OISTO
SUPER PRODUCTS

C~DOS

MANUFACTURED & DISTRIBUTED BY

C.R.C COMPUTER inc
C.R.C ORDINATEUR inc

10802 LAJEUNESSE suite 102
MONTREAL, QUEBEC, CANADA
H3L 2E8 (514) 383-5293

C~DOS

DISK BASIC SUMMARY

This is a short summary on each new DISK BASIC "command." You may also use any of the EXTENDED COLOR BASIC commands. (See *Getting Started with Extended Color BASIC* or the *Color Computer Quick Reference Card* for a complete listing.)

The first line gives the format to use in typing the command. The italicized words represent "parameters"—values which you can specify with the command.

This is the meaning of some of the parameters you may specify:

filename
All information stored on a disk must have a *filename*. The *filename* should be in this format:
name/extension:drive number
The *name* is mandatory. It must have 1 to 8 characters.
The *extension* is optional. It can have 1 to 3 characters.
The *drive number* is optional. If you do not use it when opening a disk file, the Computer will use drive 0 (or the drive specified in the DRIVE command).

number
This may be a number (1, 5.3), a numeric variable (A, BL), a numeric function (ABS(3)), or a numeric operation (5+3, A−7).

string
This may be characters ("B," "STRING"), a string variable (A$, BL$), a string function (LEFT$(S$, 5)), or a string operation ("M"+A$).

data
This may be *number* or *string*.

BASIC WORD

BACKUP *source drive* TO *destination drive*
Duplicates the contents of the *source drive* to the *destination drive*. If you only have one drive, specify it as the *source drive*. The Computer will prompt you to switch disks as it makes the backup copy. Executing this command will erase memory.
```
BACKUP 0 TO 1     BACKUP 0
```

CLOSE # *buffer, ...*
Closes communication to the *buffers* specified. (See OPEN for *buffer* numbers). If you omit the *buffer*, the Computer will close all open files.
```
CLOSE #1     CLOSE #1, #2
```

COPY *filename1* TO *filename2*
Copies the contents of *filename1* to *filename2*. Each *filename* must include an *extension*. (See format for *filenames* above.) Executing this command will erase memory.
```
COPY "FILE/BAS" TO "NEWFILE/BAS"
COPY "ORG/DAT:0" TO "ORG/DAT:1"
```


BASIC WORD

CVN(*string variable*)
Converts a 5-byte coded *string* (created by MKN$) back to the number it represents.
```
X=CVN(A$)
```

DIR*drive number*
Displays a directory of the disk in the *drive number* you specify. If you omit the *drive number*, the Computer will use drive 0. (Unless you use the DRIVE command to change this default.) This is a typical directory display:
```
MYPROG     BAS    0 B 3
YOURPROG   BAS    0 A 1
HERDATA    DATA   1 A 5
USRPROG    BIN    2 B 2
```
The first column is the name of the file. The second column is its extension. The third is the file type (0 = BASIC program, 1 = BASIC data file, 2 = machine language file, 3 = editor source file). The fourth column is the storage format (A = ASCII, B = Binary). The fifth column is the file length in granules.
```
DIR0     DIR
```

DRIVE *drive number*
Changes the drive default to the *drive number* you specify. If you do not use the DRIVE command, the Computer will default to drive 0.
```
DRIVE 1
```

DSKINI*drive number*
Formats a disk in the *drive number* you specify. Executing this command will erase memory.
```
DSKINI0     DSKINI1
```

DSKI$ *drive number, track, sector, string variable1, string variable2*
Inputs data from a certain *sector* within a certain *track* on the disk in *drive number*. The first 128 bytes of data are input into *string variable1;* the second 128 bytes into *string variable2*.
```
DSKI$ 0, 12, 3, M$, N$
```

DSKO$ *drive number, track, sector, string1, string2*
Outputs string data into the *sector, track,* and *drive number* you specify. *string1* is output into the first 128 bytes of the sector; *string2* is output into the second 128 bytes. Used improperly, this command could garble the contents of the disk.
```
DSKO$ 0, 2, 1, "FIRST DATA", "SECOND DATA"
```

EOF *(buffer)*
Returns a 0 if there is more data to be read in the *buffer* and a −1 if there is no more data in it. (See OPEN for *buffer* numbers.)
```
IF EOF(1) = -1 THEN CLOSE #1
```

FIELD # *buffer, field size* AS *field name,...*
Organizes the space within a direct access *buffer* into fields. (See OPEN for *buffer* numbers.) You specify the *size* and *name* of each *field*.
```
FIELD #1, 10 AS A$, 12 AS B$, 5 AS C$
```

BASIC WORD

FILES *buffer number, buffer size*
Tells the Computer how many buffers to reserve in memory *(buffer number)*, and the total bytes to reserve for these buffers *(buffer size)*. If you do not use FILES, the Computer will reserve enough memory space for buffers 1 and 2, and will reserve a total of 256 bytes for those buffers.
```
FILES 1, 1000    FILES 5
```

FREE*(drive number)*
Returns the number of free granules on the disk in the *drive number* you specify.
```
PRINT FREE(0)
```

GET # *buffer, record number*
Gets the next record or the *record number* you specify, and puts it in the *buffer*. (See OPEN for *buffer* numbers).
```
GET #1, 5    GET #2, 3
```

INPUT # *buffer, variable name,...*
Inputs data from the *buffer* you specify and assigns each data item in the *buffer* to the *variable name* you specify. (See OPEN for *buffer* numbers.)
```
INPUT #1, A$, B$
```

KILL *filename*
Deletes the *filename* you specify from the disk directory. (See the format for *filenames* above.) You must include the *extension* with the *filename*.
```
KILL "FILE/BAS"    "KILL FILE/DAT:1"
```

LINE INPUT # *buffer, data*
Inputs a line (all the *data* up to the ENTER character) from the *buffer* you specify. (See OPEN for *buffer* numbers).
```
LINE INPUT #1, X$
```

LOAD *filename*, R
Loads the BASIC program file you specify from a disk into memory. By including R, the Computer will RUN the program immediately after loading it. If your *filename* does not have an *extension*, the Computer assumes it is BAS. (See the format for *filenames* above.) Executing this command will erase memory.
```
LOAD "PROGRAM", R    LOAD "ACCTS/BAS:1"
```

LOADM *filename, offset address*
Loads a machine-language program file from disk. You can specify an *offset address* to add to the program's *loading address*. If your filename does not have an *extension*, the Computer assumes it is BIN. (See the format for *filenames* above.)
```
LOADM "PROG/BIN, 3522
```

LOC*(buffer)*
Returns the current record number of the buffer you specify. (See OPEN for *buffer* numbers.)
```
PRINT LOC(1)
```


BASIC WORD

LOF*(buffer)*
Returns the highest numbered record of the buffer you specify. (See OPEN for *buffer* numbers.)
```
FOR R = 1 TO LOF(1)
```

LSET *field name = data*
Left justifies the *data* within the *field name* you specify. If the *data* is larger than the field, the RIGHT characters will be truncated (chopped off).
```
LSET A$ = "BANANAS"    LSET B$ = T$
```

MERGE *filename*, R
Loads a program *file* from disk and merges it with the existing program in memory. If you include R, the Computer will immediately run the program after merging it. (See the format for *filenames* above.) The disk program *file* cannot be MERGEd unless it was SAVEd with the A (ASCII) option.
```
MERGE "SUB/BAS"    MERGE "NEW", R
```

MKN$*(number)*
Converts a *number* to a 5-byte coded string, for storage in a formatted disk file.
```
LSET B$ = MKN$(53678910)
```

OPEN *"mode," # buffer, filename, record length*
Opens a place in memory called a *buffer* which will communicate data to and from a certain device. The *buffers* and the devices they communicate with are:
0 — screen or printer (it is not necessary to open this buffer)
-1 — tape recorder
-2 — printer
1-15 — disk drive
The communication *modes* you can use are:
I — inputting data from a sequential access file
O — Outputting data to a sequential access file
D — Inputting or outputting data to a direct access file
The *filename* you use should be in the format defined above. If you do not give *filename* an *extension*, the Computer will give it the *extension* DAT.
If you are opening communication to a direct access file, you can also specify the *record length*. If you don't, the *record length* will be 256 bytes.
```
OPEN "O", #1, "FILE", 15
OPEN "I", #2 "CHGE/DAT"
```

PRINT # *buffer, data list*
PRINTs the *data* to the *buffer*. (See OPEN for *buffer* numbers.) You may use a comma or a semi-colon to separate each item in the *data list*.
```
PRINT #1, "DATA"
```

PRINT # *buffer,* USING *format; data list*
Prints data to the *buffer* using the *format* you specify. The *format* is a *string* which can either specify a *numerical* or *string format*.
numerical formats may consist of any of the following:

#	sets the field of a number
.	formats a decimal point

BASIC WORD

,	formats a comma every third number
**	fills leading spaces with asterisks
$	places $ ahead of number
$$	floating dollar sign
+	in first position, causes sign to be printed before number; in last position causes sign to be printed after the number
^^^^	prints number in exponential notation
-	prints a minus sign after negative numbers

```
PRINT USING #1, "##.#"; 53.76
PRINT USING #2, "**$#.##-"; -3.678
```

string formats may consist of either:

% % fields the length of a string.

! ! prints the first character of the string

```
PRINT USING #1, "!"; "WHITE"
PRINT USING #1, "% %"; "YELLOW"
```

See *Going Ahead With Extended Color BASIC* for more information on the formats.

PUT # *buffer, record number*
Assigns a *record number* to the data in the *buffer*. If you do not specify a *record number*, the Computer will assign it to the current record. (See OPEN for *buffer* numbers.)

```
PUT #2, 3    PUT #1, 4
```

RENAME *old filename* TO *new filename*
Renames a file on disk to a *new filename*. You must specify the extension of both *filenames*.

```
RENAME "MFILE/DAT:1" TO "BFILE/DAT:1"
```

RSET *field name = data*
Right justifies the *data* within the field name you specify. If the *data* is larger than the field, the RIGHT characters will be truncated (the same as with LSET).

```
RSET M$ = "SOAP"
```

RUN *filename*, R
Loads *filename* from disk and runs it. If R is included, all open files will remain open. (See the format for *filenames* above.)

```
RUN "FILE'    RUN "PROG/BAS", R
```

SAVE *filename*, A
Saves *filename* on disk. If you do not give *filename* an *extension*, the Computer will give it the extension BAS. By using the A option, your program will be saved in ASCII format. (See the format for *filenames* above.)

```
SAVE "PROG/BAS"    SAVE "TEST:1", A
```

SAVEM *filename, first address, last address, execution address*
Saves *filename* — a machine language program beginning at *first address* (in memory) and ending at *last address*. You also specify the *address* in which it will be *executed*. If you do not give *filename* an *extension*, the Computer will give it the extension BIN. (See the format for filenames above.)

```
SAVEM "FILE/BIN:1", &H5200, &H5800, &H5300
```


BASIC WORD

UNLOAD *drive number*
Closes any open files in the *drive number* you specify. If you don't specify a *drive number* the Computer will use drive 0 (or the *drive number* you specified with DRIVE).

```
UNLOAD 0 UNLOAD
```

VERIFY ON
VERIFY OFF
Turns the verify function ON or OFF. When VERIFY is ON, the Computer will verify all disk writes.

WRITE # *buffer, data list*
Writes the data to the *buffer* you specify. (See OPEN for *buffer* numbers.) Use a comma to separate each item in the *data list*.

```
WRITE #1, A$, B$, C
```

ERROR MESSAGES

/0 Division by zero. The Computer was asked to divide a number by 0, which is impossible. You could also get this error message if you do not enclose a filename in quotation marks.

AE File Already Exists. You are trying to RENAME or COPY a file to a filename which Already Exists.

AO Attempt to open a data file which is Already Open.

BR Bad Record Number. You have used an impossible record number in your PUT or GET line. Either it is too low (less than 1) or too high (higher than the maximum number of records the Computer can fit on the disk). Use a different record number in the PUT or GET line, or assign a smaller record length in the OPEN iine.

BS Bad Subscript. The subscripts in an array are out of range. Use DIM to dimension the array. For example, if you have A(12) in your program, without a preceding DIM line which dimensions array A for 12 or more elements, you will get this error.

CN Can't continue. If you use the command CONT and you are at the END of the program, you will get this error.

DD Attempt to redimension an array. An array can only be dimensioned once. For example, you cannot have DIM A(12) and DIM A(50) in the same program.

DF Disk Full. The Disk you are trying to store your file on is Full. Use another disk.

DN This is either a Drive Number or Device Number error.

Drive Number Error. You are using a drive number higher than 3. You will also get this error if you do not specify a drive number when using DSKINI or BACKUP. If you have only one drive specify drive 0 with these two commands (DSKINI0 or BACKUP 0).

Device Number error. You are using more buffers than the Computer has reserved. Use FILES to reserve more. You might also get this error if you use a nonexisting buffer number (such as buffer # – 3) or omit the buffer (such as FIELD 1 AS A$ rather than FIELD #1, 1 AS A$).

DS Direct Statement. There is a direct statement in the data file. This could be caused if you load a program with no line numbers.

ER Write or Input past End of Record (direct access only). You are attempting to PUT more data in the record than it can hold or INPUT more data than it contains.

FC Illegal Function Call. This happens when you use a parameter (number) with a BASIC word that is out of range. For example SOUND (260,260) or CLS(10) will cause this error. Also RIGHT$(S$,20), when there are only 10 characters in S$, would cause it. Other examples are a negative subscript, such as A(–1), or a USR call before the address has been POKEd in.

FD Bad File Data. This error occurs when you PRINT data to a file, or INPUT data from the file, using the wrong type of variable for the corresponding data. For example, INPUT # 1, A, when the data in the file is a string, causes this error.

FM Bad File Mode. You have specified the wrong file mode ("O," "I," or "D") in your OPEN line for what you are attempting to do. For example, you are attempting to GET a record from a file OPENed for "I" (use "D") or WRITE data to a file OPENed for "I" (use "O").

FN Bad File Name. You used an unacceptable format to name your file.

FO Field Overflow. The field length is longer than the record length.

FS Bad File Structure. There is something wrong with your disk file. Either the data was written incorrectly or the directory track on the disk is bad. See IO for instructions on what to do.

ID Illegal Direct statement. You can only use INPUT as a line in the program, not as a command line.

IE Input past End of file. Use EOF or LOF to check to see when you've reached the end of the file. When you have, CLOSE it.

IO Input/Output error. The Computer is having trouble inputting or outputting information to the disk.

APPENDIX

(1) Make sure there is a disk inserted properly in the indicated drive and the drive door is closed.
(2) If you still get this error, there might be something wrong with your disk. Try reinserting the disk first. Then try using a different one or reformatting it. (Remember that reformatting a disk erases its contents.)
(3) If you still get this error, you probably have a problem with the Computer System itself. Call the Radio Shack Repair Center.

This error could also be caused by input/output problems with another device, such as the tape recorder.

LS String too Long. A string may only be 255 characters.

NE The Computer can't find the disk file you want. Check the disk's directory to see if the file is there. If you have more than one disk drive, you might not have included the appropriate drive number in the filename. If you are using COPY, KILL, or RENAME (discussed in the next chapter), you might have left off the extension.

NF NEXT without FOR. NEXT is being used without a matching FOR statement. This error also occurs when you have the NEXT lines reversed in a nested loop.

NO File Not Open. You cannot input or output data to a file until you have OPENed it.

OB Out of Buffer space. Use FILES to reserve more space.

OD Out of Data. A READ was executed with insufficient DATA for it to READ. A DATA statement may have been left out of the program.

OM Out of Memory. All available memory has been used or reserved.

OS Out of String Space. There is not enough space in memory to do your string operations. Use CLEAR at the beginning of your program to reserve more string space.

OV Overflow. The number is too large for the Computer to handle.

RG RETURN without GOSUB. A RETURN line is in your program with no matching GOSUB.

SE Set to non-fielded string. The field in which you are attempting to LSET or RSET data in has not yet been FIELDed. Check the FIELD line.

SN Syntax error. This could result from a misspelled command, incorrect punctuation, open parenthesis, or an illegal character. Type the program line or command over.

ST String formula too complex. A string operation was too complex to handle. Break up the operation into shorter steps.

TM Type Mismatch. This occurs when you try to assign numeric data to a string variable (A$ = 3) or string data to a numeric variable (A = "DATA"). This could also occur if you do not enclose a filename in quotes.

UL Undefined Line. You have a GOTO, GOSUB, or other branching line in the program asking the Computer to go to a nonexisting line number.

VF Verification. You will only get the error when you have the VERIFY command ON and are writing to a disk. The Computer is informing you that there is a flaw in what it wrote. See IO for instructions on what to do.

WP Write Protected. You are trying to store information on a disk which is Write Protected. Either take the label off the write protect notch or use a different disk. If your disk is not Write Protected, then there is an input/output problem. See IO for instructions on what to do about this.

CDOS COMMANDS

The following is a list and description of the extra commands of CDOS.

COLD:

The COLD command is equal to turning the computer off then on again. It resets all the pointers, stack, variables, and PIA's. It does not change what is in memory other than what it needs to initialize Disk Extended Basic. It is the same as typing in, POKE 113,0 : EXEC 41195

Example - COLD

DOS:

The DOS command has two functions. The first is to boot (or start up) from disk, an alternate Disk Operating System such as OS9. Some application programs, such as the VIP series or DynaCalc, will auto-start with the command DOS. The second function for the DOS command is to change the Active Chip (or DOS) on the DISTO Super Controller card. The argument X can be a value from 1 to 3. This will cause the computer to; 1)Change the Active Chip to the value given by the argument X, 2) Simulate a Cold start and 3) Jump to the new DOS. Note, DOS X will only work with the DISTO Super controller and another DOS in the appropriate slot.

Examples - DOS1, DOS3

DTT:

The Disk To Tape utility is used when a file on disk is to be transferred to cassette. The first prompted asks if All files are to be transferred or only Selected files are to be transferred. The All option then prompts for how many copies of each is needed. The Select option

will go through the whole directory one at a time, and prompt the user for how many copies of that file are to be transferred to cassette. A "0" answer to the prompt will not transfer that file to cassette and proceed to the next file. A "Q" answer will end the transfer and exit back to Basic.

Example - DTT

TTD:

The Tape To Disk utility is used when a file on cassette is to be transferred to disk. When all files are transferred, hit the reset button to return to Basic.

Example - TTD

Fast:

The Fast command is used to put the CPU into the dual speed mode. This is better known as the "Fast Poke" mode. Not all computers will work with the FAST command, try it and find out. Warning, do not attempt any I/O to disk, cassette or RS232 while in the FAST mode. This command is the equivalent to POKE 65495,0.

Example - FAST

SLOW:

The SLOW mode will put the CPU in the regular speed mode. It is the complement to the FAST command. It has no effect if the CPU is already in the regular mode. This command is the equivalent to POKE 65494,0.

Example - SLOW

MRUN:

The MRUN command does three operations; 1)

loads the specified machine language program, 2) turns the drive motors off and 3) EXECutes the specified machine language program. This command does the same as LOADM and EXEC all in one.

Examples - MRUN "MYFILE", MRUN "THAT/BIN:2",1024

POUT:

This command is used to route all printer output to the Centronics Parallel Printer Adapter. (Note, the Pprint adapter or any adapter that includes the parallel printer option must be properly installed in the DISTO Super Controller in order for the POUT command to work. See the appropriate manual for installation instructions.) Only BASIC programs and those using the proper Print Hook will work with the POUT command. This command has two forms:

1 POUT ON - Routes the printer data to the Pprint adapter.

2 POUT OFF - Routes the printer data to the COCO's serial port.

Examples - POUT ON
- 10 POUT OFF

RAM:

The command RAM will transfer the contents of Basic, Extended Basic and Disk Basic into the upper half of a 64K computer. It will then return to Basic, leaving the computer in the 64K RAM mode.

Example - RAM

UNDO:

The UNDO command will remove all extra commands and unhook the screen editor. In effect, this command will make this DOS, Radio Shack DOS

compatible.

Example - UNDO

SCRED:

The SCReen EDitor, is used to edit Basic lines or direct commands that appear on the screen. To invoke the SCRED, hit the SHIFT key and the "@" key simultaneously. The cursor will immediately change to the SCRED cursor. In this mode, the following options are available;

RIGHT ARROW - Moves the cursor one character to the right.

LEFT ARROW - Moves the cursor one character to the left.

UP ARROW - Moves the cursor up one line.

DOWN ARROW - Moves the cursor down one line.

"@" KEY - Enters the character that the cursor is presently sitting on, into Basic's line input buffer. All characters in Basic's input buffer will be processed only after the ENTER key has been pressed.

BREAK KEY - Clears Basic's input buffer and exits the SCRED mode.

ENTER KEY - Exits the SCRED mode and passes the buffer to BASIC.

CLEAR KEY - Has no effect in the SCRED mode.

INSERT - To insert a character (s) into an existing line;

1) Move the cursor, using the arrow keys only, to the beginning of line, line number

included, to be corrected.

Example: 10 PRINT "TEST"

2) Move the cursor, using the "@" key. This will enter all characters that are sitting under the cursor, into the BUFFER.

Example: 10 PRINT "TEST"

3) Now type in the extra characters. Note that the characters will not appear on the screen, but will be processed after the (ENTER) key has been pressed.

Example: 10 PRINT "TEST" type in "ING"

4) Finally, move the cursor using the "@" key to the end of the line and hit the (ENTER) key. Again nothing will change on the screen. List the line in question to confirm that all is well.

```
Example:    10 PRINT "TEST"<ENTER>
            LIST
            10 PRINT "TESTING"
```

DELETE To delete character (s) in an already existing line;

1) Follow steps 1 and 2 of the INSERT mode.

2) Instead of typing in new characters, skip over the unwanted characters using the RIGHT ARROW.

3) Follow step 4 of the INSERT mode.

DUPLICATE - To duplicate an existing line to another line.

1) Type in a new line number.

Example: 20

2) Move the cursor using the ARROWS to just past the line number of the line to be duplicated.

```
Example:  10 PRINT "TESTING"
          20
```

3) Move the cursor using the "@" key over the line to be duplicated. Hit (ENTER) when the end of line is reached and list your new line.

Example: 10 PRINT "TESTING" (ENTER)

20

Remember, using the "@" key will enter any character that the cursor is sitting on into the BUFFER, and using the RIGHT ARROW to skip over a character will effectively delete that character in that line. In the SCRED mode, any key which is held down for longer than about one second will auto-repeat. To remove the SCRED option from DOS, type the UNDO command.

CLOCK:

The CLOCK command is used to access the real time, date, and year. (Note that the Rtime adapter or any adapter that includes the clock option, must be properly installed in the DISTO Super Controller in order for the CLOCK command to work. See the appropriate manual for installation instructions.) This command has 4 forms.

1 CLOCK - This command will print out the contents of the clock as follows;

XX HH:MM:SS DAY DD/MO/YY

all references to the "clock contents" or the "clock contents format" will be in the above format. The length of the clock contents will always be 24 characters where;

XX - AM or PM in the 12 hour format.
   - blank in the 24 hour format.
HH - hours 1 to 12 in the 12 hour format.
   - hours 0 to 23 in the 24 hour format.
MM - minutes 0 to 59.
SS - seconds 0 to 59.
DAY - SUN, MON, TUE, WED, THU, FRI or SAT.
DD - date 0 to 31.
MO - months 0 to 11.
YY - years 0 to 99.

2 CLOCK ON - continuously displays the contents of the clock on the the top line of the screen, right justified. This mode uses the COCO's 60 hz interrupt to update the time. If the interrupts are masked, the clock will not update.

3 CLOCK OFF - Terminates the CLOCK ON mode.

4 CLOCK (string variable name) - Transfers the contents of the clock into the specified string of which will always be 24 characters in length. An example of this mode is;

```
10 CLOCK A$
20 PRINT A$
```

This string can be any string variable name that can be used with BASIC, arrays included. If the string already exists, the contents will be changed and the old data will be lost. You cannot change the contents of the clock using strings. To change the contents of the clock, a short Basic program called "SET.BAS" must be used. This program is used whenever the contents of the clock must be changed. LOAD and RUN the program. You

will be greeted by the contents of the clock. To change the contents of the clock, enter the desired values. Use the top line as a guide on how to enter the values in their proper sequence. Use the space bar and the right arrow to position the values. Hit the ENTER key when finished. Use the CLOCK command to check if the desired time is displayed. Run the program again if the time is wrong. Remember to use the clock contents format shown above or the wrong data will be transferred to the clock. Note that the seconds cannot be set, they will, however reset to zero if the value entered for seconds is 00. The clock chip has a leap year flag. The flag must be set after January 1 and before February 28 of any leap year. The software to do this is in the SET.BAS program. All that has to be done is RUN the program once in that time period.

D80:

The D80 command is used to activate and de-activate to 80 column display mode. (Note that the DISPLAY 80 adapter must be properly installed in the DISTO Super Controller in order for the D80 command to work. See the DISPLAY 80 manual for installation instructions.) This command has two forms:

1 D80 ON - Used to initialize and use the 80 column display. All screen characters will now appear on the 80 column display. See the DISPLAY 80manual for more details on the display options.

2 D80 OFF - Used to turn the 80 column display off and return to the COCO's 32 column display.

Examples : 10 D80 ON
           D80 OFF

RDISK:

The RDISK command is available only on CDOS+ version 4.0 or later. In order to use the RDISK command, a DISTO SUPER RAMDISK and a Multi-Pak Interface is needed. See your RAMDISK manual for proper installation. In order to keep compatibility with Radio Shack DOS, the RDISK command can only use about 172K of the RAMDISK. Using the RDISK command emulates one Disk Drive. This drive can be any number from 0 to 3. When selected, that real disk drive connected to your computer will no longer respond. Instead, any access to the selected drive will be routed to the RAMDISK. The RDISK command needs 3 parameters.

The first parameter is the drive number the RAMDISK will emulate. The valid drive numbers are from 0 to 3. The next parameter is the slot number your RAMDISK hardware is plugged in. The valid numbers for this are from 1 to 4. The final parameter is to format (DSKINI) the RAMDISK. The possible answers are "Y" for yes or "N" for no. All disks need to be formatted (DSKINIed) before you can use them, including the RAMDISK. You must answer yes the first time and after every time you turn the computer off, in order to use the RAMDISK properly.

Examples : RDISK 12Y
RDISK 31N

The first example sets your RAMDISK to drive 1, must be in slot #2 and is formatted (DSKINIed). The second example sets your RAMDISK to drive 3, must be in slot #1 and is not formatted. This is needed if you want to change the drive number and not the contents. Warning: Using the RAMDISK command on a slot # when there is no RAMDISK in that slot can cause un-predictable results.

The SUPER RAMDISK and the RDISK commands are compatible only to programs that use the DSKCON system call and do not use any variables in the Cassette Buffer. The RDISK command is compatible with the popular graphics program, COCOMAX II. Backup your working copy of COCOMAX II to the RAMDISK, change the RAMDISK drive number to 0, then RUN "COCOMAX". Example: plug the COCOMAX cartridge into slot #1, insert the RAMDISK into slot #2, then type;

```
RDISK 12Y (ENTER)
BACKUP 0 TO 1 (ENTER)
RDISK 02N (ENTER)
RUN "COCOMAX" (ENTER)
```

After you are finished using COCOMAX, don't forget to transfer all the files you created or changed, to a real disk before you turn off your computer.

NOTE : C-DOS IS AVAILABLE IN MANY
DIFFERENT VERSION.

FROM; 6mSEC. Step rate to 30mSEC.
Single or double sided

To find out which version you have,look at the C-DOS OPTION on the package.