TANGERINE USERS GROUP

16 Iddesleigh Road Charminster
Bournemouth Dorset U.K. BH3 7JR
Telephone (0202) 294393

P.G.M. TOOLKIT.

INTRODUCTION

Designing toolkits for the system is a difficult task for any programmer, there being so many of those individual requirements required by the users of the package. We have in this instance bought to you a comprehensive toolkit with the emphasis on graphic capability. The additional facilities we have included as our experience has shown that those offered are the most useful for the user.

Systems analysis has shown that the Microtan 65 has lacked many user defined facilities available to the user, this package may help some way towards a greater all round user friendliness of our chosen machine. It is with this in mind that we have combined all those facilities possible into a given space to bring you the user a useful and viable firmware package.

This is another TUG product that we hope meets with your approval and brings computing to you in a much friendlier atmosphere.

TUG.

The Publishers assumes no responsibility for the use of this Firmware package, nor any infringements of patents or other rights of third parties which would result. No part of this firmware package may be copied, reproduced in any way, without the prior written permission of the publishers. (C) Tangerine Users Group Ltd..... 1982....

INSTALLATION

The P.G.M. Graphic Toolkit has been provided in 2716 Eprom for a resident address on Tanex socket E2 at ram address $E8ØØ - $EFFF.
Insert the Eprom checking for the correct orientation of Pin 1 to socket Pin 1.

Enter Basic in the normal manner and input the following code to activate the Toolkit:

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

The Toolkit will now activate and clear the screen automatically leaving only the 'Prompt' 'OK' in the normal position.

The Toolkit commands are now available until the computer is switched off or Basic is re-initialized.

If you are using the Tanbug 2.3 moniter Resets and Warm starts back to Basic will not effect the Toolkit. However, if you re-enter Basic with the 'Basic' cold start the Toolkit must be activated in the usual manner as above.

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

Note: '£' Denotes 'HASH' sign.

COMMANDS

≠PLOT(X,Y,Ø)

This command will Plot, Clear or Test a point at the coordinates X,Y with the parameter being determined by Ø.

Example:

If $Ø < Ø < 127$	; The point is Set.
$128 \leqslant Ø \leqslant 255$	; The point is Tested.
$Ø = Ø$	; The point is Cleared.

When testing a point, the result may be found by using PEEK(115), which may be found to be Ø if the point was clear, or another value if the point was set.

Example:

```
100 ≠PLOT(X,Y,128): IF PEEK(115)=Ø THEN 2ØØ
110 REM POINT IS SET, TAKE APPROPRIATE
120 REM ACTION
 "
 "
200 REM POINT IS CLEAR
```

In Lo-Res Mode X & Y may range between Ø - 63. (MODEØ)
In Hi-Res Mode X & Y may range between Ø - 255.(MODE1)
The current X & Y pointers are not affected by PLOT.

≠MOVE(X,Y,Ø)

This alters the current X & Y pointers to the specified X & Y. The point is then Set or Tested if Ø is NON-ZERO, but no action is taken if Ø is ZERO (point is not cleared). Note therefore, that if you move out of the permitted range for the mode you are in, then an ILLEGAL QUANTITY ERROR will not be generated unless the point is plotted.

Example:

Whilst in MODEØ (Lo-Res), where the limit of X & Y is 63.

```
≠MOVE(100,100,1)              ; Will generate ILLEGAL QUANTITY ERROR
≠MOVE(100,100,Ø)              ; Will not generate an ILLEGAL QUANTITY
                              ; ERROR as the Plot routine is not
                              ; called.
```

The current values of X & Y may be determined by PEEKING locations locations 242 & 243.

Example:

```
X = PEEK(242): Y = PEEK(243)
```

≠DRAW(X,Y,Ø)

DRAW or UN-DRAW the best straight line from the current X,Y to specified X,Y, according to Ø.
The current X,Y become the specified X,Y. The line becomes DRAWN or UN-DRAWN according to the value of Ø (see ≠PLOT).

Note that to ensure a previously drawn line is completely un-drawn, the UN-DRAW instruction must go in the same direction as the original DRAW instruction.

Example:

```
≠MOVE(X1,Y1,1)                ; Move to start position.
≠DRAW(X2,Y2,1)                ; Draw line.
≠MOVE(X1,Y1,Ø)                ; Move back to start.
≠DRAW(X2,Y2,Ø)                ; Erase line.
```

If $128 \leqslant Ø \leqslant 255$ (i.e. Test Point) then location 115_{10} will contain the status for the endpoint of the line.

≠MODEN

Selects Hi-Res or Lo-Res mode for the ≠PLOT, ≠DRAW & ≠CLR instructions.

```
If N = Ø                      ; Lo-Res chunky graphics are selected.
If N = 1                      ; Hi-Res 256x256 graphics are selected
                              ; provided a P.G.M. module is fitted
                              ; memory mapped to $84ØØ - $87FF.
```

Any other value of N will generate a mode error, as will MODE1 if the P.G.M. is not fitted.

In MODEØ, X & Y values for ≠PLOT & ≠DRAW routines must be in the range of Ø - 63, else an ILLEGAL QUANTITY ERROR will result. Note that text mode will be selected before the error message is sent.

The mode selected affects the operation of the ≠CLR routine as follows:

<u>≠MODEØ</u>	; ≠CLR will clear the screen to graphics ; mode and set the current X & Y to Ø. ; Graphics mode is left set on exit.
<u>≠MODE1</u>	; ≠CLR will clear the screen to Text ; mode, sets current X & Y = Ø

<u>≠CLR</u>

Clears the screen to the appropriate state, according to the mode selected. See discussion on above.

<u>≠CPGM N</u>

Clears the P.G.M. ram to Ø starting from the character N, up to but not including the Cursor 'FF' ($87FØ - $87FF). The P.G.M. character pointer is set to N-1.

In order to demonstrate the use of this command, we must examine how the Hi-Res ≠PLOT & ≠DRAW routines work.

When a point is to be plotted using Hi-Res, the plot routine first determines which of the screen memory locations is to be affected, it then examines that location to see if it already contains a P.G.M. character. (i.e. A value of 80H - 9FH or EØH - FEH)

If it is an existing P.G.M. cell, then the plot routine modifies that cell in order to show the point plotted.

If it is not a P.G.M. character, then a new P.G.M. cell is defined with the appropriate values to show the plotted point, and the P.G.M. character pointer is incremented. In this way, new P.G.M. characters are only used when necessary to make the best use of the 63 individual cells which may be defined. (excluding the cursor)

When the plot routine runs out of P.G.M. cells (i.e. the pointer gets to FEH) then further plot instructions will be ignored until the pointer is reset.

By providing this parameter N to the ≠CPGM command, the user may reserve a block of of P.G.M. cells for his own use, as they will not be used by the ≠PLOT or ≠DRAW routines.

N may range between Ø - 63 i.e. the software deals with correction into the 8ØH - 9FH & EØH - FEH ranges.

User defined P.G.M. cells should start at the bottom of the P.G.M. table and work upwards.

```
Example:                   ; Reserve 1 cell at $84ØØH-$840FH
#CPGM1                     ; for user.  Initialize remainder
                           ; for ≠PLOT & ≠DRAW.
```

The user defined cell may be set up and displayed by:

```
PRINT CHR$(128)
or
POKE(Screen Addr),128
```

No matter what the value of N, ≠CPGM N will always set the current X & Y to Ø.

The current value of the P.G.M. character may be found in location EEH, 238_{10}.

Example:

```
CP=PEEK(238)
```

Note that the pointer is at the last character used - so for example, if ≠CPGMØ has just been executed, then CP as above would be FFH,255_{10}.

The number of cells left for Hi-Res use may be found in from Basic as follows.

```
10 CP=PEEK(238):IF CP=255 THEN CP=-1
20 NC=62-CP
```

and NC will contain the number of cells left.

CURSOR CONTROL GROUP

Six commands are provided to move the cursor around the screen to assist with varying display formats.

```
≠UP                          ; Cursor UP
≠DOWN                        ;   "    DOWN
≠LEFT                        ;   "    LEFT
≠RIGHT                       ;   "    RIGHT
≠HOME                        ;   "    HOME Top Left
≠NRM                         ;   "    NORMAL Bottom Left
```

Note that these must be executed as statements and cannot be built up into strings as on the Pet. For example:

Whilst using these, terminate all print statements with a semi-colon, as otherwise spurious carriage returns will destroy the carefully designed screen layout!!.

Also, Tanbug version 1 thinks that the Cursor is always on the bottom line, and so will scroll the display at the end of each printed line.

If your cursor happens to be on the top line, then all printed output will disappear at great speed !!.
Use ≠NRM to overcome this before exiting from programs.

≠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 1ØØ,1Ø.

Example:

```
≠RENUMBER                   ; RENUMBER from 1ØØ in steps of 1Ø
≠RENUMBER 200,10            ; RENUMBER from 2ØØ in steps of 1Ø
≠RENUMBER 200,5             ; RENUMBER from 2ØØ in steps of 5
```

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

≠OFF

This command sets TEXT mode. In the case of a program crash this command may be entered from the keyboard to return the screen to normal TEXT mode. ≠ (CR) will achieve the same effect. Overall, it simply replaces the need for the POKE49139,Ø statement during system crash.

≠SYS N

Executes users machine code routine starting at N. This saves using:

```
POKE34,N-INT(N/256)*256:POKE35,INT(N/256):X=USR(I)
```

Return from user machine code routine with RTS as for USR routines. There is no need for the users code to save any registers, but the stack pointer must be at the same level on entry as on exit, else VERY unpredictable results may occur.

Example:

```
≠SYS61432                   ; JSR to E.S.C. Driver software.
```

System responds with Eprom Start etc, returns to Basic command when completed users machine code routine.

GENERAL NOTES ON COMMAND USAGE

All commands may be shortened by typing the last letter in the lower case mode.

Example:

```
≠RENUMBER
≠REn
≠UP
≠u
≠DOWN
≠Do
```

You must be aware of ambiguities, i.e.≠UP may be shortened to ≠u simply as no other command starts with ≠u, but ≠DOWN can only be shortened to ≠Do as ≠DRAW also starts with ≠D.
The extra commands cannot be used immediately after the THEN statement, some innocuous statement such as: 'X=X' must precede the command.
Example:

```
IF (condition)THEN≠(command)....
```
will generate a ?SYNTAX ERROR.
True statement:
```
IF (condition)THENX=X:≠(command)
```
will execute correctly.

FUNCTIONS

Two functions are provided for conversion between the Hexadecimal Strings and Decimal numbers & vice versa.

```
N=≠DEC(String)
```

This will convert the Hex characters in string to a decimal number in N. If the string has more than four characters, only the last four characters will be converted. If the string contains a non Hex digit (which includes spaces), a ?TYPE MISMATCH ERROR will result.

```
A$=≠HEX$(N)
```

This will convert N into a four character string in A$. If N is < Ø or > 65535 an ILLEGAL QUANTITY ERROR will result.

Example:

```
A$="F":PRINT≠DEC(A$)
```
will print 255.

```
N=255:PRINT≠HEX$(N)
```
will print ØØFF

Note that the ≠DEC function uses 2's complement arithmetic, i.e. ≠DEC("FFFF") returns -1, not 65535. Neither of these functions may be shortened, however they may be built up into larger expressions under one condition, they MUST be the last term in the expression.

Example:

```
N=2*≠DEC(String)
```
will work,

```
N=≠DEC(String)*2
```
will generate an error.

```
A$="H"+≠HEX$(N)
```
correct,

```
A$=≠HEX$(N)+"H"
```
error.

SAMPLE PROGRAMS

Low resolution (Chunky Graphics)

```
10 ≠MODEØ: ≠CLR
20 ≠DRAW(63,63,1): ≠MOVE(Ø,Ø,Ø)                    (A)
30 FOR I = 1 TO 2ØØØ: NEXT
40 ≠DRAW(63,63,Ø): ≠OFF
```

If you entered the program correctly it should have drawn a straight line across the screen from bottom left to top right. Without more to do, enter the program below.

```
10 ≠MODE1: ≠CLR: ≠CPGMØ
20 ≠DRAW(255,255,1): ≠MOVE(Ø,Ø,Ø)                  (B)
30 FOR I = 1 TO 2ØØØ: NEXT
40 ≠DRAW(255,255,Ø)
```

The high resolution capability of the P.G.M. is shown in contrast to that of the chunky graphics, quite startling. These programs provide useful information as to the simplicity of graphic control when using the Toolkit. Enter the programs below.

```
10 ≠MODEØ: ≠CLR
20 FOR I = Ø TO 63                                 (C)
30 ≠PLOT(I,3Ø+3Ø*SIN(6.28*I/63),1)
40 NEXT: GETA$
50 ≠OFF
```

Interesting ?, try changing SIN into COS, likewise below.

```
10 ≠MODE1: ≠CLR: ≠CPGMØ
20 FOR I = Ø TO 255                                (D)
30 ≠PLOT(I,128+127*SIN(6.28*I/255),1)
40 NEXT: GETA$
```

These programs can be extended further to include keyboard input for values etc, a summary of their operation is given overleaf.
If the Low res programs crash whilst running simply enter ≠ (CR) as this will return TEXT mode.

DRAW A CIRCLE

```
10 ≠MODE0:INPUT"RADIUS";R          ; Lo-Res
20 IF R> 24 THEN 10
30 ≠CLR:≠MOVE(30+R,30,1)
40 FOR I=0 TO 6.3 STEP.1
50 ≠DRAW(30+R*COS(I),30+1.25*R*SIN(I),1)
60 NEXT:≠OFF
70 GETA$
```

```
10 ≠MODE1:INPUT"RADIUS";R          ; Hi-Res
20 IF R> 80 THEN 10
30 ≠CLR:≠CPGM0:≠MOVE(128+R,128,1)
40 FOR I=0 TO 6.3 STEP.1
50 ≠DRAW(128+R*COS(I),128+1.25*R*SIN(I),1)
60 NEXT:GETA$
```

RANDOM LINES

```
10 ≠MODE0:≠CLR                     ; Lo-Res
20 X=INT(RND(1)*64):Y=INT(RND(1)*64:
   Z=INT(RND(1)+.5)
30 ≠DRAW(X,Y,Z):GOTO 20
```

Stop with CTL-C and use ≠OFF in direct mode to reset text mode.

```
10 ≠MODE1:≠CLR:≠CPGM0              ; Hi-Res
20 X=INT(RND(1)*100):Y=INT(RND(1)*100):
   Z=INT(RND(1)+.5)
30 ≠DRAW (X,Y,Z):GOTO 20
```

These routines simply demonstrate the plotting capabilities of the Toolkit, although slower than machine code for display, it can show the user the format required for the plotting routines. The user is welcomed to alter the values for experimentation and effects.

TOOLKIT MODIFICATIONS

The Tookit routines may be used under machine code programming, however there are some conditions to be met and are as follows:

≠PLOT (Lo-Res)

```
JSR $EA43                  ; With X in $71
                           ;      Y in $72
                           ;      Ø in $73
```

If X,Y, or Ø are out of range the Basic interpreter error handling routines are used, with spurious results.

≠PLOT (Hi-Res)

```
JSR $E96F                  ; As above.
```

The P.G.M. character pointer at $EE must be initialized before using Hi-Res plotting - see discussion of ≠MODE N command.

```
≠MOVE                      ; Store the new X & Y into
                           ; $F2 =X,   $F3 = Y.
```

≠DRAW

Set up a vector to the appropriate plot routine for this. The vector is at $F4, $F5, $F6 and should be set to:

```
JMP $EA43                  ; Lo-Res
JMP $E96F                  ; Hi-Res
```

A JSR $EA88 will draw a line from the current X,Y $F2,$F3 to the new X,Y $71,$72.

ON/OFF is in location $73 as usual.

≠CLR

```
JSR $E956                  ; Lo-Res
JSR $EB2E                  ; Hi-Res
```

≠CPGM N

```
JSR $EDD6                  ; With N in .X.
```

Note ≠PLOT caution on error handling.

```
≠UP                         ; JSR $EBB5
≠DOWN                       ; JSR $EBCØ
≠LEFT                       ; JSR $EBD2
≠RIGHT                      ; JSR $EBCE
≠HOME                       ; JSR $EBF9
≠NRM                        ; JSR $ECØ8
```

All Cursor control routines use all the registers.

<u>Eprom Storage Card</u>

```
JSR $EFFØ                   ; Option 1.
    $EFF3                   ; RTS

JSR $EFF4                   ; Option 2.
    $FFF7                   ; BRK

JSR $EFF8                   ; Option 3.
    $EFFB                   ; RTS

JSR $EFFC                   ; Option 4.
    $EFFF                   ; BRK
```

See E.S.C. manual for detailed discription on use.