :
:    ELSE R> DROP DROP CR [ ' <ER1> 2+ ] LITERAL THEN ;              :
:                                                                    :
:  7 $ARRAY $msg Reading Block: ,Writing Block: ,Please insert SO:
:URCE disk and,Please insert TARGET disk and,hit any key when rea:
:dy,Disk Copy Complete!,PLEASE ENTER # SECTORS ON DISK: ,            :
:   -->                                                              :
+--------------------------------------------------------------------+
```

MARCH,1987

WHEREFORTHS OF FORTH CONTINUES...

```
\ Disk Copy program SCR#3
 : .RB    0 $msg 2 10 DISPLAY_AT ; : .Rblk 17 10 GOTOXY . ;
 : .WB    1 $msg 2 12 DISPLAY_AT ; : .Wblk 17 12 GOTOXY . ;
 : ANYKEY 4 $msg 2  8 DISPLAY_AT KEY DROP ;
 : .SD    2 $msg 2  7 DISPLAY_AT ANYKEY ;
 : .TD    3 $msg 2  7 DISPLAY_AT ANYKEY ;
 : .COMPL 5 $msg 2 20 DISPLAY_AT ;
 : .ASK   6 $msg 2 23 DISPLAY_AT ;

  -->
```

```
\ Disk Copy Program SCR#4
 BASE->R DECIMAL
 0 VARIABLE #B  \ Number of Blocks
 0 VARIABLE BP  \ Block Pointer
 S0 @ PAD 40 + -   B/BUF /  CONSTANT #HB \ # Hi  Bufs
 LIMIT FIRST -     B/BUF /  CONSTANT #LB \ # Low Bufs
 HEX 8370 @ 1400 - B/BUF /  CONSTANT #VB \ # VDP Bufs
 1400 CONSTANT VDPBUF \ Address of first VDP Buffer

 R->BASE
  -->
```

```
\ Disk Copy Program SCR#5
 : stoi ( addr -- n ) 0. ROT DUP >R (NUMBER) R COUNT + = 0=
   IF R COUNT CR TYPE ."  NOT A NUMBER" THEN R> DROP DROP ;
 : GETN     ( -- n ) ACCEPT_AT IOBUF stoi ;
 : !#B      ( #sectors -- ) 4 / #B ! ;
 : #B@      ( -- )  #B @ ;
 : B@       ( -- n ) BP @ ;
 : B!       ( n -- ) BP ! ;
 : B+       ( -- )    1 BP +! ;
 : READIT   ( addr blk# -- ) DUP .Rblk 1 R/W B+ ;
 : WRITEIT  ( addr blk# -- ) DUP .Wblk 0 R/W B+ ;
 : ASK_SIZE .ASK GETN !#B ;
  -->
```

```
\ Disk Copy Program SCR#6
 : GET_#B 0 BLOCK 10 + @ DUP
   CASE  360 OF !#B ENDOF
         720 OF !#B ENDOF
        1280 OF !#B ENDOF
        1440 OF !#B ENDOF
        ASK_SIZE ( NON-STANDARD DISK HEADER )
   ENDCASE 2 5 GOTOXY ." Your disk has " #B@ . ." blocks" ;

 : LO(I)  ( I -- addr )  B/BUF * FIRST  + ;
 : HI(I)  ( I -- addr )  B/BUF * PAD 40 + + ;
 : VB(I)  ( I -- vaddr ) B/BUF * VDPBUF + ;
 : 3DROP DROP DROP DROP ;

 -->
```

CONTINUED ON PAGE 11 ==>

-----------------------------MARCH,1987-----------------------------

Computer Tutor

by Tom Moran

SORTING OUT THE SORTS

One of the most useful applications of your computer is using it's processing power to sort and alphabetize lists. In this month's edition of COMPUTER TUTOR we are going to examine a couple of basic programming sort routines that will help you keep those lists in order.

Volumes have been written on various sorting techniques. The theory examined in these books and thesis' is really beyond the scope of this article. Our main purpose is practical application or what is the easiest and fastest way we can alphabetize or sort a list. Before we look at the basic routines, there are memory limitations you should know about.

DISK SORT vs **RAM SORT:** The whole purpose of using a computer to keep track of lists for you is speed and ease. The longer your lists, the more valuable is a computer to sort. However, the longer the list, the more computer memory you will need to accomplish this task.

The fastest method is to sort your lists in RAM. Just how long that list can be is determined by how much "Stack Memory" you have available in your program. Computer memory is divided into two types: **"Program Memory"** and **"Stack Memory"**. Program memory is a count of the bytes taken up by basic instructions. Stack Memory is the area that keeps track of all the variables you include in your basic program and is the area where the sorting takes place. To determine how much program and stack memory you have available, in any given program, simply type the command "SIZE" in Extended Basic.

Let's say you are going to create a basic mailing list program that keeps track of first and last name, address, city, state, zip-code and phone number. With all that information you should be able to sort about 100 to 125 entries in computer RAM before you will run out of stack memory. Well... what if you have 250 or 500 names you want to organize? If this is the case, you will then need to perform a disk sort. A disk sort partitions your list, loading a segment of your list into RAM, sorting it, writing it to a temporary file and then loading another segment until the sort process is completed. Disk sorts are very sloooow. You are probably better off creating separate files that you can sort in RAM than having one monster list that requires a disk sort routine to organize. For example, have one file that keeps track of your Christmas Card list... another that keeps track of business contacts and a third that keeps track of personal records. If you keep all those files on one disk you'll find that it is faster to load those files separately and sort them in RAM.

One last thing before we look at a couple of specific routines. There are a couple of assembly language sort routines floating about in the TI-99 world. These are pretty fast but they don't help much in disk sorts. The speed of your disk drive, a mechanical device, is what slows everything down.

The two sort routines we will examine are the "Selection Sort" and the "Quick Sort". The Selection Sort is the easiest to write and understand but performs the slowest. To make these routines simple, we'll sort lists of numbers rather than strings. To change these routines for string sorts, you'll simply need to add the $ after the variables A and B.

```
100 DIM A(50)
110 N=50
120 CALL CLEAR
130 FOR I=1 TO N
140 RANDOMIZE
150 A(I)=INT(RND*100)+1
160 PRINT A(I);" ";
170 NEXT I
```

MARCH,1987

Computer Tutor Continues

```
180 PRINT
```

The above routine simply creates 50 numbers between 1 and 100 to sort. This is our list.

```
200 FOR I=1 TO N-1
210 FOR J=I+1 TO N
220 IF A(I)<=A(J)THEN 240
230 B=A(I)
240 A(I)=A(J)
250 A(J)=B
260 NEXT J
270 NEXT I
```

This is the sort routine. It processes the list of numbers, putting them into ascending order, smallest to largest.

```
300 FOR I=1 TO N
310 PRINT A(I):" ";
320 NEXT I
330 END
```

This will print out the new list of sorted numbers.

By changing the value of N in line 110 you can change the number of items to be sorted. You must also change the DIM statement in line 100 to reflect the new list size as well.

You will note that as you increase the value of N, you increase the time it takes to sort the list. Sorting 50 numbers takes about 25 seconds. An increase in items does not cause a proportional increase in sort time, however. Change the value of N to 100 and I believe you'll discover that the sort time is about 90 seconds. You doubled the list size but almost quadrupled the time it takes to sort the list.

Never fear! There is a solution. Next month we'll present the "Quick Sort" routine. This little gem not only works faster but will actually sort 100 items faster than the selection sort was able to sort 50 items.

You may also be wondering how to keep track of all the other variables in your list. For instance, if you are sorting a mailing list by last name, how do you make sure that the first name, address and all the other variables are kept in te proper order. This is accomplished through the use of "pointers". We'll also show you how to use pointers in sort routines.

TOM MORAN

HINTS AND TIPS

TMS 9900 ASSEMBLY LANGUAGE TUTORIAL
Part 2
(FASTER_THAN_THE_HUMAN_EYE)

by Steve Royce - WNY 99'ERS

The most amazing aspect of Assembly Language is its unbelievable speed. Since the Object Code which is generated by the Assembler needs no translation as it is being run, the response to any instruction seems instantaneous. In the example we're going to use to demonstrate the speed, we are going to have to build in a time-delay loop just to slow the program down enough so you can see what it is doing. Remember, your video display can only be changed sixty times per second, but a good Assembly Language routine can attempt to change it thousands of times per second.

The speed of Assembly Language is beneficial in game programs (sprite coincidences are never missed), utility routines (a 200 element alphabetical sort executes in six seconds), and in mathematical processing of large amounts of data. In fact, think of any Basic or Ex-Basic routine that you think is a bit slow. You can speed it up hundreds of times by using Assembly Language.

Last month, I gave a short Ex-Basic program to move the letter 'A' across the screen one space at a time. I hope you have all tried it to see what we are trying to duplicate in Assembly Language.

The following is my Assembly Language program, written for the Editor/Assembler, to do the same thing:

```
0001          DEF A
0002          REF VSBW
0003 A        LI R0,0
0004 A1       LI R1,>4100
0005          BLWP @VSBW
0006          LI R1,1000
0007          DEC R1
0008          JNE $-2
0009          LI R1,>2000
```

HINTS AND TIPS CONTINUES...

```
0010        BLWP @VSBW
0011        LIMI 2
0012        LIMI 0
0013        CI R0,767
0014        JEQ A
0015        INC R0
0016        JMP A1
0017        END A
```

Lines 0001 to 0005 should be familiar from last month's article. We are DEFining the program name, REFerencing an external label, and using the VSBW routine to place the letter 'A' at the upper left corner of the screen.

0006 LI R1,1000 This is the start of our time delay loop. We are loading R1 with the fixed number 1000.

0007 DEC R1 DECrease the contents of R1 by a value of one. It's significant to remember that this instruction takes two bytes, as line 0008 will show.

0008 JNE $-2 JNE means Jump if Not Equal. The system will look at the results of the last instruction (line 0007) and see if the Equal Bit is set in the Status Register. In our example, the Equal bit will be set if R1 contains the value zero. If it doesn't, the program will go back two bytes ($-2) to line 0007 and DECrease R1 again, until it finally reaches zero, at which time the program will continue to the next instruction. This ends our delay loop.

0009 LI R1,>2000 In this step, we are preparing for another VSBW routine. We are loading R1 with the character which we wish to write to the VDP address given in R0. Since we haven't altered R0 in any step since line 0001, we are over-writing the current location of the letter A. The character which we are going to place is the BLANK character, Hex >20 or ASCII 32.

0010 BLWP @VSBW Writes the blank to the screen.

0011 LIMI 2 Allow interrupt
0012 LIMI 0 Disable interrupt

Line 0011 allows you to use the quit key, then line 0012 disables interrupts so we can again access the VSBW routine.

0013 CI R0,767 Compare Immediate (CI) the contents of R0 to the fixed value 767. This line checks to see if we have written to the last screen location, the lower right hand corner, VDP Address 767. Depending on the results of this comparison, appropriate action is taken in lines 0014 to 0016.

0014 JEQ A If the comparison proves equal, that is if we have just written to the last screen location, jump to the instruction labled A, which will load R0 with the value of zero as the next VDP Address to be written to. This starts the program over at the upper left hand corner. JEQ means Jump if EQual.

0015 INC R0 If the comparison is not equal, the program advances to this line. Here we INCrease the value in R0, which loads R0 with the next screen location to be written to.

0016 JMP A1 Jump to the instruction labled A1, which loads the letter A into R1 for the VSBW routine.

0017 END A The END directive followed by the program name.

Following the instructions given last month, save the source code, assemble it and load the object code. Impressed? If not, try reducing the delay loop in line 0006. The blur gets incredible as the delay loop approaches zero.

How much faster is it than the Ex-Basic version? The Ex-basic version takes thirty-two seconds to move the A from the upper-left corner to the lower-right corner on my computer, with no delay loop. Once I add a zero to 1000 delay loop to the Ex-basic version to make the programs identical, it takes 2520 seconds. The Assembly version takes 9.5 seconds with its one thousand delay loop. So, the Assembly version is about 270 times faster. Putting it another way, a program that would take over four minutes to execute in Ex-basic would execute in under one second in Assembly. That ain't bad.

Next month, we'll start to simplify our Assembly Language programming by writing subroutines to imitate the TI BASIC and Extended BASIC subroutines, such as 'CALL CLEAR' or 'CALL SPRITE'. Once these are available to us, repititive programming of them becomes unnecessary, and our work becomes much easier. Sections 7.20.1 to 7.20.3 of the E/A Manual provide some background into the procedures we will use to create subroutines. Particular attention should be directed to Section 7.20.1 and Page 135 which deal with the 'COMMON WORKSPACE' method of subroutine use, as this is the method which I will employ in my examples next month.

-----------------------------MARCH,1987-----------------------------

WHEREFORTHS OF FORTH CONTINUES...

```
+------------------------------------------------------------------+
: \ Disk Copy Program SCR#7                                        :
:  : GET-LO-BUFS ( -- ) #LB 1 ( Skip FIRST )                       :
:    DO I LO(I) B@ #B@ OVER = IF 3DROP LEAVE THEN READIT LOOP ;    :
:                                                                  :
:  : GET-HI-BUFS ( -- ) #HB 0                                      :
:    DO I HI(I) B@ #B@ OVER = IF 3DROP LEAVE THEN READIT LOOP ;    :
:                                                                  :
:  : GET-VDP-BUFS ( -- ) #VB 0                                     :
:    DO B@ #B@ = IF LEAVE THEN FIRST B@ READIT                     :
:       FIRST I VB(I) B/BUF VMBW                                   :
:    LOOP ;                                                        :
:                                                                  :
:  -->                                                             :
+------------------------------------------------------------------+
: \ Disk Copy Program SCR#8                                        :
: : PUT-LO-BUFS ( -- ) #LB 1 ( Skip FIRST )                        :
:   DO I LO(I) B@ #B@ OVER = IF 3DROP LEAVE THEN WRITEIT LOOP ;    :
:                                                                  :
: : PUT-HI-BUFS ( -- ) #HB 0                                       :
:   DO I HI(I) B@ #B@ OVER = IF 3DROP LEAVE THEN WRITEIT LOOP ;    :
:                                                                  :
: : PUT-VDP-BUFS ( -- ) #VB 0                                      :
:   DO   B@ #B@ = IF LEAVE THEN I VB(I) FIRST B/BUF VMBR           :
:        FIRST B@ WRITEIT                                          :
:   LOOP ;                                                         :
:                                                                  :
:  -->                                                             :
+------------------------------------------------------------------+
: \ Disk Copy Program SCR#9                                        :
:  : COPY-DISK TEXT DR0 .SD GET_#B 0 B!                            :
:    BEGIN B@ #B@ <                                                :
:    WHILE CR .RB B@ >R                                            :
:          GET-LO-BUFS GET-HI-BUFS GET-VDP-BUFS .TD                :
:          CR .WB R>,B!                                            :
:          PUT-LO-BUFS PUT-HI-BUFS PUT-VDP-BUFS .SD                :
:    REPEAT                                                        :
:    CR .COMPL ;                                                   :
:                                                                  :
:  S0 @ PAD 40 + - B/BUF / ' #HB ! \  Set #HB to final value      :
+------------------------------------------------------------------+
```

Rene' LeBlanc
<<<<<<<>>>>>>>

QUIK TIP....

Since we are doing a column on Assembly Language, I thought a little "One Liner" on converting decimal numbers to HEX numbers might be in order. As you may know, hexidecimal is a numbering system based on 16 instead of 10 (normal numbers you are used to). There are various ways you can do the conversion from decimal (Base 10) to HEX. You could do it with a special calculator or, if you are really good, figure it out in your head, or just use this little program for your computer. The program is courtesy of the TI-Omaha Users' Group.

```
1 A$="0123456789ABCDEF" :: INPUT X :: Y=INT(X/16) :: Z=X-Y*16 ::
PRINT SEG$(A$,Y+1,1)&SEG$(A$,Z+1,1)
```

VALLEY OF THE SUN TI USER GROUP
1425 E. DEL RIO DR.
TEMPE, AZ. 85282

PHOENIX AZ
MAR 25
PM
1987
85026

EDMONTON 99ERS COMPUTER SOC.
PO BOX 11983
EDMONTON, ALBERTA
CANADA T5J 3L1