COPYRIGHT AND TERMS AND CONDITIONS

DANOSOFT, P.O. Box 124, Station "A", Mississauga, Ont. Canada.

Third Printing

INTRODUCTION

Welcome to "BIG BASIC", a powerful basic system modification.

Its memory management provides up to 472K of user programming and/or data storage in 512K CoCos or up to 92K in 128K CoCos.

Unlimited sized programs or data can be chained from disk without erasing variables or re-initialization.

A slightly modified "CLEAR" command and three new basic words activate the magic of "BIG BASIC".

"WINDOW" switches you between two distinct user programming areas. "Window 1" can act as a control or menu base, switching programs, variables or sections of a single large program in "Window 2".

"BLOCK" switches between 59 blocks of 8K user memory in a 512K CoCo or 11 blocks of 8K user memory in a 128K machine.

"VSWITCH" temporarily allows a program's variables in one window to be used by a program in the other. Repeating the command restores the original variables intact.

In Window 2 the "CLEAR" command works normally. But in Window 1 it can also be used to set the partition between the two windows. Window 2 can be one to three blocks of 8K in size, (8192 to 24576 bytes) or you can leave the computer in the normal single window mode.

This manual provides details of your power options as outlined above and includes a memory table of the available 8K Blocks you can use.

Seven demonstration programs are included on your "BIG BASIC" disk. They show just some of the possibilities with BIG BASIC. After you get started, be sure to list and study them for examples of programmming with BIG BASIC.

We trust "BIG BASIC" will open new opportunities for your use of the computer and wish you enjoyment with our favorite machine.

GETTING STARTED

The best way to get an idea of the expanded power of "Big Basic" is to see a simple demonstration. Let's start by installing BIG BASIC from a "Coldstarted computer. (See "COLDSTART" at the end of this manual.)

Enter > LOADM "BB/512"

(128K machines use the "BB/128K" program. For DOS 2.0 use the "BB 512/1-0" or "BB 128/1-0" programs. For ADOS, LOAD and LIST either "ADOS-BB/512" or "ADOS-BB/128" and read instructions on how to use BIG BASIC with ADOS.)

Press a key and then enter> ?MEM

You will see that BIG BASIC starts with more than 28K, not the usual 22+K. (See "PCLEAR" in Appendix "A" for more detail.) This is your first BIG BASIC bonus.

On your "BIG BASIC" disk is the "BB DEMO" program and its "LOADER". We will run this now.

Enter> RUN "LOADER".

Only eight demo test programs were loaded in, but from this you can get the idea. This program can be expanded on a 512K CoCo to run 58 basic programs of less than 8K each simultaneously. (It also is possible to run one huge 400K+ program with any mix of program or data.)

Select a program number and press ENTER. You will find yourself in the centre of a short test program. Press any key and return to the Menu program. Now select "WINDOW/DEMO"; read the message; press BREAK; and list that program for more information. Enter RUN and press any key to return to the main menu program.

Listing "WINDOW DEMO" or "PROGRAM 3" to "PROGRAM 8" reveals the code that makes them possible. Each one is in a window space of its own. You can edit the programs, save them, load them, do a "NEW", whatever. The only thing they have in common is they share the same screens. Therefore screens are cleared when switching

windows and redrawn by the program in the next window on entry; unless you use the technique described on page 8 of this manual.

The menu program is in the base window, or Window 1. All the other programs are in Window 2, but Window 2 is using different memory for each. Similar line numbers in each window do not effect each other. Variable "A" in Window 1 is not the same variable "A" as in Window 2. (But either window can access the variables of the other, if desired, with the "VSWITCH" command.)

Instead of multiple self-contained programs in Window 2, you may use that window for multiple sections of one large program or for storage of data. Window 1 must have a control program that selects which block of your computer's memory you want to place in Window 2. Then the Window 1 control program uses the WINDOW command to jump you into your program section in Window 2. Normally, when your program sends you from Window 2 back to Window 1, you will have code there to further direct with the BLOCK command other program sections into Window 2.

If Window 2 is saved from Window 1 with the "SAVEM" command, when it is subsequently reloaded with the "LOADM" command from Window 1, you can re-enter a running program in Window 2 and you will find all variables in Window 2 will be intact from the time of the "SAVEM".

TO USE BIG BASIC WITHOUT DIFFICULTY YOU MUST CAREFULLY STUDY THE COMMANDS AND INFORMATION CONTAINED IN APPENDIX "A" AND APPENDIX "B".

EQUIPMENT REQUIREMENTS

"BIG BASIC" runs in a CoCo3 with Disk Extended Basic RS-DOS. It's a machine language program that patches the computer's basic operating system in some 70 locations. Its presence is invisible to the user and it does not occupy user memory.

In order to find memory in the operating system for some of BIG BASIC'S features, nearly all memory relating to CASSETTE operations has been replaced. Therefore you cannot use cassette commands with BIG BASIC. (Use of these commands will produce a "SN" error.)

It is suggested that running of multiple programs, or huge programs, would not likely be undertaken in a practical way from a cassette storage medium.

On the technical side users should be aware this program does not use any basic system variable memory in the low end of the computer except the reset vector at $72 to a "Reset Protect" routine which other M.L. programs can overwrite if they wish. Memory at the end of the Disk Extended Basic section and CoCo3's expanded commands section also is unused. These are popular locations with other machine language programs and were avoided to try to develop compatability.

The use of HARD DISKS is what we have in mind. A few, but not all, hard disk systems have been tested. We know RGB DOS by RGB Computer Systems of 294 Stillwell Ave., Kenmore, N.Y. 14217 works well with BIG BASIC. This company's hard disk system is among the best for the Color Computer and is especially compatible with most basic programs.

So far, BIG BASIC will not run under Burke & Burke's "Hyperio" system.

We welcome any information users of this program can provide regarding using BIG BASIC with various hard disk interfacing software packages.

Most existing basic programs will run in BIG BASIC windows without change, but conflicts, even crashes, can definitely arise if BIG BASIC protocols are not followed. Study of Appendix "A" will show you what commands may require editing to make an existing program conform.

THE CONCEPT OF BIG BASIC

The CoCo3, unlike its predecessor models, contains a special Memory Management Unit (MMU). This powerful addition manages all memory in the machine in blocks of 8K. Machine language programmers have been able to use it for large programs and it is relied upon by the OS-9 system.

The MMU is required because the computing chip in the CoCo can only work with 64K at one time.

In order to bring more memory to Basic users, BIG BASIC has developed the concept of using two windows. Programming in the base window (Window1) can direct the computer to flip in all its extra memory to Window 2. BIG BASIC's simple new command word, BLOCK, does the job of communicating between you, the basic user, and the MMU.

Blocks are brought into Window 2 one at a time, or up to three at a time, depending on the size you set for Window 2 with the CLEAR command. If you work with just one block at a time, executing programs and searching for variables will require basic to search through just a maximum of 8K. Thus under BIG BASIC you can have a fast running program even though it might be more than 400,000 bytes in length!

We believe this concept of operation is the best way for basic programmmers to create massive programs. Machine language programmers may even wish to save time by letting BIG BASIC do the memory management for them so they can do the programming that requires speed or special operations.

LOADING AND SAVING

Under BIG BASIC you may load and save from either window in the normal manner. By saving in blocks of 8 to 24K at a time, you may have programs prompt the changing of several disks in order to store massive programs or data.

Because LOAD and RUN erase existing memory and variables when transferring from disk to computer, BIG BASIC has developed the concept of SAVEM and LOADM. If from Window 1 you SAVEM the contents of Window 2, you will save the parameters of the running program and the variables in that window. When you LOADM again and switch to Window 2, you will find everything as you left it.

LOADM and SAVEM will not work this way in standard CoCo3 basic because the Basic operating system stores its "Stack" between your variable tables and your string variables and, when you LOADM back your program, stack conflict causes a crash (sometimes so subtly you may not notice at first.) BIG BASIC has moved the "stack" and taken other necessary steps to make LOADM and SAVEM work in this unique way.

Using LOADM and SAVEM you can transfer to disk unlimited amounts of program sections or data!

The normal syntax for the SAVEM command is required. That is, you need start, end, and execution addresses. You will find the starting and ending addresses of each window size in the table in Appendix "A" under the CLEAR command. Use 0 for the execution address. i.e. To SAVEM a one-block Window 2 use: SAVEM "TEST",24576,32767,0

You may find it helpful to set variables in your program for these addresses. Then you can SAVEM "TEST",A,B,0

CAUTION!

Since LOADM is a straight memory load, it is possible to load a 2-block program to a 1-block window; or a 3-block program to a 2-block window, or

some similar misadventure. To prevent a crash, or memory loss, you must be careful not make this error. We suggest a code denoting window size be included in the program name when it is being saved. Or all memory being saved from Window 2 should bear a related name to the base program in Window 1 and all LOADM/SAVEM operations be executed from program lines in the base program. If the same program that saved is also the program that loads there will be no window size problems.

BIG DISK AND DOUBLE40

Danosoft markets two utilities, BIG DISK and DOUBLE40, which provide basic users who have double-sided drives (like Tandy's drive FD-502 for the CoCo) with the ability to store 360K on one floppy disk. Conflicts with the autostarting of these utilities and BIG BASIC's autostart require that the Disk Utilities be loaded into the computer FIRST.

MAINTAINING THE SCREEN

BIG BASIC windows all use separate programming memory but share the same screens. (As per references on Pages 4 and 13.) These screens are cleared every time you jump between windows.

It is possible to go to the other window without clearing the screen provided the same type of screen (i.e. 32 or 40/80 or graphics) is being used by both windows. POKE 33022,57 (after installing BIG BASIC) will disable the clearing action caused by switching. POKE 33022,150 will restore to normal.

USING TWO HI-RES TEXT SCREENS AT ONCE
THE "TWOVIEWS" DEMO

A Demonstration Program named "TWOVIEWS" is included on your BIG BASIC program disk. You can hold and switch between two hi-res text screens at once. (For instance: one screen for a main menu, another for a disk menu.) One screen can even be in 40 columns while the other is in 80.

APPENDIX B of this manual shows you that the Hi-Res Text screen uses the 8k memory block No. 54. However, both the 40-column and the 80-column screens are contained in this block in less than 4K of memory. Therefore we have developed the pokes that modify the operating system to let you have another HI-Res Text screen in the second unused 4K.

These two video screens are distinct from the programming windows. Both can be used in any one of the BIG BASIC programming windows. In fact, BIG BASIC is not required at all. "TWOVIEWS" is a stand-alone system that will work at any time in any CoCo3. When combined with BIG BASIC, basic programmers now have even greater flexiblity and power.

When you list "TWOVIEWS" (in the 40-column screen) it will seem long because it is filled with remarks and notes about every part of its function. It is best to print it out if you have a printer.

The program actually consists of two subroutines....one to restore the video to normal, the other to set the video to the second 4K memory area. If you select what you need from a backup of the program, and delete all spaces and remarks, and combine the various pokes together on a single line, each subroutine will be only two or three lines long. Packed this way, switching between video screens will be instant.

AUTOEXEC

Some users want their program to autoexec after installing BIG BASIC.

If you install BIG BASIC with: POKE 44539,57: LOADM "BB/512" (or "BB/128") the title screen will come up as usual but BIG BASIC then will try to run a basic file on your disk named "*/BAS". If it doesn't find it, you will return to direct mode with a "NE" error.

You must create the "*" file on your working disk to autoexec. (This is distinct from the "*/BAS" file now on your master disk which is used when installing BIG BASIC with "ADOS3".) It might look like:

```
10 RUN "TEST/BAS"    or    10 LOADM "TEST/BIN"
```

RGB DOS, which can autostart from hard disk, autoexecs in a similar way to "*". Under this DOS you would have a basic program named "AUTOEXEC/BAS" which would have a one line program: 10 POKE 44539,57: LOADM "BB/512". Then, the one line program "*" would take over and run your program under BIG BASIC.

When autoexecuting, you might want to change the last line of the sign-on message to something like "LOADING MY PRG", instead of the "READY!...PRESS BAR". While you cannot change the entire sign-on message and copyright without causing a crash, you can change 18 bytes for your own message on an exact copy of the original BIG BASIC disk according to the following table:

Version	Track	Sector	Bytes
BB/512	16	2	116 - 133
BB/128	16	11	116 - 133
BB 512/1-0	27	2	116 - 133
BB 128/1-0	27	11	116 - 133

For Example to change "BB/128":

```
DSKI$0,16,11,X$,Y$: MID$(X$,116)=" LOADING
 YOUR": MID$(Y$,1)=" PGM ": DSKO$0,16,11,X$,Y$
```

NOTE: There are 13 bytes inside the quotations for X$ and 5 bytes for Y$.

APPENDIX "A"

"BIG BASIC" has 3 new commands. Also, minor changes have been made to a few other basic commands to permit them to operate in the improved "BIG BASIC" environment.

CLEAR:

- Functions normally in Window 2.
- Functions normally in Window 1 if just string space is being set. (i.e. CLEAR 200).
- Otherwise in Window 1, "CLEAR" sets the window partition to one of three sizes to either create Window 2; or no 2nd Window at all. A single Window is the "BIG BASIC" default when first loaded.
- Window partition addresses are:

NO./BLOCKS	START	END
0	32720	32767
1	24576	32767
2	16384	32767
3	8192	32767

- "CLEAR 200,32720" or higher sets the single window status.
- "CLEAR 200,24576" sets a 1-block Window 2.
- "CLEAR 200,16384" sets a 2-block Window 2.
- "CLEAR 200,8192" sets a 3-block Window 2 (Note: A PCLEAR0 or 1 setting must be in place to have a 3-block Window. Otherwise no memory would be available for the base Window 1.)
- Please note that any address number less than 16384 will set a 3-block window; any address between 16384 to 24575 will set a 2-block window; any address number from 24576 to 32719 will set a 1-block Window; and any number higher than 32719 will cancel the partition and maintain a single window status. This makes remembering partition addresses easier. i.e. "CLEAR 200,30000" will set a 1-Block Window. "CLEAR 200,20000 will set

a 2-Block Window. "CLEAR 200,10000" will set a 3-Block Window.

WARNING:

It is possible to crash the computer and lose all your programming in the "Normal" 64K of computer memory using the "CLEAR" statement to partition a second time. (Coldstarting, without turning the machine off, however, will preserve all programming outside the "Normal" 64K, except the HGET/HPUT buffer in Block 52.)

The way to cause the crash is to partition after a program, or variables, is installed in Window 2. This is because new window sizes will cause the two programs residing in the windows to overlap along with their parameters which BIG BASIC needs to maintain the windows. You can do a normal "CLEAR" string space in Window 1 with no problem. (i.e. "CLEAR 1000") Or you can set protected memory in Window 2 (i.e. "CLEAR200,31000"); but to change the partition, you must do a "NEW" in Window 2 and erase the memory there to initial condition.

WINDOW#

- Switches the user between 2 programming windows.
- The option must be 1 or 2 and must come after the 2nd "W" without a space. Otherwise you will get a "SN" error.
- You may use "Window 1" even if you are already there and nothing will occur. This is useful if you are not sure which window you are in. If you want to be in Window 1 simply enter "WINDOW1" and you jump there or you are already there.
- If no Window 2 has been set up by the "CLEAR" command, "WINDOW2" will produce a "FC" error.
- If "VSWITCH" has switched the variables between windows, "WINDOW"(option) will produce a "FC" error. This is to make sure windows and their variables do not become mixed up. A "VSWITCH" will correct the error.
- If you transfer out of a window in direct mode, you will be in direct mode when you later return.

- If you transfer out of a window from within a program, your later return to that window will be to the next command of the program after the "WINDOW" command. Since both windows share the same screens, your screen will be cleared on the return. If you wish to maintain a screen display or menu, your next command after "WINDOW" should re-establish the screen, but not the width. Screen widths (i.e. 32, 40 or 80 columns) are preserved during window changing. (See the demo program that came with the BIG BASIC program.) See page 8 for another method of keeping your screen.

BLOCK##

- This command points Window 2 to memory elsewhere in the computer.
- Study Appendix "B" to select appropriate 8K memory blocks to use.
- The option must be a Decimal number between 0 and 59, and not be 56, or less than 48 in a 128K CoCo. Other numbers will produce either "SN" or "FC" errors.
- The option cannot be a variable and there must be no space between "K" and the number to avoid an error.
- Blocks that are in use in one size of a window cannot be switched into the wrong size. i.e. 2-Blocks cannot be switched into a 1-Block Window.
- The Block command checks the size of Window 2 and switches in the block specified by the option plus the next two highest block numbers if the window size will accomodate them. i.e. Block 50 will switch 50, 51 & 52 into a 3-Block window.
- Any attempt to switch in an inappropriate set of blocks will produce a "FC" error. i.e. - in a 2-Block Window, "Block 55" will produce a "FC" error because Block 56 cannot be used. (56 is a basic system block). If earlier, Block 48, 49 and 50 were switched out as a 3-Block window, you will not be able to switch in just blocks 49 & 50 in a 2-Block Window.

- To unflag blocks for free use, you must first replace them in their original window size, then erase their memory with "NEW". Coldstarting (without turning off the machine) will not do this job. Even though the machine appears to have coldstarted, memory outside the "Normal" CoCo 64K is not affected.
- Care must be taken in block switching to select usable blocks, especially graphics blocks 48, 49, 50, 51 & 52 and the Hi-Res text screen Block 54. If you are not using the Hi-Res text screens, you may use Block 54 for programming and data; but if you subsequently use the Hi-Res screen, your program will be erased in that block. i.e. You put a program in Block 54. Later you enter "WIDTH40". The screen will jump to the 40-column width, but the screen clearing action of doing so, will erase the program. Similar action applies to the Hi-Res graphic screen, or the HGET/HPUT buffer in Block 52 if you use HGET/HPUT for graphics. Best to use these blocks last when you have no other choice.

VSWITCH

- VSWITCH has many uses, one of which is to let you rapidly access from Window 1 large amounts of data being block switched into Window 2, or even block loaded from disk with "LOADM" (see the SAVE/LOAD section.) Another use is to monitor loop variables or flags. (See "GOSUB/RETURN" & "FOR/NEXT" below.)
- "VSWITCH" switches into the current window all of the variables and arrays contained in the other window; but sets aside the current window variables for restoring later.
- A subsequent "VSWITCH" does the restoration of the original data.
- You may change add or delete the data while it is switched and the changes will still be there when you later go to the other window.
- If you are not sure whether data has been switched, the command "PEEK(65259)" will help. Not switched will equal 0. Switched will equal

1. That memory location contains a "BIG BASIC" status flag. Do not "POKE" or change it, or you will get scrambled memory.
- WARNING: When your data is "VSWITCHED" do not edit, delete, or add basic program lines. These actions will overlap variable memory and will probably cause a crash.
- Just change or use your variables while you are "Vswitched", then "VSWITCH" back to normal.
- Also, avoid using "SAVE" or "LOAD" when "Vswitched" to prevent problems.
- At least one line of program or one variable must exist in Window 2 when you use VSWITCH in Window 1 or you will get a "FC" error.
- Some users may want to permanently transfer variables between windows. The files "VSWITCH/STR" and "VSWITCH/NUM" on the BIG BASIC disk, provide one-line basic subroutines to do this. List them to an 80 column screen or print them to see lengthy comments about the method.

GOSUB/RETURN
FOR/NEXT"

- GOSUB/RETURN and FOR/NEXT loops work normally in their respective windows but are cancelled when you switch windows. However, you can still loop between windows. Loop count numeric variables, and return variable flags, can be accessed from the other window with the "VSWITCH" command. Therefore FOR/NEXT loops and GOSUBS can be simulated with variables even though a window switch has been made.

For Example:

PROGRAM IN WINDOW 1:

```
10 WINDOW1  'Main program in Window 1
20 A=4  'We will access the subroutine in
        Window 2 for four times.
30 WINDOW2 'Switch to Window 2
40 If A>4 THEN END ELSE 30 'Window 2 will
   always return to this line in Window 1.
```

PROGRAM IN WINDOW 2:

```
10 PRINT "TEST"  'Do the subroutine here.
20 VSWITCH   'Get Window 1 variables
30 A = A+1  'Increment loop count
40 VSWITCH 'Restore Window 2 Variables
50 WINDOW 1 'Return from the subroutine
60 GOTO 10 ' Once the program in this window
          has been set running, WINDOW
          switches to Window 2 will always
          come to this line.  This condition
          can be made permanent after original
          programming by saving and loading
          this window with the "SAVEM" and
          "LOADM" commands from WINDOW 1.
          (See "SAVING" & "LOADING")
```

PCLEAR

- Functions normally in Window 1 but you will get a "FC" error in Window 2.
- The default startup setting of "PCLEAR is PCLEAR0 for maximum user memory, not the standard CoCo startup of PCLEAR4.
- If you are using an existing basic program that requires PCLEAR4 (A "FC" error is the indicator.) edit PCLEAR4 into the start of the program. IF the existing program is in Window 2 do a PCLEAR4 in Window 1 first, because Window 2 does not accept the PCLEAR command.
- The "Pclear" command has been improved. "Pclear" under BIG BASIC now will work with any number from 0 to 17, provided sufficient memory is available in Window 1 to accomodate the selection. With a 1-block Window 2 the highest usable "Pclear" number is 13.
- If you are not using the low-res graphics, "PCLEAR" can be used to protect an area of memory for machine language subroutines in low memory in Window 1.

FILES

- Functions normally in WINDOW 1 but you will get a "FC" error in Window 2.

DSKINI

- Functions in Window 1 or a no-window status only. Gives a "FC" error in Window 2. Window 1 must contain at least 8K of memory or computer must be coldstarted after "DSKINI".

FIELD

- While a disk data file can remain "Open" when you transfer between windows, the FIELD command uses the operating system "stack" for its special field variables. The "stack" is reset during window switching. Therefore you must repeat the FIELD command in each window if you want a data file to be accessed by both.

ON ERR GOTO & BRK GOTO

- These two commands are cancelled by the "Stack" reset when you transfer between windows. Therefore, they must be repeated on entry into a new window if you are using them.

APPENDIX B: MEMORY BLOCK TABLE

Memory is managed in a CoCo3 in blocks of 8K each. (1K = 1024 bytes; 8K = 8192 bytes)

The top 8 blocks are the normally used 64K.

BLOCK	USE	ACCESS by "BIG BASIC"
63	CoCo3 Extra Basic	No
62	Disk Basic	No
61	Color Basic	No
60	Extended Basic	No
59	User Programs/Variables	Yes
58	User Programs/Variables	Yes
57	User Programs/Variables	Yes
56	Basic's Variables/Buffers	No
55	Free Memory	Yes
54	Both 40 & 80 Column Hi-Res Text Screens	Yes if not using Hi-Res Text Screens
53	Free Memory/Originally a Secondary Stack Area	Yes (Stack was Moved)
52	HGET/HPUT Buffer	Yes if not using HGET/HPUT
51	Hi-Res Graphics Screen	Yes if not using Graphics Screen
50	Hi-Res Graphics Screen	Yes if not using Graphics Screen
49	Hi-Res Graphics Screen	Yes if not using Graphics Screen
48	Hi-Res Graphics Screen	Yes if not using Graphics Screen
47 to 0	448K of Free Memory only in the 512K CoCo3	Yes

(NOTE: The CoCo3 Owner's Manual contains a memory map that correlates with above block areas.)

COLDSTART

Different methods are used to re-initialize the computer, or "COLDSTART" it, either to start up a new program or to recover from a "Crash".

A "Crash" can be defined as occurring when the computer's processor has become confused and either fails to respond to the user or malfunctions in subtle ways, often accompanied by a scrambled screen display.

The ultimate and most complete "COLDSTART" is to turn off the machine for at least 15 seconds, but other solutions are usually possible.

CoCo3 users should hold down the "ALT" and "CTRL" keys simultaneously and press the reset button. When the picture of the three authors of the operating system appears, release the two keys and press reset again.

* * * * * * * * * * * *

TO OUR VALUED CUSTOMERS:

Users of our programs are invited to comment. If we have missed some vital point, or something is not explained clearly enough, please contact us and tell us so we can ammend any reprints. We realize others may see things differently than us, and are concerned that users are satisfied with our programs.

For enquiries, comments, or assistance write:

DANOSOFT
P.O. Box 124, Station "A",
Mississauga, Ont. L5A 2Z7
Canada
Telephone: (416) 897-0121 .