SUPER SCREEN

1. A big 52 character by 24 line screen.
2. **PRINT @** is fully implemented on the big screen.
3. Easily combine text with Hi-res graphics.
4. Auto-key repeat for greater keyboard convenience.
5. The **ON ERROR GOTO** statement is fully implemented.
6. Control codes for additional functions.

SUPER SCREEN is a powerful, machine language program that significantly upgrades the performance and usefulness of 16K or greater, Extended and Disc Basic Color Computers. Replace your old 32 character by 16 line display with a brand new, 51 character wide by 24 line display screen including a full upper and lower case character set. In addition, **SUPER SCREEN** fully supports the important **CLS** and **PRINT @** Basic commands that make it easy to format business and personal programs on the new big screen with professional results. **SUPER SCREEN** is so versatile, you can even combine text characters with graphics created using the Basic **DRAW, CIRCLE** and **LINE** commands.

SUPER SCREEN corrects a serious deficiency of Color Computer Basic with a full implementation of the **ON ERROR GOTO** statement, including the **ERL** and **ERR** functions. Now you can trap errors and take corrective action to prevent crashed programs and lost data using the same syntax as many other computers. This addition to the Basic instruction set greatly enhances the capabilities of the Color Computer to handle serious tasks.

Another important feature you will appreciate is the Key Press Auto-Repeat. No more frustration as you edit a long line in your Basic program; just hold the space bar down and automatically step to the desired position in the line. Special control codes are also included so that you may select a block or underline, solid or blinking cursor as well as reverse screen display and other useful functions. **SUPER SCREEN** will be the most valuable and frequently used utility program in your library because of the increased power, versatility and operating convenience it brings to your Basic programming.

GETTING STARTED

The cassette version of **SUPER SCREEN** is loaded the same as other machine code programs: type **CLOADM** and press**<ENTER>**. In a similar manner, load the disc version by typing **LOADM"SUPRSCRN"** followed by **<ENTER>**. Once loaded, type **EXEC** and press **<ENTER>** to commence operation. **SUPER SCREEN** will automatically adjust itself to 16K, 32K or 64K computers and then respond with the familiar **OK** Basic prompt on a new wide screen.

Type in and run the following short program:

```
10 CLS
20 FOR N=33 TO 126:PRINT " ";CHR$(N);:NEXT
30 PRINT " ";
40 FOR N=1 TO 2
50 FOR X=1 TO 1000:NEXT
60 PRINT CHR$(6);
70 NEXT N
80 PRINT
```

SUPER SCREEN will display your new **ASCII** character set with true upper and lower case. Note how the program momentarily reverses the display screen using the control code character in line 60. If you are using a color monitor, you may want to turn the color control down to get a true black and white text display. Observe that the **CLS** command in line 10 is not suffixed with a number to set screen color and an error will result if you attempt to do so.

CONTROL CHARACTERS

SUPER SCREEN recognizes several control codes which tell it to perform special functions. These control code characters are sent to **SUPER SCREEN** by means of the Basic **PRINT CHR$(N)** statement where **N** is the decimal equivalent of the control code character. In line 60 of the preceeding program, **SUPER SCREEN** recognized the decimal code '6' and reversed the display screen. The following list summarizes all of the special control codes:

6	Reverses the display screen character and background color.
8	Backspace code. Backs the cursor one character position.
10	Line feed character. Moves the cursor down one line.
13	Carriage return. Returns the cursor to the beginning of the line.
14	Switches between block or underline cursor type.
15	Switches between blinking or non-blinking cursor.
28	Home up. Positions the cursor at the upper left of the screen.
30	Erases from the cursor to the end of the line.
31	Erases from the cursor to the end of the screen.

Type in the following: **PRINT CHR$(28);CHR$(30)** **⟨ENTER⟩**

This sequence of two control codes erases the entire top line. The special code 28 positions the cursor at the upper left of the screen (Homes the cursor), and the code 30 erases to the end of the line.

Next type: **PRINT CHR$(28);CHR$(31)** **⟨ENTER⟩**

Note that this combination of control codes performs the same function as the Basic **CLS** command. The code 28 homes the cursor and the code 31 erases to the end of the screen.

Now type: **PRINT CHR$(15)** **⟨ENTER⟩**

The cursor will stop blinking until you repeat the instruction.

Type: **PRINT CHR$(14)** **⟨ENTER⟩**

The block cursor is now an underline and will remain so until you repeat this instruction.

AUTO KEY PRESS REPEAT

Press any alpha or numeric key such as the **'L'** and hold it down for a moment. The Auto Repeat will begin after the key is pressed for approximately one half second and will continue until the key is released or the input buffer has been filled. After printing a few L's on the screen, press the left arrow and hold it. We think you will agree that this is one of the handiest improvements available for your Color Computer.

PRINT @

One of the features that makes **SUPER SCREEN** truly unique is it's support of the Basic **PRINT@** statement. If you do not fully understand the use of **PRINT@**, please refer to your Basic and Extended Basic instruction manuals including the **PRINT@** worksheet towards the rear of the Extended Basic manual. **PRINT@** with **SUPER SCREEN** works in the same manner as before, but you no longer get an error if you attempt a **PRINT@** beyond location 511. Refer to the new **SUPER SCREEN** worksheet and observe that the screen locations are numbered from 0 through 1223.

Consider the powerful screen formatting tools at your disposal when the **PRINT@** statement is used in conjunction with special control code characters. It is now a simple matter to print text at any location on the screen and then perform various special functions as well!

Type the following: **PRINT@510, "THIS IS SAMPLE TEXT"** **⟨ENTER⟩**

Next type: **PRINT@510, "SO IS THIS"** **⟨ENTER⟩**

The second **PRINT@** statement printed right over the first as you would expect.

Now type: **PRINT@510, "THIS TOO";CHR$(30)** **⟨ENTER⟩**

Once again, the text printed in the same location but the special control code 30 then proceeded to erase the remaining old text to the end of the line. Try the same thing again substituting a 31 for the special code 30.

Now try: **PRINT@0,CHR$(31)** **⟨ENTER⟩**

This is another way to duplicate the Basic **CLS** statement. The **PRINT@** portion of the statement homed the cursor and the control code 31 erased the screen.

Type in and run the following:

```
10 CLS
15 PMODE 4,1
20 CIRCLE (128,96), 70, 0, .9
25 LINE (128,96)-(198,96), PRESET
30 LINE (128,34)-(128,157), PRESET
35 PRINT@112, "Diameter and radius of a circle"
40 X=432:A$="Diameter"
45 FOR N=1 TO 8: PRINT@X,MID$(A$,N,1):X=X+51:NEXT
50 PRINT@590,"Radius"
55 PRINT@1020
```

Although the preceeding program is very simple it demonstrates a number of important points. The **PMODE** statement in line 15 sets up for the graphics to mix text and graphics using both upper and lower case letters. Switching from upper to lower case and back is done as before, by pressing the **SHIFT** and **0** keys. Finally, note how the **PRINT@** statement in line 55 positions the cursor near the bottom of the screen.

ERROR TRAPPING AND REPORTING

You have probably experienced the frustration of having an error halt the execution of a Basic program leaving you with no easy way to resume operation. In fact, when Basic encounters an error while executing a program, it immediatley closes all open files and adjusts several registers which can result in a serious program crash, lost data and a very angry operator. The **ON ERROR GOTO** statement offers an alternate method of handling errors and 'user-friendly' programs use the technique extensively.

Type in and run the following:

```
10 CLS
20 PRINT "THIS IS AN EXAMPLE OF ERROR TRAPPING"
30 ON ERROR GOTO 1000
40 PRINT
50 PRUNT: REM NOTE THE ERROR IN SPELLING 'PRINT'
60 STOP
1000 PRINT "WE TRAPPED AN ERROR";ERR;"IN LINE";ERL
1010 STOP
```

When you tell Basic to **RUN** a program it will report all errors in the usual manner unless it executes an **ON ERROR GOTO** statement before the error is encountered. The **ON ERROR GOTO** statement in line 30 tells Basic to immediately branch to line 1000 for further instructions if any error is detected, and our little program conveniently provides an error in line 50. Note that there is nothing special with the line number 1000 and you may use any legal line number that suits your purposes.

Line 1000 makes use of two powerful new functions; **ERR** and **ERL**. The **ERR** function returns a number equivalent to the error encountered (see the conversion table) and the **ERL** function gives us the line number in which the error occurred.

Type: **PRINT ERR,ERL** **<ENTER>**

It is possible to reference these functions in the direct mode as you have just proven, but they may be tested also so that you can take action based upon their values.

Add the following line to the program and run it again:

1005 IF ERR=2 AND ERL=50 THEN GOTO 60

Line 1005 tests the numeric error code, the line number in which it occurred, and finding them both true causes a branch back to line 60. Keep in mind that all of these activities take place before Basic has closed any files or changed any registers that could disrupt your program or data.

Now change the program as follows and run it again:

1005 PRUNT:REM NOTE THE SECOND SPELLING ERROR

Observe that Basic detected the first error in line 50 which resulted in a branch to line 1000 but it reported the second error in the usual manner. This is because the first error cancelled the error trap preventing the possibility of an endless loop in your error trapping procedure. Once an error has been trapped, your program must provide another **ON ERROR GOTO** statement to trap subsequent errors. It is important to note that Basic responds to the latest error trap it encounters during execution of a program. As your program proceeds from one task or routine to the next, you are free to specify a new trap line which can most effectively deal with the type of errors you will most likely encounter. A typical example would be to precede any disc or tape I/O routines with an **ON ERROR GOTO** statement that would permit valuable data to be saved if any errors are encountered.

Now type: **ON ERROR GOTO 1000** **⟨ENTER)**

It is illegal to use an error trap statement in the direct mode. All such statements must be a part of a Basic program.

Type: PRINT ERR, ERL **⟨ENTER⟩**

Note that the **ERR** and **ERL** functions are holding the values associated with the unsuccessful direct statement you just typed and they will continue to do so until the next error is encountered. Note also that the line number defined by the **ERL** function is zero because the error was committed in the direct mode.

ERROR CONVERSION TABLE

ERROR NUMBER	EQUIVALENT ERROR MESSAGE	ERROR NUMBER	EQUIVALENT ERROR MESSAGE
0	NF	38	DN
2	SN	40	IO
4	RG	42	FM
6	OD	44	NO
8	FC	46	IE
10	OV	48	DS
12	OM	50	--
14	UL	52	NE
16	BS	54	BR
18	DD	56	DF
20	/0	58	OB
22	ID	60	WP
24	TM	62	FN
26	OS	64	FS
28	LS	66	AE
30	ST	68	FO
32	CN	70	SE
34	FD	72	VF
36	AO	74	ER

NOTE: Many of the error codes are found only in Disc Basic.

SOME TECHNICAL INFORMATION

After **SUPER SCREEN** is loaded and **EXEC**uted, it first determines how much memory is available. If your computer has greater than 16K, **SUPER SCREEN** moves itself from the high end of the first 16K to the high end of the second 16K section of ram, after which a number of registers are modified to inform Basic of **SUPER SCREEN**'s presence.

SUPER SCREEN takes advantage of the fact that Basic automatically reserves four 1.5K pages of memory for screen graphics. For this reason you should never use a **PCLEAR** statement with a suffix less than 4 when running **SUPER SCREEN**.

Disc Basic provides the **FILES** statement to tell your computer when and how much additional memory to reserve for disc file buffers. Use of the **FILES** statement moves the memory reserved for screen graphics however, so any such use should be before **SUPER SCREEN** is **EXEC**uted. Let's assume the computer has just been turned on and you want to run an accounting program requiring three disc buffers with a total of 1000 bytes.

Type: **FILES 3,1000** **⟨ENTER⟩**

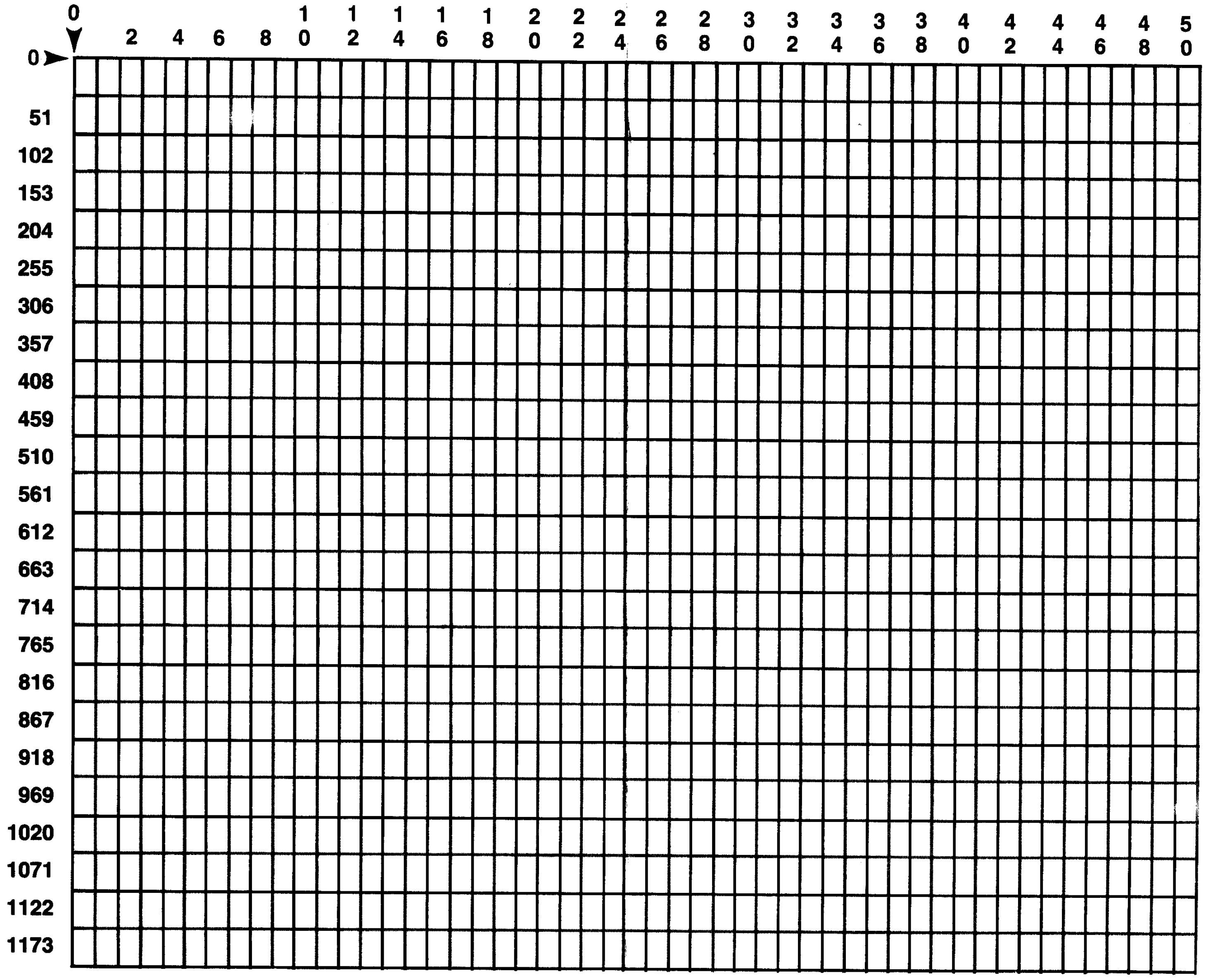
This instruction reserves 1,000 bytes for 3 disc files and moves the screen memory up by that amount to provide the room. You may now **EXEC**ute the **SUPER SCREEN** program and it will determine the proper screen location. Avoid further use of the **FILES** statement to prevent another relocation of the screen.

Some of the Disc Basic commands will momentarily change the appearance of your display screen. The **BACKUP** and **DSKINI** commands for example, switch the screen back to the old 32 by 16 format and use some of **SUPER SCREEN**'s display memory as they perform their tasks. As these disc operations are completed, **SUPER SCREEN** takes over again but you may see some garbage on the screen which you can ignore or remove with the **CLS** command. These effects are normal and do not interfere with the operation or usefulness of **SUPER SCREEN**.

The prime purpose of **SUPER SCREEN** is to give your Color Computer the power to run more professional business and personal Basic programs. Because of the nature and complexity of this software we cannot predict or guarantee performance when it is used with patches or modifications to the Basic roms originally supplied.

SUPER SCREEN

PRINT @ WORKSHEET



SUPER SCREEN ADDENDUM

Please add the following to the list of CONTROL CODES in your instructions.

5 - Changes screen color from buff to green or green to buff. Some users will prefer to use the green background as it prevents artifact color interference with text letters.

16 - Switches the cursor off and on.