SuperBASIC 64

Martin C. Kees

How would you like to be able to access 37 valuable new commands when you're programming on your 64? SuperBASIC adds sprite, color, graphics, sound, and memory management features and also enhances eight of BASIC'S own commands. And it's designed to work as easily and as quickly as any ordinary BASIC instruction. Once you try it, you'll wonder how you programmed without SuperBASIC — it's an especially valuable addition to any 64 owner's library of programs. As a bonus, there's also a PET emulator and several demonstration programs so you can see SuperBASIC in action.

SuperBASIC adds commands to BASIC using a special technique. BASIC is automatically copied to its matching RAM and modified to change the STOP command to a wedge vector (similar to Apple's ampersand (&) wedge). The character chosen was the left bracket ([). Then, using four-letter mnemonics following the wedge character, you can select what you want SuperBASIC to do.

These machine language routines make it very easy to control virtually all the VIC-II chip special features. Sprites and hi-res graphics can be controlled from BASIC without having to POKE or use Boolean functions to enable special graphics modes. Since BASIC was moved to RAM to implement the [wedge, this made it convenient to enhance a few BASIC commands. I added the use of variable expressions for GOTO and GOSUB, and RESTORE by line number. These changes to BASIC in RAM don't slow execution as they would have if CHRGET wedging techniques had been used.

SuperBASIC Command Format

The commands can be used in both direct or program mode. The general format is [xxxx <exp>,<exp> where xxxx represents the four-character mnemonic and <exp> is a number, variable, or a valid BASIC expression. When a color is selected, use the standard value ordinarily POKEd to the VIC chip. I have used the same coordinate system for sprite positions as given in Commodore documentation. The hi-res upper-left corner is 0,0, and the lower-right is 319,199. Commands that switch a function on or off use 0 for off and 1 for on.

SuperBASIC includes two types of changes to normal BASIC: enhanced commands and new commands. Enhanced commands include GOTO and GOSUB and variants with IF and ON. You can use a line number expression for these commands. This can help in program readability, allowing constructions such as GOTO KEY where KEY = 1000. This would transfer control to line 1000. RESTORE can also be followed by a line number expression. RESTORE KEY would cause the next READ to use the first DATA statement encountered at or after line 1000. This allows DATA statements to be selected under program control. Small files could be maintained in DATA statements and accessed by line number. When LISTing a program, the SHIFT key pauses the list until released. The ASC function will return a value of zero for null strings.

The new commands can be divided into five categories: sprite, sound, color control, VIC memory mapping, and graphics control. A convenience command [CATA ('catalog') is also included. This lists to the screen all mnemonics defined in SuperBASIC.

Loading The Program

You must reset memory to its normal conditions before LOADing and RUNning SuperBASIC. You can do this by turning the computer off and back on, or with the command `SYS 64738`. When you

RUN the SuperBASIC loader, it first copies BASIC from ROM into the underlying RAM and makes modifications to certain commands. Then it copies the machine language for the rest of the Super- BASIC routines into memory at $C000 - $CC00. No other machine language subroutines which use memory starting at $C000 can be used with SuperBASIC 64, but the DOS Wedge program can be used without conflict. The loader erases itself from the BASIC memory area after it is RUN.

The SuperBASIC commands will be enabled until you hit RUN/STOP-RESTORE or `POKE 1,55`. Once loaded, SuperBASIC can be reenabled with `POKE 1,54`. The programs you write with Super-BASIC commands are loaded and saved in the normal manner. The only conflict with normal BASIC is the use of the STOP command. It is not available; use END instead. When SuperBASIC commands are listed while SuperBASIC is disabled, the [character will print as STOP.

Sprite Commands

[DSPR s,blk,xexp,yexp,x,y,multi,col,mc0,mc1
[MOVE [KSPR [ESPR [BSPP

These commands are used in defining sprite characteristics and controlling sprite movement. [DSPR (Define Sprite) is a general setup command that initializes a sprite for the VIC-II chip. The ten arguments in the parameter (see SuperBASIC commands at the end of the article) specify most of the options available for sprite control. [DSPR enables the selected sprite (numbered 0-7), stores block (blk) address in current screen pointer table, expands if xexp or yexp = 1, determines initial display position (xpos,ypos), and sets sprite color registers. Multicolored sprites are selected by setting multi = 1, single color by multi = 0. Mc0 and mc1 are optional arguments in the list which set up multicolor 0 and 1. [MOVE moves the selected sprite to xpos,ypos. **Horizontal values greater than 255 are handled automatically.** [KSPR and [ESPR kill or enable the selected sprite. [BSPP sets background/sprite priority for the selected sprite (sel = 1 sets background in front of sprite).

Sound Commands

[SSND voice, ad, sr, wave, freq, pwidth
[PLAY 256*wave + voice, freq, pwidth

These commands access some of the features of the SID chip. [SSND (Set up sound) produces a sound from one of the three voices of the SID chip. Voice (1-3) selects the voice, ad and sr control the attack/decay and sustain/release registers of the selected voice. Wave controls the waveform, gating, and special effects functions of the sound chip. Wave, ad, and sr use the same values that would normally be POKEd to these registers. Freq controls the frequency of the voice but is a 16-bit value in the range 0 - 65535. Pwidth is the pulse width value for the pulse waveform and is needed only when wave = 65. Pwidth is an 11-bit value in the range 0-12228. [SSND sets the volume register to 15. [PLAY is a short form of [SSND that assumes AD/SR values have been set previously. Waveform and voice values are coded into the first parameter argument by wave*256 + voice. Freq and pwidth are used the same as in [SSND.

Value	Attack Rate	Value	Decay/Release Rate
0	2 ms	0	6 ms
16	8 ms	1	24 ms
32	16 ms	2	48 ms
48	24	3	72
64	38	4	114 ms
80	56	5	168
96	68	6	204
112	80	7	240
128	100 ms	8	300
144	250	9	750
160	500	10	1.5 s
176	800	11	2.4 s
192	1 s	12	3 s
208	3 s	13	9 s
224	5 s	14	15 s
240	8 s	15	24 s

Value	Wave Type	Off
17	Triangular	16
33	Sawtooth	32
65	Pulse (also uses pwidth)	64
129	White Noise	128

VIC Color Control

[BKGD [BKG4 [EXTC [FCOL

These commands control background, border, and text character color. [BKGD sets the background to the selected color. [EXTC sets the exterior border color to the selected color. [BKG4 sets all four background color registers (used in extended color and multicolor bitmap modes). [FCOL (fill color memory) fills the color memory block with the selected color. This causes all text on the current screen to be displayed in the selected color. [FCOL is also useful in multicolor bitmap mode to set multicolor pixels.

VIC Memory Mapping

[BANK [VS1K [CB2K

The VIC chip views memory differently than does the 6510 chip. VIC sees only 16K at a time and maps the ROM character set into part of this 16K bank at times. These commands allow changes to the normal locations of the screen and character sets. [BANK selects which one of four banks (0 - 3) the VIC chip sees. Normally this is bank 0. [BANK resets the pointer BASIC uses to locate the screen. [VS1K determines which 1K block of the 16 available is used for the text screen. The blocks are numbered 0 - 15. The BASIC screen pointer is reset for this location. [CB2K controls which 2K block of the 8 available is used for the character set. In banks and 2 the ROM set is located at 2K blocks two and three. [CB2K is also used to select which 8K block is used for the bitmap screen, values 0-3 select the lower 8K block, and values 4-7 select the upper 8K block. These three commands must be used in coordination to smoothly relocate the screen. Caution must be exercised in selecting locations since a system crash will result if the screen overwrites important RAM such as page zero. Banks 2 and 3 must be used with great care. (More on bank 3 usage later.) Program 6 demonstrates relocation to PET standard locations for the screen and BASIC.

Graphics/Text Control

[ECGR [MCGR [BMGR

These commands select extended color, multicolor, or bitmap graphics modes. A value of turns the mode off and a value of 1 turns the mode on. Only multicolor and bitmap work in conjunction with each other to form a combined mode. When extended color and bitmap are both on, the screen will appear blank. This effect might be useful for temporarily hiding the screen.

[MXGR reg,mask,rast1,val1,rast2,val2
[KMXG
[CMXV val1, val2

These commands set up a simple interrupt routine that allows mixed modes to appear in two sections of the screen. [MXGR will change the contents of one VIC register (reg) or part of its contents (the bits OFF in mask) each time the raster counter register equals one of the two raster select values (rastl and rast2). The values in val1 or val2 will be stored into the selected VIC register. You must determine the appropriate value for the particular register. For example, [MXGR 33,240, 152,6,252,1 will cause screen lines 51 to 151 to be displayed with background white and lines 152 to 251 with background blue.

The visible portion of the screen extends from raster 51 to raster 251. [KMXG will kill the interrupt and leave the selected register in an unknown state. [CMXV (change mixed-mode values) allows changing val1 and val2 while mixed mode is in force. By setting them equal, a known state will be in effect after [KMXG. The interrupt routines are simple in that normal IRQ still occurs (keyboard scan, clock update, etc.) so that the transition will tend to creep. To keep the change precise, you must disable interrupts from the CIA. This will kill the keyboard, however, so I/O would be limited to joystick ports only.

[SIZE [XYSC

These commands help use the smooth scroll registers of the VIC chip. [SIZE selects 40 or 38 columns for the text display chosen by setting colsel to 1 or (colsel = 1 selects 40 columns) and sets number of lines to 25 or 24 (rowsel = 1 selects

25 lines). [XYSC moves the entire text screen up to seven pixels horizontally or vertically. By setting xpos and ypos to a value in the range 0-7, the screen can be stepped a pixel at a time to produce a smooth scroll. When used in conjunction with a machine language scroll routine or the automatic scroll up, text can be scrolled smoothly across or up the entire screen.

[DECS

[DLCS (download character set) assists in using banks without ROM character set images and in designing custom character sets. You can copy the uppercase graphics set, upper- and lowercase set, or both by setting set equal to 0, 1, or 2 respectively. This is followed by the address of the first location in memory where you wish the ROM set to be positioned. This should be on a 2K boundary unless you wish to change the order of the set. When the address is 53248, the set will be copied into the RAM beneath the ROM set for use in bank 3.

[FBMS [FSCR

The current hi-res screen (determined by the last [CB2K command) can be filled with any byte value with [FBMS (fill bitmap screen). [FBMS would clear the entire 8K screen. [FSCR works in a similar way with the current text screen. The entire screen is filled with a byte value. Since the text screen is used for color control in hi-res mode, [FSCR can be used for hi-res color control.

[PLOT [FLIP [CLPX [MCPL

These commands are used in plotting pixel points in hi-res graphics modes. The first three plot in 320 X 200 resolution two-color mode, the last in 160 x 200 resolution four-color mode. [PLOT sets the selected pixel on, [CLPX turns the pixel off, and [FLIP changes the pixel to the opposite state. [MCPL (multicolor plot) accepts horizontal coordinates in the range 0,159 and plots in one of four colors determined by sel, with sel in the range 0,3. A value of selects background color, 1 selects text screen low-byte color, 2 selects text screen high-byte color, and 3 selects color memory color. Before you execute any of the plotting commands, [CB2K must be used to select the appropriate 8K block and [BMGR 1 must be in force for the plot to be seen. Remember that y coordinates increase as you go down the screen.

[DRAW

[DRAW is used to draw line segments on the hi-res screen. [CB2K and [BMGR must be used in preparation as in plot commands. [DRAW connects the endpoints given in the parameter list. The line is drawn from xl,yl toward x2,y2.

[HRCS [CHAR [CHRX [CODE

These commands make it easy to put text on the hi-res screen. [HRCS (hi-res character set) stores the address of the character set to be used. It need not be located on a 2K boundary or even be the same set as used on the text screen. The address given is of the first byte of the set. A value of 53248 will select the ROM set (upper/graphics). [CHAR and [CHRX plot an 8 x 8 character to a selected position on the current hi-res screen. The character code (char) to select which character to plot corresponds to the screen POKE codes as listed in Commodore documentation. Example: [CHAR 1,100,100 would plot the letter A with position 100,100 being the upper-left corner of the 8 X 8 character ceil. [CHAR plots tiie cell to the hi- -res screen absolutely while [CHRX uses the exclu- sive OR function to flip the cell pixels. So [CHRX can be used to unplot a previously plotted charac- ter. [CODE helps in translating to the screen POKE code used by [CHAR and [CHRX in character selection.

The argument for [CODE must be the name of a defined string variable. Upon execution the ASCII values stored in the string will be converted to screen POKE codes. The RVS ON and RVS OFF control characters can be used within the string to select the upper 128 or lower 128 charac- ters of the set. All other control characters will produce unpredictable results. Once the string is converted using [CODE, use the ASC function and MID$ function to read the codes. The ASC function will give correct results for the character of the set. Be

careful when using strings not built to high memory because [CODE will modifv the actual string data stored within the BASIC text area.

[HRAM [LOOK [STUF

These commands make use of [BANK 3 possible from BASIC. When bank 3 is selected, the VIC chip uses RAM in the 64 from $C000 to SFFFF and ignores ROM located at the same addresses, including the ROM character set. SuperBASIC allows the location of one text screen ([VS1K block 3 located at $CC00) in bank 3. RAM from $0000 to $FFFF can be used for character sets, sprites, and a hi-res screen. The main problem confronting the bank 3 user is the switching required to read and write to these RAM locations. All plotting commands need to read as well as write to RAM so they can be preceded by [HRAM to accomplish this in bank 3. For example, [HRAMDRAW 1,0,100,100 would draw to the hi-res screen in RAM under the $E000 and $F000 ROMs. [HRAM should be used in this manner with [PLOT, [FLIP, [CLPX, [MCPL, [DRAW, [CHAR, and [CHRX in bank 3. [MXGR should be avoided in bank 3. Using the first 3K of bank 3 will crash SuperBASIC, so make sure the text screen is relocated by [VS1K 3. When the transition to bank 3 is accomplished.

The 1K block at $0400 can be reclaimed for BASIC program storage. [LOOK and [STUF are PEEK and POKE equivalents that can be used with [HRAM to examine and change RAM. [LOOK is different from PEEK in that a defined variable name is used in the parameter list to store the value read from memory. [STUF works the same as POKE and is primarily useful for storing to block $D000 RAM (for example, [HRAMSTUF 53248,255).

Programs 2-6 are demonstration programs which should be helpful in seeing the commands used in actual applications.

If you're not up to typing in SuperBASIC yourself, send $3 along with a blank disk (no tapes) and a stamped, self-addressed mailer to:

Martin C. Kees,
711 YJest Henry,
Pasco, WA 99301.

SuperBASIC Command Reference

Enhanced BASIC Commands

```
RESTORE <exp>                       IF <exp> GOSUB <exp>
GOTO <exp>                          ON <exp> GOTO <expl>,<exp2>,...
GOSUB <exp>                         ON <exp> GOSUB <expl>,<exp2>,...
IF <exp> GOTO <exp>                 LIST (Shift Key halts list)
```

New SuperBASIC Commands

Sprite Commands

```
[DSPR spr,blk,xexp,yexp,xpos,ypos,multi,sprcolr,mc0,mc1
[MOVE spr,xpos,ypos
[KSPR spr
[ESPR spr
[BSPP spr,sel
```

Sound Commands

```
[SSND voice,ad,sr,wave,freq,pwidth
[PLAY 256*wave + voice,freq,pwidth
```

VIC Color Control

```
[BKGD col                           [EXTC col
[BKG4 col0,col1,col2,coI3           [FCOL col
```

VIC Memory Mapping

```
[BANK sel                           [CB2K sel
[VS1K sel
```

Graphics Control

```
[ECGR sel                           [BMGR sel
[MCGR sel                           [KMXG
[MXGR reg,mask,rastl,vall,rast2,val2
[CMXV vall,val2                     [MCPL x,y,sel
[SIZE colsel,rowsel                 [DRAW xl,yl,x2,y2
[XYSC xpos,ypos                     [HRCS address
[DLCS set, address                  [CHAR char,x,y
[PBMS byte                          [CHRX char,x,y
[FSCR byte                          [CODE str$
[PLOT x,y                           [LOOK address, variable
[FLIP x,y                           [STUF address,byte
[CLPX x,y
[HRAM <SuperBASIC mnemonic> <parameterlist>
```

Program 2: Moire Pattern

```
1 REM MOIRE TITLE PAGE DEMO
5 [EXTC0
10 [CB2K4;[BMGR1:[FBMS0:[FSCR1
15 FORJ=0 T0318 STEP2
20 [DRAWJ,198,160,I00 :NEXT
22 FORJ=0 TQ318 STEP2
23 [DRAWJ, 0,160, 100 :NEXT
24 FORJ=0 T0198 STEP2
25 [DRAW160,100,318,J{3 SPACES}:NEXT
26 PORJ=0 T0198 STEP2
27 [DRAW161,100,0,J{3 SPACES}:NEXT
29 [EXTC4
30 M$="SUPERBASIC":[HRCS53248:M?=M$+""
40 X=120:Y=80:GOSUB50
45 M?="{RVS}BY MCSOFT":M?=M$+"";X=124:Y=120:GOSUB50
47 [CHRX54,152,89:[CHRX52,160,89
48 FORJ=1TO800:NEXT
49 [FSCR16:{5 SPACES }GOTO100
50 [CODEM$;FORJ=1TOLEN{M$)
60 [CHRX ASC(MID$(M$,J,1)),x,y
70 X=X+8:NEXT
80 RETURN
100 GETA$:IFA$=""THEN100
110 [BMGR0:[CB2K2
```

Program 3: Geometric Pattern

```
1 REM STAR DEMO
10 PI=3.1415925
20 INPUT "{CLR}POINTS WANTED (0 TO END)";P
W
21 IFPW=0THENEND
22 INPUT "SKIP"; SK
23 INPUT "RADIUS <100 "; R
30 P=PI/PW
50 [BMGRl: [CB2K4:[FBMS0:[FSCR1
60 X=160:Y=100-R:TL=0
70 FORJ=lTOPW
80 TH=TL+SK
90 TL=TH;TH=TH*P-Cpi/4)
100 X2=COS(TH)*R+160
110 Y2=SIN{TH)*R+100
120 [DRAWX,Y,X2,Y2
130 X=INT(X2) :Y=INT{Y2) :NEXT
140 GETA$:IFA$=""THEN140
150 [BMGR0:[CB2K2:PRINT"{CLR} " :GOTO20
```

Program 4: Joystick-Controlled Sprites

```
1 REM DOODLE
5 GOSUB900:[DSPR1, 13, 0,0, 160+16, 100+44,0,0;GOSUB140
10 [BANK0:[CB2K4:[BMGRl :[FBMS0: [FSCR1: [BSPP 1,1
20 E=l:X=160:Y=100:C=-1 :FORQ=1TO100:NEXT
30 IFPEEK(203)=60THEN130
31 IFPEEK(203)=4THENE=-E:IFE>0THEN [DSPR1, 13,0,0,0,0,0,0
32 IFE<0THEN[DSPR1,13,0,0,X+16,Y+44,0,12: [CLPXX,Y
35 JV=PEEK( 56320 ) :FR=JVAND16
40 JV=15-(JVAND15)
50 IFJV=0ANDFR=16THEN30
60 IFJV=10RJV=50RJV=9THENY=Y-1 : IFY<0THENY=199
70 IFJV=2ORJV=6ORJV=10THENY=Y+1:IFY>199THENY=0
80 IFJV>=4ANDJV<=6THENX=X-1 : IFX<0THENX=319
90 IFJV>=aANDJV<=10THENX=X+l : IFX> 319THENX=0
100 IFFR=0ANDJV=0THENC=-C : E=l : FORQ=1TO100 : NEXT
:IFC>0THENCKSPR1:POKE53 288,0
105 IFE<0THEN[ESPR1 : [MOVEl , X+16 , Y+44 : [CLPXX,Y:GOTO30
110 IFC>0THEN[PLOTX, Y:GOTO30
120 IFC<0THEN[ESPR1 : [MOVEl , X+16 , Y+44 : GOTO 30
130 [BANK0: [BMGR0: [CB2K2 : POKE198 , 0: PRINT"{CLR}":[KSPR1:END
140 PRINT" {CLR} DOODLE 64"
150 PRINT" {down} USE JOYSTICK IN PORT 2"
160 PRINT"BUTT0N TURNS INK ON/OFF"
165 PRINT"F1 TURNS ERASE MODE ON/OFF"
170 PRINT "HIT A KEY TO START"
180 PRINT"HIT {RVS} SPACE {off} TO STOP"
185 PRINT"THE BLACK + IS YOUR CURSOR WHEN INK=OFF"
186 PRINT"THE GREY + IS YOUR CURSOR WHEN { SPACE } ERASE=ON " : [BKGDl :
[ PCOL0
190 GETA$:IFA$=""THEN190
200 IFA$=" "THENRETURN
210 RETURN
900 X=13*64
910 READ Y: IFY<0THENRETURN
920 POKEX,Y:X=X+1:GOTO910
1000 DATAl, 192, 0,1, 192, 0,1, 192, 0,1, 192,0, 1,192,0
1010 DATA0, 128, 0,126, 63, 0,0, 128, 0,1, 192,0 ,1,192,0
1020 DATAl, 192, 0,1, 192, 0,1, 192, 0,0, 0,0,0, 0,0
1030 DATA0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0, 0,0
1040 DATA0, 0,0,-1
```

Program 5: Sprite Animation

```
1 REM FALLING SHAMROCKS
2 REM HIT A KEY TO STOP PROGRAM
5 [EXTC 13: [CB2K 4: [BMGR l: [FSCR 5: [FBMS 171
10 X=832:V=53265:R=128
20 READ A: IFA<0THEN35
30 POKE X, A: X=X+1:GOTO20
35 FORJ=0TO7
40 [DSPRJ,13,1,1,0,0,0,5+J{2 SPACES}:NEXT
50 FORJ=lT0256: F0RK=1T08: [MOVEK-1, J+K*K, J*K+K: NEXT: WAITV, R: [FSCRJ/2
55 GETA$:IFA$<>""THEN300
56 NEXT
60 X=PEEK(8192)+1:[FBMS X: GOTO50
100 DATA0, 102, 0,0, 255, 0,1, 255, 128, 3, 255,192
110 DATA3, 255, 192, 2 5, 255, 152, 60, 126, 60, 126,126,126
120 DATA255, 60, 255, 255, 255, 255, 127, 255, 254,255,255,255,255
130 DATA24, 255, 126, 24, 126, 60, 24, 60, 24, 24,
24,0,24,0,0,24,0,0,0,0,0,0,0,0,0,0,-1
300 [CB2K 2: [BMGR 0:FORJ=0TO7: [KSPR J: NEXT
```

Program 6: Simple pet Emulator

```
10 REM ROUTINE TO SET BASIC MEMORY AND SCREEN TO PET STANDARD LOCATIONS
20 REM SCREEN AT 32768
30 REM BASIC 1024 TO 32767
40 REM ASSUME IN C-64 STANDARD MAP
50 [FSCR 0: [VS1K 0: [BANK 2: PRINT" {CLR} "
60 POKE44,4;POKE 45, 3: POKE46, 4
70 POKE55,0:POKE56, 128
99 NEW
```