SUPER SPRITE
****

Congratulations on buying Super sprite. We are sure it will greatly enhance the games you write on your computer.

CASSETTE LOADING INSTRUCTIONS
****

1) Type LOAD and press the RETURN button.
2) Press play on tape as requested. Loading will be automatic.
3) To use the DEMO program or the Sprite Definer, Super Sprite must already be loaded.
4) DEMO or Sprite Definer may be loaded as outlined in 1 and 2 above.

DISC LOADING INSTRUCTIONS
****

1) LOAD "SUPER.C16",8,1 or "SUPER.PLUS4",8,1 - for super sprite
2) LOAD "DEMO",8,1 - for Demo program
3) LOAD "DEF",8,1 - for Sprite definer
4) To use Demo or Sprite definer, Super sprite must be loaded first.

Introduction
****

Super sprite allows you the control of up to eight sprites under BASIC control. Each sprite is 16 pixels by 16 pixels and can move over or under characters on the text screen. The sprites can be any one of the 121 colours available to you on the C16 / PLUS4. They can even be set to flash!!!

The sprites operate on a sprite plane which is 352 pixels wide and 232 pixels high. The screen resolution of the C16 / PLUS4 is 320 by 200. The extra range on the sprite plane is an invisible band around the screen. It is very useful for hiding sprites or achieving the effect of a sprite gliding off the screen.

With a little practice you will be able to achieve some VERY impressive results. Our demonstration program will give you some good ideas of what can be achieved.

The following notes will describe to you how to use your Super sprites

The extended Basic command set
****

1) SPRAK. This command has only one parameter. It is used to choose how many sprites you would like to have active at any one time. A value of nought deactivates all sprites.

```
eg:  SPRAK 2  ;Enables two sprites.
              ;Sprites number 1 and 2.
```

If at any time you reset your computer by holding down RUN/STOP whilst pressing RESET you can reactivate Super sprite using SPRAK. Unfortunately if you RESET your computer by pressing RESET alone you will have to re-load Super sprite.

2) SPRP. (SPRite Position) - This command has five parameters as shown below:

```
SPRP [Sprite No.],[Xpos],[Ypos],[Xinc],[Yinc]
```

(a) Sprite no. - This governs which sprite you are communicating with. Values - 1 to 8.
(b) Xpos - This tells the sprite how far ACROSS the screen it should be. Values - 0 to 351
(c) Ypos - This tells the sprite how far UP the screen it should be. Values - 0 to 231.
(d) Xinc and Yinc - These parameters govern how fast the sprites move and in what direction. A value of 0 in both of the parameters will result in no sprite movement. Values between 1 and 127 will result in movement of the sprite (i) To the right - Xinc. (ii) Downwards - Yinc. Values greater than 128 and up to 255 result in reverse movement. i.e. Left or up.

```
eg: SPRP 1,100,100,0,0 ;Sprite 1 will be stationary at 100,100.

    SPRP 3,100,100,1,0 ;Sprite 3 will appear at 100,100 and glide to
the right.

    SPRP 1,50,50,0,129 ;Sprite 1 will sapper at 50,50 and glide up the
screen.
```

3) SPRC - (SPRite Colour)

```
 SPRC. [sprite no.],[colour],[over/under text]
```

The first parameter [sprite no.] determins which sprite we are communicating with. Values - 1 to 8.

The second parameter selects the colour of the sprite. Values - 0 to 255.

The colour of the sprite is calculated thus: The value of the colour required plus eight times the luminocity required (plus 128 if you wish the sprite to flash.)

-- Colour Values --

Color No.	Colour	Colour No.	Colour
0	Black	8	Orange
1	Grey	9	Brown
2	Red	10	Yel/Grn
3	Cyan	11	Pink
4	Purple	12	Blu/Grn
5	Green	13	Lght Blue
6	Blue	14	Dark Blue
7	Yellow	15	Lght Grn

The third parameter can have one of two values. It can either tell the sprite to go over text or under text. - 0 or 1 respectively.

```
eg: SPRC 1,2,0        ;Sprite 1 will appear dark red and go over text.
    SPRC 1,0,1        ;Sprite 1 will be black and go under text.
```

4) SPRB - (SPRite Bump) - Collision detection.

SPRB [sprite no.],[variable]

This command returns the result of collisions between sprites and their enviromnment. It has two parameters, the first is the number of the sprite to be questioned and the second a variable into which the collision status is to be put.

eg: SPRB 5,BUMP ;Sprite 5's collision status will be stored in the variable 'BUMP'.

The data returned from the command can be interpretted according to the table below.

-- Sprite Collisions --

Value returned.	Meaning.
0	;No collisions.
1	;Hit text.
2	;Sprite off screen.
4	;Hit text and the sprite is off screen.
8	;Hit another sprite.
16	;Hit text and a sprite.
32	;Hit sprite and off screen.
64	;Hit sprite and text and off screen.

Sprite definition. (The shape of the sprite)

****

A sprite definition command was found to be clumsy and so it was left to the user to decide how he/she prefered defining sprites. The sprite data is stored on the Plus 4 from $F000 to $F0FF (61440 to 61695) and on the C16 from $3700 to $37FF (14080 to 14335). It is arranged in eight blocks (one for each sprite) of thirty-two bytes each.

It is easiest to define sprites using the SPRITE DEFINITION PACKAGE provided FREE with this program. For those who know how to define characters on an 8 X 8 grid and wish to define their sprites by hand the procedure is outlined below.

Any sprite is defined on a 2 character by 2 character grid. The numbers are stored starting with the eight numbers for the top left character, followed by the eight numbers for the bottom left character, and then the numbers for the top right and bottom right characters. This makes a total of thirty two bytes.

eg: To turn sprite 1 into a sprite with a solid square at the top left and bottom right corners and blank in the other two corners on the PLUS4.

```
10 FOR X = 0 TO 31
20 READ A:POKE 61440+X,A (On the C16 - 20 READ A:POKE 14080+X,A)
30 NEXT X
40 DATA 255,255,255,255,255,255,255,255; Top left of sprite.
50 DATA 0,0,0,0,0,0,0,0                ; Bottom left of sprite.
60 DATA 255,255,255,255,255,255,255,255; Top right of sprite.
70 DATA 0,0,0,0,0,0,0,0                ; Bottom right of sprite.
```

NOTES:

1) Plus 4 owners may still use the High resolution screen with Super sprite in memory. They must, however, execute a SPRAK 0 before activating the graphic screen and must execute a GRAPHIC CLR before reactivating Super sprite. Obviously due to shortage of memory C16 owners may not use the High resolution screen and the GRAPHIC command has been removed from their BASIC vocabulary by Super sprite. N.B. Sprites can not be seen on a high-res screen.

2) The text screen should not be scrolled with the sprites active and we recommend you deactivate sprites before editting your programs with a SPRAK 0 to prevent them interfering with the editing of programs.

3) To implement Super sprite a redefined character set has been used, and this is positioned at $3800 (14336) to $4000 (16384) on the C16 and at $E800 (59392) to $F000 (61440) on the Plus 4. The character set does not have reverse field characters and instead lower case characters can be obtained in their place. The characters from 128 to 199 (ASCII values) should not be used as they are exclusively for the sprite programs use. If you wish to redefine the character set then feel free as Super sprite will cope with it as long as you use the section of memory outlined above.

4) Sprite extended BASIC commands may be used in exactly the same way as normal commands with variables and complex arithmetic expressions being allowed as their perameters with two exceptions;

1) An extended basic command may not be the first command after an ELSE statement.

```
eg: IF X=1 THEN SPRB 1,A:ELSE SPRP 1,X,Y,0,0         ; Is not allowed.
    IF X=1 THEN SPRB 1,A:ELSE A=A:SPRP 1,X,Y,0,0     ; Is allowed.(A=A
```

is simply a dummy command to space the sprite command from the ELSE command.)

2) An extended basic command may not come immediately after a REM, otherwise it will be executed. To prevent this the command should have a charater between it and the REM.

```
eg: REM SPRAK 4      ;Will activate 4 sprites.
    REM * SPRAK 4 *  ;Will NOT activate 4 sprites.
```

Notes for advanced users:

If you wish to manipulate the sprites directly from machine code then there follows a list of locations you may find usefull.

```
$5F5        ;No of sprites active.
$5F6 - $5FD ;High byte of X-coor.
$5FE - $605 ;Low byte of X-coor.
$606 - $60D ;Y-coordinate.
560E - $615 ;Colours.
$616 - $61D ;Y increment.
$61E - $625 ;X increment.
$626 - $62D ;Priority. (Only:$11 or $51)
$62E - $635 ;Collision registors.
```

Use of MERLIN with Super sprite.

Merlin (A Plus4 or C16 assembler sold seperately by Wizard.) may be used with Super sprite, using the following technique:

1) Load Super sprite, then type: POKE 1160,10 <RETURN>.
2) Load Merlin.

Super sprite and Merlin are now in memory, you will find that the basic commands which manipulate sprites have been deactivated, however, the sprites can still be manipulited using the memory locations listed above. After a RESET whilst holding down RUN/STOP sprites can be reactivated by a JSR $640.

NOTE: $5F5 should contain 0 if source code is assembled by Merlin or the computer may crash.

Memory organisation with Super sprite.

Plus 4:

Top of memory: $E7FF   (59391)
Character set: $E800   (59392)
Sprite data : $F000   (61440)
Spare !!??!! : $F100   (61696)
Super sprite : $F400   (62464)
              To $FCFF   (64767)

Low mem.block: $05F5   (1525)
              to $06EB   (1771)

C 16:

Top of memory: $2E9F   (11935)
Super sprite : $2EA0   (11936)
              To $36FF   (14079)
Sprite data : $3700   (14080)
Character set: $3800   (14336)
              To $3FFF   (16383)

Low mem.block: $05F5   (1525)
              To $06EB   (1771)