Ø94 8704
VAST 99

50¢

VALLEY OF THE SUN TI 99 USERS GROUP newsletter

VOL. 3 APRIL 11, 1987 NO. 4

****

CONTENTS

VAST 99 BBS 437-4335

------------------------------APRIL,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
DS/SD Disk $4.00

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. A complete list of what is in the library is available on 2 disks free of charge if you supply the disks or for $1.00 per disk if the User Group supplies the disks.

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

APRIL, 1987

From the Editors Desk

MINUTES MARCH, 1987

The March meeting of VAST 99 TI USER GROUP was held on Saturday March 14th at the Los Olivos Hotel in Phoenix, AZ. The meeting was called to order by President Gerry Kennedy at 10AM.

We opened with the usual question "When will the new Myarc computer be shipped?" The answer was "maybe this month!" Then again maybe not.

The only serious business conducted at this meeting was the nomination of a slate of candidates for the board of directors of this group. The following are the nominees that were willing to serve:

PRESIDENT--Mike Marfisi
VICE PRES--Stu Olson
SECRETARY--Bob Nixon
TREASURER--Ike Van Kampen

No one else was willing to accept a nomination for any office so a motion was made, seconded and approved to accept this slate as nominated. There will be no election at the April meeting and these nominees will assume office as of May 1.

Mike Marfisi
Secretary

COMINGS AND GOINGS

May we all extend a warm welcome to our new members:

Steve Peddy-Coart
Don Shores

We say goodbye to:

~~Claire Rottenberg~~
Eloise Peterson

Next month I will have a complete update of the membership roster for everyone at the meeting. Remember they are issued once every 6 months. If you want one between times, a SASE will get you the latest listing. Of course Bob Nixon who will be taking over this post soon may have other ideas so check with him.

NEWSLETTER EXCHANGE

We are now trading newsletters with over 60 other TI user groups around the US and Canada. We are getting a collection of really fine publications and building up quite a library. Check them out at the next meeting. (Please see addendum to this paragraph elsewhere in this issue. *Editor*)

That's it for now.

Mike

(—) (—) (—) (—) (—)

IN THIS ISSUE

If you remember, (how could you forget?) last month there was an article in this Newsletter called "TINYGRAMS." This month we have a great example of how a TINYGRAM is arrived at. It's on page 5. Page 6 has Rene' LeBlanc's Forth Column. This month we go back a couple of issues for a correction to a previous WHEREFORTHS on putting a frame around text and "stuff" on the screen. A few bugs got into it. Next month he will begin his promised description of the DISK UTILITY started last month (HONEST). COMPUTER TUTOR is on Page 8 and continues last month's subject of sort routines. Our ASSEMBLY LANGUAGE TUTORIAL continues on Page 9. This month we show how to emulate some taken-for-granted subroutines of BASIC (CLEAR and HCHAR). One of the last "TIPS FROM THE TIGERCUB" had an interesting article on using those "un-usable" character sets (#15 and #16) that were taken away in X-BASIC. By merging this program into a BASIC ONLY program, you can run your BASIC program in X-BASIC! That is on page 11 and rounds out this month's issue.

OTHER STUFF...

I guess it is true that about 2000 of the MYARC 9640 computers have been shipped! Anybody around here have

Editors Desk Continues

one? A number of magazines (MICRO-PENDIUM, COMPUTER SHOPPER) have received one to try out and preliminary results are quite good. Sure hope we can see one soon... Incidently, it seems that those GATE ARRAYS that MYARC waited sooo long for and finally got and tested good, wellll turned out they were bad again and that is the hold-up once more.

The BBS has passed another milestone. We have turned over 4000 callers. There has been some excitement on the board this month in the form of a new user and maybe if we look real close, we will see Brenda Wall at the meeting becoming a New Member of our Users Group.

The group is planning to have another of its famous SWAP MEETS at the next meeting. Announcements will be put in the newspaper to try and attract more new users. At our last SWAP, we signed a number of new people. If you have been holding on to it and don't use it anymore and want to get rid of it, bring it to next month's SWAP meet!

That's about it. Now here'sssss MIKEY with an important announcement:

Jim (THE SYSOP, EDITOR) Ely

NEWSLETTER EXCHANGE

For the past several months we have been mailing our newsletters to about 60 user groups around the US and Canada. These groups have in turn sent us copies of their newsletters. This exchange program allows us to see what is happening in the TI community and allows us to share information among dozens of other groups. It also gives us a direct tap into the minds of hundreds of very smart programmers and experts in every aspect of the TI world.

These newsletters are collected into monthly binders and are available for your use. At this point I am bringing them to each meeting where you can look through them and take a volume home if you like. We ask that you take only one volume at a time, let me know you have it, and then return it the next month.

These newsletters are so full of valuable information that I feel we have to find a way to know exactly what they contain. Therefore, I have started a major project to catalog each edition so that we can find articles that may be of special interest to us.

I have started this project but, as always, my desires outlast my available time. Sooo.....I am asking for your help. I would like anyone who can spare some time to take a volume home and key in the info for the general index.

I have chosen to use Asgard's TOTAL FILER program for this task. It is a free form data base that is ideal for this type of project. I will supply the disks and program, along with the format we are using so we can get total consistency in the data base.

If you are interested in helping with this project, please let me know. A complete list of the clubs with whom we currently exchange newsletters is available separately (See me or Jim Ely for a copy).

Mike Marfisi

CLASSIFIED ADVERTISING

Introducing Roll Credits
Create your own titles and credits for your video productions using the TI-99/4A. For more information send SASE to:

Newcastle Enterprises
P.O. Box 31881-V04
Phoenix, AZ 85046

------------------------------APRIL,1987------------------------------

ODDS AND ENDS

Last month I included an article on an idea called TINYGRAMS. In taking that idea a little bit farther, this month I would like to show you what is meant by "logical thinking" in programming. In the March issue of **The Greater Omaha TI Users' Group Newsletter**, John Witte writes...

What we have this month is a three liner (w h a t ?). I was looking through another club's newsletter (West Penn. - Jan.), and they had a program that put waves on the screen. I thought it would be fun to see if it could be a "one liner". The original program follows with REM statements deleted:

```
100 CALL CLEAR :: CALL SCREE
N(1):: S=1
110 A$="000000000000FFFF" ::
 B$="00000000000FFFF" :: C$=
"00000000FFFF" :: D$="000000
FFFF" :: E$="0000FFFF" :: F$
="00FFFF" :: G$="FFFF"
120 H$="30303EFF7F3E1E04" ::
 CALL CHAR(103,H$):: CALL SP
RITE(#5,103,2,180,180,-30,0)
:: CALL SPRITE(#6,103,2,80,1
00,-30,0):: CALL MAGNIFY(2)
130 CALL CHAR(96,A$):: CALL
CHAR(97,B$):: CALL CHAR(98,C
$):: CALL CHAR(99,D$):: CALL
 CHAR(100,E$):: CALL CHAR(10
1,F$):: CALL CHAR(102,G$)
140 AA$,MM$=CHR$(96)&CHR$(97
)&CHR$(98)&CHR$(99)&CHR$(100
)&CHR$(101)&CHR$(102)&CHR$(1
01)&CHR$(100)&CHR$(99)&CHR$(
98)&CHR$(97)&CHR$(96)
150 BB$,LL$=CHR$(97)&CHR$(98
)&CHR$(99)&CHR$(100)&CHR$(10
1)&CHR$(102)&CHR$(101)&CHR$(
100)&CHR$(99)&CHR$(98)&CHR$(
97)&CHR$(96)&CHR$(97)
160 CC$,KK$=CHR$(98)&CHR$(99
)&CHR$(100)&CHR$(101)&CHR$(1
02)&CHR$(101)&CHR$(100)&CHR$
(99)&CHR$(98)&CHR$(97)&CHR$(
96)&CHR$(97)&CHR$(98)
170 DD$,JJ$=CHR$(99)&CHR$(10
0)&CHR$(101)&CHR$(102)&CHR$(
101)&CHR$(100)&CHR$(99)&CHR$
(98)&CHR$(97)&CHR$(96)&CHR$(
97)&CHR$(98)&CHR$(99)
180 EE$,II$=CHR$(100)&CHR$(1
01)&CHR$(102)&CHR$(101)&CHR$
(100)&CHR$(99)&CHR$(98)&CHR$
(97)&CHR$(96)&CHR$(97)&CHR$(
98)&CHR$(99)&CHR$(100)
190 FF$,HH$=CHR$(101)&CHR$(1
02)&CHR$(101)&CHR$(100)&CHR$
(99)&CHR$(98)&CHR$(97)&CHR$(
96)&CHR$(97)&CHR$(98)&CHR$(9
9)&CHR$(100)&CHR$(101)
200 GG$=CHR$(102)&CHR$(101)&
CHR$(100)&CHR$(99)&CHR$(98)&
CHR$(97)&CHR$(96)&CHR$(97)&C
HR$(98)&CHR$(99)&CHR$(100)&C
HR$(101)&CHR$(102)
210 FOR X=1 TO 1000 :: CALL
SCREEN(S)
220 PRINT AA$&BB$ :: PRINT B
B$&CC$ :: PRINT CC$&DD$ :: P
RINT DD$&EE$ :: PRINT EE$&FF
$ :: PRINT FF$&GG$ :: PRINT
GG$&HH$
230 PRINT HH$&II$ :: PRINT I
I$&JJ$ :: PRINT JJ$&KK$ :: P
RINT KK$&LL$ :: PRINT LL$&MM
$
240 IF X>1 THEN S=6
250 NEXT X
```

All of that amounts to 23032 bytes remaining when SIZEd. And now the fun begins. After working for a while, the following program results, representing 24136 bytes remaining:

```
1 CALL CLEAR :: A$(1)="ABCD
EFGFEDCBA" :: FOR I=1 TO 7
:: CALL CHAR(72-I,RPT$("0",
2*I-2)&"FFFF",47,"30303EFF7
F3E1E04"):: A$(I+1)=SEG$(A$
(I),2,12)&SEG$(A$(I),2,1) :
: NEXT I
2 CALL SPRITE(#5,47,2,180,1
80,-23,0,#6,47,2,80,100,-23
,0):: CALL MAGNIFY(2):: CAL
L SCREEN(6)
3 FOR I=1 TO 12 :: PRINT A$
(I+(I>7)*2*(I-7))&A$(1+I+(I
>6)*2*(I-6)):: NEXT I :: GO
TO 3
```

A couple of notes...

A$(1) is manipulated in the FOR - NEXT loop of line 1 to offset the sequence of letters. In both lines 1 and 2, multiple characters or sprites are CALLed by one statement. In line 3, the A$ strings are recombined in a manner that produces a sawtooth wave. The (I>7) or (I>6) clauses are equal to 0 until true, at which time they equal -1 and begin to reduce the value of the array parameter.

But still no oneliner.

(*Editor's comments.* This shows what a little time and thinking about a problem can yeild. With the TI's limited memory, this is the kind of efficient programming that is needed. Keep your eyes out for other possibilities and thanks, John, for the great example!)

APRIL,1987

WHEREFORTHS OF FORTH

DRAWGRID REVISITED

In WHEREFORTHS #14, I presented a complete Forth program to perform sector-by-sector single drive disk copy, and promised to begin explaining it in this issue. However, before I get into that, I want to commend our VAST member TOM SHARP for living up to his name and discovering that your WHEREFORTHS author gave you a buggy program in WHEREFORTHS #12. Tom noticed that if he didn't locate the upper left corner of a grid at "0,0" the program would take off and draw a confused bunch of lines all over the screen that didn't begin to resemble a grid!

As it turns out, I confused the vertical tab and horizontal tab definitions on the 3rd screen f o r t h e words "verts" and "horizs" and then compensated for it in the way I provided the example on the test screen. Just by improbable chance, the test cases happened to work ok!

As I analyzed the problem, I noticed another oversight in my original design. If one wanted a simple box with no interior grid lines, I neglected to provide the case where the number of vertical and hori-zontal tabs is zero (I must have been rushing to get that issue in on time).

Tom Sharp made another observation that the vertical and horizontal tabs were specified relative to the screen, rather than relative to each individual grid structure. It would be much better to specify a grid structure as a self-contained "object" so you could relocate it anywhere on the screen by specifying its upper left-hand corner location, and in moving it around not need to respecify the values of the vertical and horizontal tabs.

Tom also noticed that if he tried to compile the drawing of a grid into another program instead of only interactively entering the DRAWGRID command, it didn't work. I have changed the DRAWGRID command so that it now expects the address of the specific grid to be drawn on the stack. In this form, it can now be executed from within another Forth program. For example, you could compile the following:

```
: TEST TEXT G4 DRAWGRID ;
```

TEXT ensures the screen is in text mode and clears the screen. G4 must be previously defined using the MKGRID defining word. It returns its parameter field address to the stack. Then DRAWGRID will draw the G4 grid. This will all execute when executing the word TEST.

I provided an interactive word "DG" which can be used like the old DRAWGRID word. Interactively (from your keyboard or from load screen) you can enter:

```
DG G4
```

and it will draw the G4 grid on the screen (be sure it is in TEXT mode first). I think this version of the DRAWGRID screens corrects all the bugs and adds the suggested enhancements to make it a much more useable function.

I want to extend my thanks to Tom Sharp for his participation and useful comments. Next month, I really will begin to analyze the COPY-DISK program.

```
+------------------------------------------------------------------------+
: \ GRID - define graphic elements                                        :
:  BASE->R HEX                                                            :
:  90 CONSTANT ul 91 CONSTANT ur 92 CONSTANT ll 93 CONSTANT lr            :
:  94 CONSTANT le 95 CONSTANT re 96 CONSTANT tp 97 CONSTANT bt            :
:  98 CONSTANT cr 99 CONSTANT vl 9A CONSTANT hl                           :
:                                                                         :
: : SETCHAR ( -- )                                                        :
:   0000 003C 3C30 3030 ul CHAR    0000 00F0 F030 3030 ur CHAR            :
:   3030 303C 3C00 0000 ll CHAR    3030 30F0 F000 0000 lr CHAR            :
:   3030 3030 3030 3030 vl CHAR    0000 00FC FC00 0000 hl CHAR            :
:   3030 303C 3C30 3030 le CHAR    3030 30F0 F030 3030 re CHAR            :
:   0000 00FF FF30 3030 tp CHAR    3030 30FF FF00 0000 bt CHAR            :
:   3030 30FF FF30 3030 cr CHAR ;                                         :
:  R->BASE -->                                                            :
:                                                                         :
+------------------------------------------------------------------------+
```

APRIL,1987

WHEREFORTHS OF FORTH CONTINUES...

```
+--------------------------------------------------------------------+
\ GRID
 : MKGRID ( vt1 .. vtn vtc ht1 .. htn htc ulc ulr wdth hgth ) .
   <BUILDS C, C, C, C,
   2 0 DO DUP 1+ 0 DO C, LOOP LOOP HERE 2 MOD ALLOT DOES> ;

 \ ( adr -- u adr ) for following words:
 : hgth DUP     C@ SWAP ;       : wdth DUP 1+  C@ SWAP ;
 : ulr  DUP 2+  C@ SWAP ;       : ulc  DUP 3 + C@ SWAP ;
 : &htc DUP 4 + ;               : htc  &htc C@ SWAP ;
 : &vtc htc SWAP OVER 5 + + ; : vtc  &vtc C@ SWAP ;
 : ht(I) ( adr I -- ht adr ) \ ht relative to ulc
   SWAP &htc ROT + C@ SWAP ulc >R + R> ;
 : vt(I) ( adr I -- vt adr ) \ vt relative to ulr
   SWAP &vtc ROT + C@ SWAP ulr >R + R> ;
  -->
+--------------------------------------------------------------------+
\ GRID - verts horizs
 : verts ( adr -- adr ) htc SWAP -DUP IF 1+ 1
   DO I ht(I) ulr hgth >R      vl VCHAR R>
      I ht(I) ulr      >R    1 tp VCHAR R>
      I ht(I) ulr hgth >R + 1 bt VCHAR R> LOOP THEN ;

 : horizs ( adr -- adr ) vtc SWAP -DUP IF 1+ 1
   DO ulc SWAP OVER          I vt(I) wdth DROP   hl HCHAR
      ulc SWAP OVER          I vt(I)      DROP 1 le HCHAR
      DUP ulc wdth >R + R> I vt(I)        DROP 1 re HCHAR
      htc SWAP 1+ 1
      DO  DUP I ht(I) J vt(I) DROP 1 cr HCHAR LOOP
   LOOP THEN ; -->
+--------------------------------------------------------------------+
\ GRID - drawgrid
 : DRAWGRID ( addr -- ) SETCHAR
   ulc ulr wdth >R hl HCHAR R>
   ulc ulr hgth >R + R> wdth >R hl HCHAR R>
   ulc ulr hgth >R vl VCHAR R>
   ulc wdth >R + R> ulr hgth >R vl VCHAR R>
   ulc ulr >R 1 ul HCHAR R>
   ulc wdth >R + R> ulr >R 1 ur HCHAR R>
   ulc ulr hgth >R + 1 ll HCHAR R>
   ulc wdth >R + R> ulr hgth >R + 1 lr HCHAR R>
   verts horizs  DROP ;

 : DG ( interactive usage: DG <grid name> )
   [COMPILE] ' DRAWGRID ;
   -->
+--------------------------------------------------------------------+
\ GRID - test
 BASE->R DECIMAL

         5 15 2              10 20 2   0 0 30 20 MKGRID G1

 3 6 9 12 15 5  5 10 15 20 25 30 6   0 0 39 23 MKGRID G2

                                       0 0 10 5 10 15 MKGRID G3

                            8 1   5 1 10 5 10 15 MKGRID G4

                  3 6 9 3   4 8 12 3  10 5 16 15  MKGRID G5
 R->BASE
+--------------------------------------------------------------------+
```

by Rene' LeBlanc

APRIL,1987

SORTING OUT THE SORTS

Part II

Last month we briefl covered some theory on basic language sort routines and showed how you a very simple selection sort routine. This month we will include a more complex sort routine called the Quick Sort.

Selection Sort vs Quick Sort:

The selection sort is a simple algorithm that consists of a pair of nested FOR-NEXT loops. It always goes through the complete number of passes set in those loops regardless of whether the list is in order or not. This wastes alot of time, especially if you have a bunch of data. The quick sort works by dividing your data into two parts... a top list and a bottom list. It first chooses an item on the bottom of the list and places it in its proper place relative to the items in the list. Then all the items of lesser value go to the bottom and the items of a greater value go to the top. The two lists are repeatedly divided with items being exchanged until the entire array is sorted.

Interestingly enough, the quick sort is a much longer routine but sorts a list of 100 items in half the time a selection sort routine can sort 50 items. Here's the routine:

```
110 DIM A(100)
120 N=100
130 CALL CLEAR
140 FOR I=1 TO N
150 RANDOMIZE
160 A(I)=INT(RND*100)+1
170 PRINT A(I);
180 NEXT I
190 PRINT
200 P=1
210 L(P)=1
220 R(P)=N
230 IF P<=0 THEN 610
240 LB=L(P)
250 RB=R(P)
260 P=P-1
270 IF RB<=LB THEN 230
280 I=LB
290 J=RB
300 T=A(I)
310 IF J<1 THEN 350
320 IF T>=A(J)THEN 350
330 J=J-1
340 GOTO 310
350 IF J>I THEN 380
360 A(I)=T
370 GOTO 500
380 A(I)=A(J)
390 I=I+1
400 IF I>N THEN 440
410 IF A(I)>=T THEN 440
420 I=I+1
430 GOTO 400
440 IF J<=I THEN 480
450 A(J)=A(I)
460 J=J-1
470 GOTO 320
480 A(I)=T
490 I=J
500 P=P+1
510 IF I-LB>=RB-I THEN 560
520 L(P)=I+1
530 R(P)=RB
540 RB=I-1
550 GOTO 270
560 L(P)=LB
570 R(P)=I-1
580 LB=I+1
590 GOTO 270
600 REM PRINT NEW SORTED LIS
T
610 FOR I=1 TO N
620 PRINT A(I);
630 NEXT I
640 END
```

That's a lot of program lines! However, you'll notice there are NO For-Next loops anywhere in the sort routine (lines 200-590). As soon as the list is in order, the program exits the sort routine and doesn't spend useless time checking every item in your data list.

APRIL, 1987

Computer Tutor Continues

Also, this sort routine is written in console basic to make it easier for you to follow what is happening. If you desire, you can combine several of the statements on the same line, thus shortening the length of the routine and also actually speeding up the sorting process a little as well.

In comparing the sort time of 100 items I clocked the selection sort at just over 3 minutes while the quick sort got everything in order in about 20 seconds. And the difference is even greater with the more items you need to sort.

Next month we'll conclude our tutoring session on sorting by looking at pointers. Say you have a list of names, address's and phone numbers and you want to sort that list by name. How do you drag along the right address's and phone numbers during that sort? Through the use of pointers. See ya next month!

TOM

HINTS AND TIPS

TMS9900 ASSEMBLY LANGUAGE TUTORIAL
PART 3
THE_BEAUTY_OF_BASIC

by STEVE ROYCE - WNY 99'ERS

One of the very nice things about TI BASIC or Extended BASIC is the great number of built-in subroutines which do a lot of dirty work for you. Routines like CALL CLEAR or CALL SPRITE are so much easier to use than having to program the instructions each time you want to use them.

These routines don't exist in Assembly Language, but you can create them, and, if Assembly Language is your game, you should have them at your disposal. I got deeply involved and confused writing an Assembly Language game before it finally dawned on me that these subroutines are BASIC's most beautiful feature and that I needed them in my Assembly Language programs. I have since written a number of subroutines which imitate the BASIC and Extended BASIC routines. My versions are stored as a source file called 'SUBS' which I ~~copy into my program as it is being~~ assembled.

Using a subroutine in Assembly requires that we Branch to a label (the name of the subroutine) and provide a Link so that we may return to the calling program when the subroutine ends. The BL instruction (Branch and Link) provides these means-BL @CLEAR will Branch and Link to a routine called 'CLEAR'. The BL routine will place the address of the next word following the BL @CLEAR instruction into R11 so that the address is saved for us to return when we are done. The word at the next address following the BL instruction may be an executible instruction, or it may be data which we are going to pass to the subroutine. To pass data, we use the MOV *11+, (destination) instruction to "MOVe the word at the address stored in R11 to the destination, then increase the address stored in R11 by 2 bytes". Once all our data is passed, the address contained in R11 is the next executible instruction of the calling program. The last instruction in the subroutine is then B *11, or Branch to the address stored in R11.

We could alternately use the BLWP instruction, or Branch and Link with

APRIL,1987

[illegible]

Workspace Pointer. This instruction will use its own set of workspace registers and avoid one drawback with the BL method of subroutines: the temporary loss of up to five of your available 16 registers when data must be used in the subroutine. I have found that with good planning, however, you can live with the temporary loss of these registers, and so the BL route is the one that I have taken and will present here.

The following two examples will demonstrate two subroutines which I have used in my programs. The first is an imitation of 'CALL CLEAR' and requires no data to be passed, the second is 'CALL HCHAR' and demonstrates data passing.

```
        DEF TEST1
        REF VSBW
 TEST1  BL @CLEAR
        LIMI 2
        JMP $
*
* CLEAR SUBROUTINE
* USES R0, R1, R11
* NO DATA PASSED
*
 CLEAR  LI R0,>02FF
        LI R1,>2000
        BLWP @VSBW
        DEC R0
        JGT $-6 BACK TO BLWP
        B *11
        END TEST1


        DEF TEST2
        REF VSBW
 TEST2  BL @HCHAR
        DATA 65,20,10
        LIMI 2
        JMP $
*
* HCHAR SUBROUTINE
* DATA CHAR, START LOC, #REPETITIONS
* USES R0, R1, R2, R11
*
 HCHAR  MOV *11+,R1 (move 65 to R1)
        SWPB R1
        MOV *11+,R0 (move 20 to R0)
        MOV *11+,R2 (move 10 to R2)
        BLWP @VSBW
        DEC R2 (decrease counter)
        JEQ $+6 IF ZERO, GOTO B*11
        INC R0
        JMP $-10 BACK TO BLWP
        B *11
        END TEST2
```

In our first example, we are simply loading R0 and R1 to place the blank character at the lower right corner of the screen, then decreasing the screen location until all locations are filled with the blank. Once done, we return to the calling program with the B *11 instruction.

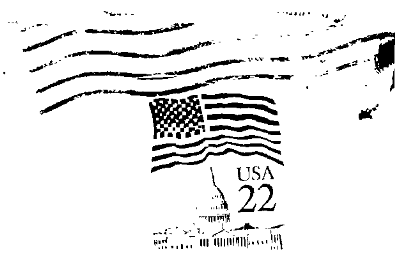
The second example moves three data values into the subroutine. These are the character to write, the screen location to start writing to and the number of repetitions, just like the BASIC 'CALL HCHAR'. Note however, that the number of repetitions must be included in the data, even if we are only placing one character, since the subroutine always expects three data values to be passed to it. Once all the data is moved, R11 contains the address of the next instruction in the calling program.

A word of caution about these subroutines, and any subroutines which I will use in future articles. I have not designed any of them to check the accuracy of data input to them or to guard against the incorrect number of data being passed to them. You can incorporate these features if you like, but remember, they will slow your execution time down somewhat.

Next time, we will begin to look at sprites in Assembly Language, and will develop subroutines to do the same type of work as the Extended BASIC 'CALL SPRITE', 'CALL MOTION', 'CALL POSITION' and so on that make sprites so easy to use on the 99/4A. Read Section 21 in your Editor/Assembler manual to acquaint yourself with the addresses necessary to generate sprites. Then we can develop the necessary subroutines to make their use in Assembly Language as easy as it is in Extended BASIC.

NEWSLETTER
[illegible]
[illegible]
[illegible]

APR 27 PM 1987



FIRST CLASS MAIL

TO:

[illegible]
PO BOX [illegible]
EDMONTON, ALBERTA
CANADA T5J 3L1