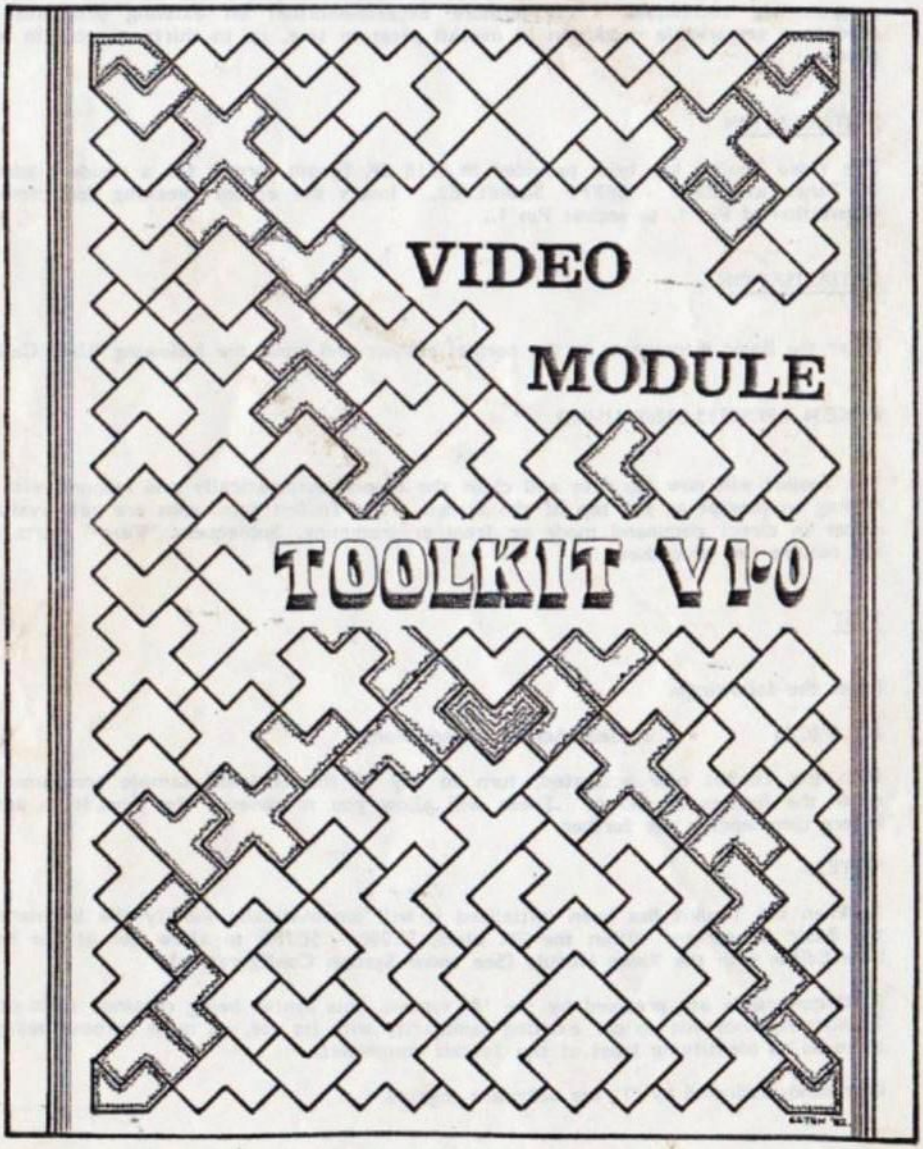
TUG

TANGERINE USERS GROUP



INTRODUCTION

The Video Toolkit has been designed to accompany the Video 80/82 Module as an interface between machine code efficiency and the Basic interpreter language. The facilities offered in this package greatly increase the overall operational speed ratio between the Basic interpreter and the Video Module, subject of course to efficient programming techniques. Furthermore experimentation on existing programs has produce a remarkable reduction in overall program size, up to thirty percent in some cases.

INSTALLATION

The Video Toolkit has been provided in 2716 2K Eprom format for a resident address on Tanex at $E800 - $EFFF Socket E2. Insert the eprom ckecking for correct orientation of Pin 1. to socket Pin 1..

INITIALISATION

Enter the Basic interpreter in the normal manner and input the following 'User Call'.

```
POKE34,0:POKE35,232:X=USR(I)
```

The Toolkit will now activate and clear the screen automatically and respond with its signing on prompt at the top of the screen. The Toolkit commands are now available either in direct command mode or from programming. Subsequent 'Warm' starts will not require this procedure.

TEST

Enter the following:-

```
#CLS (CR)          : Clear Screen Cursor Home
```

With the Toolkit now activated, turn to any of the enclosed sample programs and enter the listings as shown. These will allow you to observe the Toolkit in action before commencing any further.

NOTES:-

1. When the Toolkit has been initialised it will automatically modify the locations in the Basic interpreter within the 2K block $E000 - $E7FF to allow use of the Basic Line Editor with the Video Module (See under System Configuration).

2.All commands are prefixed by the '#' symbol, this syntax being retained from other Toolkits routines due to our existing familiarity with its use, its main purpose being to serve as an identifying label of the Toolkit commands.

Commands followed by 'T' are software toggled.

GRAPHICS CONTROL GROUP

SET x,y

It is assumed for this and other graphic commands that the full VDU 80/82 16K 512x256 display is in use.

The syntax is exactly as shown. Brackets are not required and spaces between the command word and its parameters are 'Don't Care' conditions. Full error checking is carried out by the firmware and the range of x and y is the same for these commands:-

RANGE :- $0 <= x <= 511$, $0 <= y <= 255$

x and y may be any valid Basic numeric variable, real or integer. N.B. Real variables will be rounded.

Error reporting :- ILLEGAL QUANTITY if x or y are outside the specified range.

Function :- Turns on the point at the specified co-ordinates irrespective of existing state and makes x,y the current graphics pen position. Note that the final command string is not sent to the VBUG Silo until it is free of errors therefore preventing hang ups due to incomplete commands.

RESET x,y

All comments for SET command apply here also.

Range :- As in SET command.
Error reporting :- As for SET command.
Function :- The exact reverse of the SET command.

INSET x,y

Prefix IN stands for INVERSE. Comments as for the SET command.

Range :- As for the SET command.
Error reporting :- As for the SET command. Function :- Depends on the exsiting condition of the specified point. If already ON, will perform a RESET function. If already OFF, will perform a SET function.

DRAW x,y

Comments as for the SET command with the addition that programmers should note that no track is kept by the Toolkit as to current position of the graphics pen. This would have consumed too much 0 page usage in memory which is not available with BASIC in residence. However, if the fact that VBUG keeps a track is borne in mind, then this should cause little problem.

Range :- As for the SET command.
Error reporting :- As for the SET command.
Function :- Draws a line from current "pen" position to a point specified as arguement irrespective of current state of potential line.

UNDRAW x,y

Comments as for DRAW command.

Range :- As for DRAW command.
Error reporting :- As for DRAW command.
Function :- Erases a line from current "pen" position to specified position irrespective of current state of the potential line.

INDRAW x,y

Comments the same as for DRAW with the addition that it performs rather like the INSET function does, but on line. Can sometimes be faster than UNDRAW

Range :- As for DRAW command.
Error reporting :- As for DRAW command. Function :- Depends on the existing status of the projected line. If ON performs an UNDRAW function. Then OFF performs a DRAW.

TEST x,y

Comments as for SET command.

Range :- As for SET command. Error reporting :- As for SET command.
Function :- To report back the status of the specified point. Answer is left in Loc.$EE or 238 Dec. If PEEK(238)=1 then point was set, 0 otherwise.

MOVE x,y

Comments as for SET command.

Range :- As for SET command.
Error reporting :- As for SET command.
Function :-To move the graphics "pen" to a new position. N.B. does not SET the new position.

CURSOR CONTROL GROUP.

CLS

No Range or Error reporting to consider! Performs a Clear Screen/Cursor Home function. N.B. Does not clear the Microtan screen.

HOME

Performs a cursor home function (top left corner) on the VDU 80/82 screen ONLY.

UP,DOWN,LEFT,RIGHT.

Performs the obvious ! Note that ";" is not always necessary to maintain PRINT formats on the VDU 80/82 screen.

WINDOW lo,hi

Defines a scrolling window on the VDU 80/82 Screen and if currently outside that window, will home the cursor to the top left corner of it.

Range :- (0<= lo <= 23)<(0<= hi <= 24)
Error reporting :- ILLEGAL QUANTITY if any of the above conditions are not satisfied. N.B. lo can NEVER be greater than hi.
Function :- Sets the scrolling window to between and including lo and hi. Also sets up 2 locations on 0 page as to current window status for use by CURS command. ($22 & $23)

CURS row,col

This command was probably the most complex to get right, as a number of other situations have to be checked before the text cursor can be moved to the new position. Rest assured that it is fully error trapped and, for instance connot be moved to position 60 on a 40 col. screen (WIDTH2) nor can it be moved to row 18 in a scrolling window between 20 & 24.
Range :- Depends on the width of text, but in general 0<=row<=79 , 0<=col<=24
Error reporting :- ILLEGAL QUANTITY if either co-ordinate is out of range for current text conditions.
Function :- To move the TEXT cursor to the specified position for printing inside current window only. On initialisation WINDOW defaults to 0,24 and width to 80 columns. Uses 0 page Locs. $22,$23 (window status lo,hi) and $F7 (current text width) for error trapping. For your convenience a table of text sizes/columns follows :-

WIDTH	COLUMNS
1	0 - 79
2	0 - 40 (!)
3	0 - 27
4	0 - 20 (!)
5	0 - 16

N.B. Widths 2 and 4 are not misprints. Count the line lengths yourself !

General comments on CURS control: Note that Basics PRINT char. counter ($30) is not affected by any of the above commands.

TEXT CONTROL GROUP

WIDTH n

Not much to comment on, except to say that n=0 is valid but the same as n=1. n=6 will generate an error.

Range :- 0<= n<=5
Error reporting :- ILLEGAL QUANTITY if out of range.
Function :- To change to new character size and line lengths. The use of this command
changes WIDTH status byte at $F7.

SPS (Superscript)

Purposely short in its syntax as it will be most used in a text line before a print statement.
Function :- To raise the next letter in the text up half a line. Only valid for one letter.

SBS (Subscript)

Same comments as for SPS

Function :- To lower the next letter in the tex statement. Only valid for one letter.

LU (Line Under)

Syntax is awkward but necessary because of conflicting command names. This command and the next (RVS) work like toggle switches. That is to say that they change the existing status of the relevant function.eg. If underlining was OFF when the command was issued then will turn it ON, and vice versa. Soft toggle for LU is in Loc. $F9.
Function :- To turn undrelining on and off. Changes current state:- when $F9=1 underlining is on. when $F9=0 underlining is off.

RVS (Text reverse video)

Same comments and functions as for LU except soft toggle in $F8.

MISC. GROUP

SYS n

In this implementation and because 0 page storage is at a premium, when the VDU Toolkit is initialised the USR function in BASIC is disabled and replaced by the more powerful SYS command. (This releases 2 Locs for toolkit use in 0 page.) This is of no detriment to BASIC as the jump vector it uses, starting at $21, which normally contains a JUMP (#4C) instruction, is simply replaced by a RTS (#60) inst.; the following two locs being set to WINDOW default of #0,#18(24 dec). Issueing a USR command in Basic will cause NO action to occur, just as if it was not there. This being one of the most useful commands from the PGM Toolkit, its use is exactly the same and causes a jump to a user M/C subroutine which must end with an RTS (#60) instruction.

Range :- 0<=n<=65535

DEC and HEX$ FUNCTIONS

As before they may both be assigned to their respective variable types and used in a program if required. Note that their arguement must be the correct types and enclosed in brackets.

BASIC EDITOR FOR VDU 80/82 SCREEN

Two methods were considered at the inseption of this part of the package. The first was to make the Editor totally VDU 80/82 based as a "List it then uses cursor control" type editing with information travelling both ways between the Basic and the VDU card. The second method was simply to copy the edit buffer off the Microtan screen and then whenever a change was made to make sure it got recopied back to the VDU screen using the VBUGs built in cursor control command to keep VBUGs cursor in step with Microtans. There are pros and cons for both methods, but the biggest consideration against the first method eventually swayed me towards the second method. For all the video cards speed, we are still with a host CPU that is running at only 750KHz and to do a bit test in order to recognise alpha characters and numbers using a look up table would have slowed things down to an unacceptable level.

So, the second method was settled on. There was still one consideration against this method, but, in the event it proved relatively simple to cure. the Microtan edit buffer works over three lines of the screen and it also uses cursor up and down commands. As we can get a whole line of Basic text on one line of the VDU 80/82 screen, these commands become redundant. However I did mention that we also have to keep the two cursors in step for the editing to be correct. Answer? - Make the Microtan cursor keep going right instead of stopping at the end of the line. ie.go to the start of the next line or go back to the end of the previous line when using cursor right/left etc. This done, it is simply a matter of performing the necessary conversion for cursor control (up & down are not implemented) and then the copy routine whenever a character is changed or deleted.

The result is that when you type in a line number for editing and you press CONTROL E, up pops that line on the VDU 80/82 screen with the cursor flashing away in front of the number as usual. From then on everything is as you have always been used to and ALL commands, except CONT U & CONT D, are valid.

Only two things need to be considered as danger areas. As you have been used to using CONT U & CONT D for so long now and because the EDITOR will still work in the 40 column mode, the temptation to go down to the other half of the line may cause an accidental use of the aforementioned CONTROL codes which will immediately result in the cursor going out of step. There are only two ways out of this problem without causing damage to your programs. One, always use WIDTH1 when editing or two, if already in trouble, hit BREAK/CONTROL C which will abort without damage, whereupon you may re-enter EDIT mode having set up WIDTH1 first !!

One final consideration in this section is that because the Basic Editor is primarily a screen based device, on the Microtan screen, then due consideration will have to be made as to where the cursor is on the screen prior to editing. Again there are a number of solutions but perhaps the best two are either hit CONTROL L first , whereupon all your editing will appear on the top line of the VDU screen, or secondly , LIST a few segments of your program on the screen having done a CONTROL L first, then BREAK out of the listing and go into your chosen line of editing. Both methods ensure that the cursor on the Microtan screen is out of harms way before going into EDIT mode on that screen also. The latter method is prefered as it allows the user to look at the program as it was before editing, whilst he is editing. Finally ESC and LFEED are still valid and will allow stepping thruogh the program as normal, although only the line actually in the edit buffer will be displayed. If the insertion of a letter seems slower than you have been used to, then blame the Microtan 65 not me.

SYSTEM CONFIGURATION

Certain hardware considerations will be discussed. Firstly, if you haven't yet converted your EPROM space on TANEX to RAM yet, then get it done without delay. It's been the subject of a number of articles in the TUG newsletters and is really quite easy and reliable. The reason I single this conversion out is very simple. If your's is still all ROM space then you are not going to be able to take advantage of the next part of the Toolkit, and thats the EDITOR. The reason being that certain routines in the BASIC E0 chip have to be redirected to get the VDU screen to show us what's happening with the EDITOR. These routines are re-vectored during the start-up part of the Toolkit but do NOT affect the overall operation of the Toolkit in any way. Simply, if your E0 chip is not RAM then you will lose the EDITOR facility from the Toolkit. This will not affect the EDITING on the MICROTAN screen at all, which carries on as normal.

Finally, the last hardware consideration to make is that the Toolkit, on startup, interogates the VDU ports (STATUS & COMMAND Registers) which must be at $BE00,$BE01. Further, the VDU card must be plugged in or else the Toolkit will just pass control back to BASIC without any initialisation at all. There's no point in trying to control a board that isn't there, is there! A point of intrest is that the author's system has a very small extra bit of software (about 7 bytes) tagged on to the top of Basic which checks to see if a Toolkit (VDU or PGM) is in residence in RAM at $E800 up, by looking for a #4C byte. If found, Basic automatically initialises any toolkit also! This will be the subject of a future small article in the newsletter as an extension to the XBUG mods. When implimented it makes for a nice tidy start up procedure.

Basic editor functions completely as normal and all normal commands to the editor are valid with the exception of CONTROL U and CONTROL D. What appears on the VDU 80/82 screen is a carbon copy of what is happening on the Microtan screen actually in the EDIT buffer (Micro lines 6,7 & 8) only. Should you wish to fully commit your system to the VDU 80/82 screen in Basic, then it is now possible and is the only way of not using RAM in order to use the EDIT facility. However it will involve permanent changes to the afore mentioned jump vectors in chip at $E000. Further, if these vectors are changed then the VDU 80/82 Toolkit will have to stay in permanent residence too. Watch the newsletter for greater details.

Note:- There is a special edition of the VDU Toolkit for use with the Tandos Disc System - Contact us for further details.

SAMPLE PROGRAMS

```
10 REM SPIRAL BOX
20 POKE34,0:POKE35,232:X=USR(I)
30 #CLS:A=0:B=255:C=510:D=0
40 #MOVEA,B
50 FOR I=0TO255
60 #DRAWA,B
70 #DRAWC,B
80 #DRAWC,D
90 A=A+5:B=B-5:C=C-5
100 #DRAWA,D
110 D=D+5
120 NEXT I
```

```
 1 REM LACE POLYGON
 5 POKE34,0:POKE35,232:X=USR(I)
 6 DIMMX(38),MY(38),DX(38),DY(38)
 10 #CLS:PW=3.141592654/180:R=127:R1=192:C3=256:C4=R:T=-10
 20 FOR J=1 TO 38:T=T+10:L=T*PW:MX(J)=SIN(L)*R1+C3:MY(J)=COS(L)*R+C4:NEXT
 30 U=0:FOR I=1 TO 38:U=U+10:K=U*PW
 35 DX(I)=SIN(K)*R1+C3:DY(I)=COS(K)*R+C4:NEXT
 50 A=1:B=38
 60 FORJ=1TOB:FORI=ATOB:#MOVEMX(J),MY(J):#DRAWDX(I),DY:NEXTI:A=A+1:NEXTJ
 70 GOTO70
```

```
 10 REM SPIRAL PERSPECTIVES
 20 POKE34,0:POKE34,232:X=USR(I)
 30 #CLS:A=0:B=255:C=510:D=0
 40 #MOVEA,D
 50 FOR I=0 TO 255 STEP 5
 60 #DRAWABS(D-C),ABS(B-A):#DRAWABS(D-C),B:#DRAWABS(D-B),D
 70 A=A+5:B=B-5:C=C-5
 80 #DRAWA,B
 100 NEXT I
```

```
LIST

 5 REM VDU DEMONSTRATION
 10 #CLS
 20 PRINTCHR$(27)"[3w"CHR$(27)"[1u"
 30 PRINTCHR$(27)"[4;10cT H E"
 40 PRINTCHR$(27)"[7;8cT . U . G"
 50 PRINTCHR$(27)"[10;8cV I D E O"
 60 PRINTCHR$(27)"[13;6cT O O L K I T"
 70 PRINTCHR$(27)"[16;11cI N"
```

Cont:

```
 80 PRINTCHR$(27)"[19;7cA C T I O N"
 90 PRINTCHR$(27)"[1w"CHR$(27)"[0u":#h:FORI=1TO5000:NEXT
 100 #CLS:GOSUB120:Y=255:FORX=0TO504STEP8:#DRAWX,Y:GOSUB120:NEXTX:GOSUB130
 110 FORX=504TO0STEP-8:#DRAWX,Y:GOSUB130:NEXTX:GOTO140
 120 #MOVE0,0:RETURN
 130 #MOVE504,0:RETURN
 140 GOSUB130:Y=255:FORX=0TO504STEP8:#UNDRAWX,Y:GOSUB130:NEXTX:GOSUB120
 150 FORX=504TO0STEP-8:#UNDRAWX,Y:GOSUB120:NEXTX
 160 PRINT"What did you think of that then! ? All in 6 lines of prog too !!"
 170 PRINT"(and it only took 22 secs, which is about 1 sec slower than M/C)"
 180 FORI=1TO3000:NEXT:DIMX%(1000),Y%(1000)
 190 PRINT"'Ang on a mo while Eric works a few things out....."
 200 PRINT"(It's the BASIC you know... So bloody slow at arithmetic !!)"
 210 FORI=1TO1000:X%(I)=511*RND(I)+1:Y%(I)=255*RND(I)+1:NEXT:#CLS
 220 PRINT" Right. NOW WATCH THIS FOR SPEED !!":#h
 230 FORI=1TO1000:#SETX%(I),Y%(I):NEXT
 240 PRINT" That was 1000 plots using the #SET command.NOW WATCH THEM GO !!":#h
 250 FORI=1000TO1STEP-1:#RESETX%(I),Y%(I):NEXT
 260 PRINT"THAT WAS THEM ALL GOT RID OF USING THE #RESET COMMAND. NIFTY HEY !?"
 270 PRINT:PRINT"BUT YOU AIN'T SEEN NOTHING YET !!"
 280 PRINT"Remember those little routines in the PGM Toolkit manual ?"
 290 PRINT"Well cop a butcher's at the speed of this....!"
 300 FORI=0TO510STEP5:X%(I)=I:Y%(I)=128+100*SIN(6.28*I/255):NEXT
 310 #CLS:PRINT"READY ? Don't blink or you 'll miss it !!":FORI=1TO2000:NEXT
 320 #MOVEX%(0),Y%(0):#CLS:FORI=0TO510STEP5:#DRAWX%(I),Y%(I):NEXT
 330 PRINT"Not bad hey ! Hang on a mo while I get rid of it....yawn !!
 340 FORI=1TO2000:NEXT
 350 #MOVEX%(0),Y%(0):FORI=0TO510STEP5:#UNDRAWX%(I),Y%(I):NEXT
 360 #CLS:PRINT"RIGHT, Let's try the next. (with a little twist !!)"
 370 #MOVE285,150:FORI=0TO63:X%(I)=245+40*COS(I/10):Y%(I)=150+1.25*20*SIN(I/10)
 380 NEXT:FORI=0TO63:#DRAWX%(I),Y%(I):NEXT:#MOVE245,125:#DRAW245,50
 390 #DRAW215,0:#MOVE245,50:#DRAW275,0:#MOVE245,100:#DRAW215,60:#MOVE245,100
 400 #DRAW275,60:#MOVE235,140:#DRAW255,140:#DRAW260,145:#MOVE235,140
 410 #DRAW230,145:#MOVE245,146:#DRAW245,150:#MOVE230,152:#DRAW234,152
 420 #MOVE260,152:#DRAW256,152
 430 #h:PRINT"Yes it's our very own ERIC !! Come to see us !!!                "
 440 FORI=1TO4000:NEXT
 450 PRINT"But enough of this clowning around."
 460 PRINT"Let's go out with a fanfare and.....":FORI=1TO3000:NEXT:#CLS
 470 PRINT"WHAT'S THIS !! Something's coming into land at BAY 16 !!
 480 FORI=1TO4000:NEXT
 490 #MOVE0,12:#DRAW511,12:#DRAW511,10:#DRAW0,10:#DRAW0,12
 500 #MOVE420,12:#DRAW426,66:#MOVE429,95:#DRAW435,155:#DRAW445,155
 510 #DRAW451,88:#MOVE453,66:#DRAW460,12
 520 #DRAW458,12:#DRAW451,66:#MOVE449,88:#DRAW443,153:#DRAW437,153:#DRAW431,94
 530 #MOVE428,66:#DRAW422,12
 540 #MOVE60,170:#DRAW304,170:#DRAW448,80:#DRAW308,80:#DRAW60,170
 550 #DRAW60,173:#DRAW304,173:#DRAW304,170:#MOVE304,173:#DRAW448,83
 560 #MOVE448,88:#DRAW448,70:#DRAW308,70:#MOVE308,80:#DRAW308,70
 570 #DRAW319,66:#DRAW459,66:#DRAW448,70:#MOVE459,66:#DRAW459,83
 580 #DRAW448,88:#DRAW443,88:#MOVE448,80:#DRAW459,75
 590 PRINT"Y E S ! !  IT'S THE.....":FORI=1TO3000:NEXT
 600 PRINTCHR$(27)"[1u"CHR$(27)"[3w"CHR$(27)"[10;11cV D U"
 610 PRINTCHR$(27)"[12;14cB O /"
 620 PRINTCHR$(27)"[14;17cB 2"CHR$(27)"[1w"CHR$(27)"[0u":#h:#Do:#Do
 630 POKE1,0:WAIT1,255:RUN
OK
```

COMMAND SUMMARY

SET x,y - Turns on specified point

RESET x,y - Turns off point specified by SET.

INSET x,y - Turn on or off point.

DRAW x,y - Draws line from current pen position to x,y.

UNDRAW x,y - Erases line from current pen position to x,y.

INDRAW x,y - Performs either Draw or Undraw.

TEST x,y - Reports current status of point.

MOVE x,y - Moves graphics pen to specified coordinates.

CLS - Performs Video clear screen, cursor home.

HOME - Performs cursor home.

UP, DOWN, LEFT, RIGHT. - Cursor movement.

WINDOW lo,hi - Defines scrolling window.

CURS row, col - Defines cursor position.

WIDTH n - Defines character size 1-5.

SPS (Superscript) - Sets SPS mode.

SBS (Subscript) - Sets SBS mode.

LU - Sets underlining mode.

RVS (Reverse Video) - Stes reverse text video.

SYS n - Replacement for USR(I) command.

DEC - Decimal to hex.

HEX - Hex to decimal.

ADDENDUM

RENUMER

#RENUMBER (Start),(Step)

This command renumbers a Basic program from value Start in increments of Step. The parameters Start & Step are optional, and default to 100,10.

Example:-

```
#RENUMBER100,10          ; RENUMBER from 100 steps of 10
#RENUMBER200,10          ; RENUMBER from 200 steps of 10
#RENUMBER200,5           ; RENUMBER from 200 steps of 5
```

All GOTO, GOSUB and THEN statements are correctly reset. If a non-existant number is found, the reference is set to 36767, thus enabling it easily found.

VIDEO 80/82 MODULE - DRIVER ROUTINE

The Video Toolkit uses the Video driver routine present in Tugbug, the routine described in recent newsletter articles. This routine can easily be incorporated in exsisting Tanbugs (Eproms) and is recommended for future use.

If the current Tugbug monitor is in use, entry is assumed to be through the jump vector at location $F82F, this method disregards the flag in byte $C and passes a byte using the following code:-

```
SENCHR: BIT VDUSTA ; CARD STATUS
....... BPL SENCHR ; BRANCH IF NOT READY
....... STA VDUCTL ; PASS BYTE TO VIDEO
....... RTS ...... ; RETURN TO CALLER
```

i.e.

```
FBEB 2C 00 BE BIT $BE00
FBEE 10 FB .. BPL $FBEB
FBF0 8D 01 BE STA $BE01
FBF3 60 ..... RTS
```