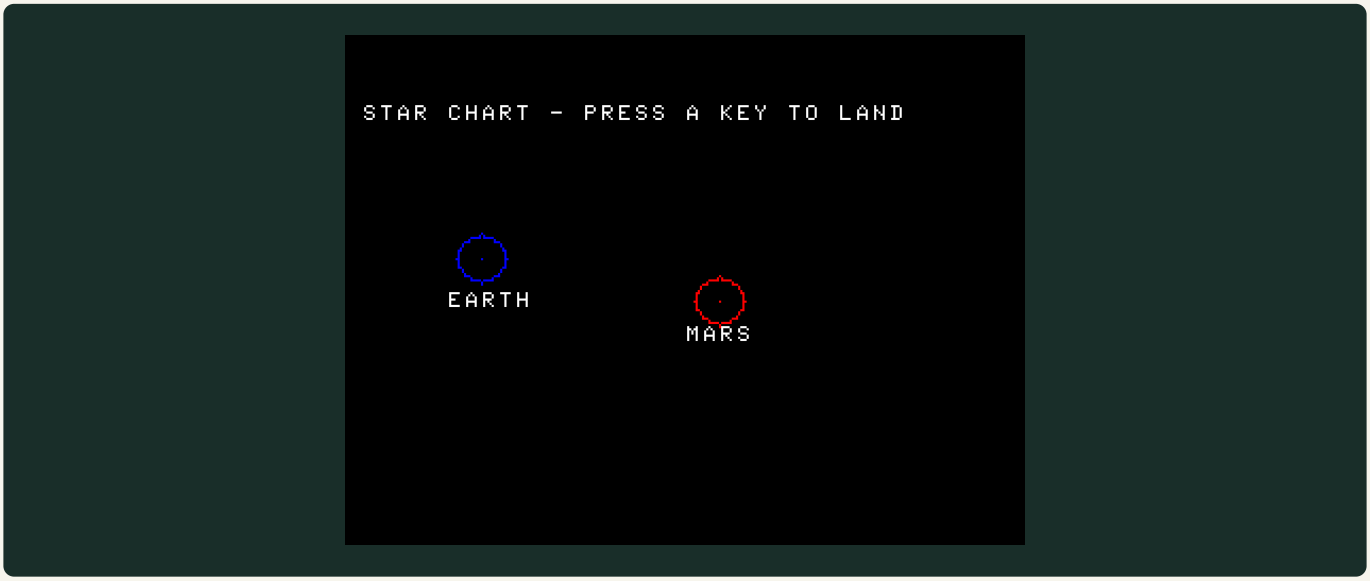
Programming in CoCo BASIC on the Color Computer 3

Earth or Mars? Build a flight deck, hunt down a supply leak, and leave a log that the next crew can read after you shut down the CoCo.



Planet Explorer running on a CoCo 3, with Earth in blue and Mars in red.

Brian K. Holman

Book 3 · VCC Plus 3.7.1

© Brian K. Holman 2026

Contents

Make the CoCo answer . . . 3
Choose a destination . . . 5
Count the supplies . . . 7
Find the missing supply . . . 9
Give a chart its names . . . 13
Give a routine a job . . . 15
Design a flight deck . . . 17
Draw a planet you can find again . . . 19
Listen for a signal . . . 24
Keep a log past bedtime . . . 26
Take the voyage to another machine . . . 30
Fly, land, power down, remember . . . 32
BASIC at your elbow . . . 34
Where BASIC meets the machine . . . 72
What did that error mean? . . . 74
Publication and artwork notices . . . 79

Make the CoCo answer

Chapter 1 of 12

Your flight computer is awake. Before you let it steer, make it show you how it thinks. You can ask three questions with almost the same keys and get three different answers.

Try the keyboard at OK

Start with **Disk BASIC** in a practice session. If you have a program in the CoCo that you want to keep, save it before using the numbered lines below. Click inside the CoCo display. When you see `OK`, type each of these lines and press **Return** after each one. Predict the next answer before you press the key.

```
PRINT 2+3*4
PRINT (2+3)*4
PRINT "2+3*4"
```

The first answer is **14**. The star means multiply. BASIC multiplies 3 by 4 first, then adds 2. The second answer is **20**. Parentheses tell BASIC to finish the addition first. The third answer is `2+3*4`, right down to the star. Quotation marks tell BASIC to take those five characters as **text**. Programmers call a piece of text a *string*.

An instruction you type this way is an **immediate command**. BASIC carries it out as soon as you press Return. Use the Mac keyboard as the CoCo keyboard while the display has focus. If typing moves a Mac control instead, click the CoCo display again.

Give tomorrow’s pilot a message

At the practice prompt, enter `NEW` to clear the old listing from guest memory. Type these two lines, pressing Return after each.

```
20 PRINT "PLANET EXPLORER"
30 PRINT "1 EARTH   2 MARS"
```

What happened to the menu? You saw your typing, then a new input line. The numbers tell BASIC to store the instructions. Type `LIST` to see what it kept. Now type `RUN`. This time the CoCo prints the title, prints the two choices, and says `OK`. You can type `RUN` again any time you want that menu back.

Slip a line between the two you already have. Enter `25 PRINT "WELCOME ABOARD"`, then `RUN`. The welcome appears between the title and the destinations. BASIC follows the **numbers**, even though you typed line 25 last. Enter just `25` and press Return. Type `LIST` again. Lines 20 and 30 remain. Entering a bare line number at the guest prompt deletes that line. An entire replacement line with the same number changes it. This is why you often see gaps between BASIC line numbers.

Keep a Mac copy

Open **Develop → BASIC Development…**. Choose **Read from CoCo**. If the editor is holding work you want, use its save option first. Look for the same two numbered lines you saw with LIST. Choose **Save As…** in the editor’s file actions and give your Mac source a name such as **Menu.bas**.

Place	What lives there	How to find it
CoCo RAM	Instructions the guest BASIC interpreter can run, plus working values while it runs	LIST at the guest prompt shows the instructions.

Place	What lives there	How to find it
Mac editor	Editable text, including any changes you have typed since the last transfer	Read from CoCo pulls the current guest listing into the editor. Save keeps the text in a Mac file.

Load into CoCo sends the editor's listing to guest RAM and leaves BASIC ready. The editor's **Run** sends it and starts it. Either operation can replace the program in guest memory; read the replacement prompt before choosing. The **Run** control in the main emulator window resumes the entire machine after a pause. Use the editor's Run when you want the BASIC listing to start at the beginning.

In the next lesson, you'll add the missing part of the menu: the CoCo will wait for a passenger to choose a planet.

Choose a destination

Chapter 2 of 12

A good pilot can change their mind. A good menu can survive a slip of the finger. Let's type the wrong answer on purpose and see whether the flight computer gives us another chance.

Fly the menu

Open HELM.bas from the Manual Library. It opens an editable example. In a browser, save the file and open it with **Develop → BASIC Development... → Open...**. Choose the editor's **Run**. If you want to replace the practice listing from chapter 1, confirm. Click the CoCo display.

1. At **SET COURSE**, enter **X** and press Return. Look for **TRY 1 OR 2** and another question.
2. Enter **1** and press Return. Look for **BOUND FOR EARTH** and `OK`.
3. Run again and enter **2**. Now you're **BOUND FOR MARS**.

Here is the entire helm. You already know the two lines that print the menu.

```
10 WIDTH 40:CLS
20 PRINT "PLANET EXPLORER"
30 PRINT "1 EARTH   2 MARS"
40 INPUT "SET COURSE";C$
50 IF C$="1" THEN P$="EARTH":GOTO 90
60 IF C$="2" THEN P$="MARS":GOTO 90
70 PRINT "TRY 1 OR 2"
80 GOTO 40
90 PRINT "BOUND FOR ";P$
100 END
```

Follow the choice with your finger

`WIDTH 40` puts the CoCo 3 in its 40-column text mode. The colon joins two instructions on one numbered line. `CLS` clears the screen before the sign goes up.

On line 40, **INPUT** prints a prompt and waits for a reply followed by Return. `C$` is a *variable*: BASIC remembers a value under that name. The dollar sign says its value is a string. When you type **2**, BASIC stores the character "2". That's why the comparison on line 60 has quotes around the digit. A room number or a menu choice can be a perfectly good piece of text.

Put one finger on line 50. The equal sign in `IF C$="1"` asks a question. If the answer is true, the part after **THEN** assigns a name to `P$` and sends execution to line 90. If it's false, BASIC skips the rest of line 50 and tries line 60. The equal sign in `P$="MARS"` has another job: it puts a value into a variable. Watch for whether you're reading a question or an assignment.

Reply	Lines after INPUT	Why
1	50 → 90 → 100	The first comparison succeeds and the jump skips the rest of the menu.
2	50 → 60 → 90 → 100	The second comparison succeeds.
X	50 → 60 → 70 → 80 → 40	Both comparisons fail. The program prints a hint and jumps back to INPUT.

The semicolon in line 90 prints two things side by side. The space at the end of the quoted words leaves room before the planet name. An extra menu trip costs almost nothing when all the program has to do is jump back a few lines.

To try the same decision in another language, give Mars a key in C. That version reads one keystroke as soon as it arrives. To see where the processor keeps the chosen color and message, catch the two assembly routes where they meet.

Ask a question about the answer

After the Mars run, stay at the guest's `OK` prompt. Your listing has ended, but `P$` still holds the name from that flight. Try these immediate commands one at a time.

```
PRINT LEN(P$)
PRINT LEFT$(P$,1)
PRINT RIGHT$(P$,2)
```

You should get **4**, **M**, and **RS**. A *function* takes a value and produces a result. LEN counts characters. LEFT$ and RIGHT$ give you pieces of the original string. None of those questions changed the name. In the chart-data lesson, BASIC will keep several names at the same time.

Count the supplies

Chapter 3 of 12

Lay six imaginary food packs on the table. Spend one on each of three turns. How many do you expect after turn one? After turn three? Your prediction is a perfectly good way to test a computer program.

Keep one number in working memory

Open RATIONS.bas and read the listing **before** you run it. A name without a dollar sign can hold a number. Here `S` stands for supplies and `T` stands for the current turn.

```
10 WIDTH 40:CLS
20 S=6
30 FOR T=1 TO 3
40 S=S-1
60 PRINT "TURN";T;" SUPPLIES";S
70 NEXT T
80 END
```

Read line 40 aloud as “take the old number of supplies, subtract one, and store the result back under S.” BASIC works out the expression on the right before it replaces the old value on the left. On the first trip through the line the arithmetic is six minus one. On the next trip it is five minus one. The same line can do a different calculation each time.

Take another turn

FOR sets `T` to 1 and gives the loop a last turn of 3. Line 40 spends one pack; line 60 prints the current turn and what is left. At **NEXT T**, BASIC adds one to the counter. If another turn fits, BASIC goes back to the first instruction inside the loop. Otherwise it carries on after NEXT.

Choose **Run** in the editor. Compare your pencil-and-paper answer with the guest screen.

Turn when line 60 prints	Supplies arriving at line 40	Supplies printed at line 60
1	6	5
2	5	4
3	4	3

At `OK`, type `PRINT S,T`. The CoCo prints **3** and **4** in separate fields. The **4** is a surprise worth keeping: NEXT moved the counter on one final time and tested it. Four was past the end. There was never a fourth trip through line 60. A comma in PRINT separates fields; the semicolons in the original listing keep the labels and numbers close together.

Check the arithmetic before changing a program

At the prompt, try `PRINT 6-1*3`. The answer, **3**, agrees with the final cargo count. That single calculation tells you the amount left at the end. The loop also tells a crew member how much is left *during* each turn. Try `PRINT (6-1)*3` and the answer becomes **15**. The parentheses asked the CoCo to repeat five packs three times.

Choose the size of one step

Try these two loops at the prompt. The colon lets several instructions share a line. The semicolon after J keeps the numbers on the same row.

```
FOR J=3 TO 1 STEP -1:PRINT J;:NEXT J
FOR J=2 TO 6 STEP 2:PRINT J;:NEXT J
```

The first loop prints **3 2 1**. The second prints **2 4 6**. In RATIONS, NEXT used the usual step of one. Here you chose the size and direction. The end value belongs to the loop when the step lands on it.

For another kind of number question, try `PRINT ABS(-3);INT(6.5)`. The results are **3** and **6**. ABS measures how far a number is from zero. INT takes the greatest whole number at or below its input. For other built-in number tools, the language reference gives each one's inputs and result. The next flight has an arithmetic problem you can catch with a watch.

Find the missing supply

Chapter 4 of 12

Some bugs make BASIC stop and print an error. This one is trickier. Every line is legal BASIC, the ship makes all three turns, and the crew ends up hungry.

Reproduce the fault

Open LEAK.bas and choose **Run**. You'll see 4, 2 and 0 supplies left after the three turns. In chapter 3 the same six packs lasted the whole trip with three to spare. Where did the first extra pack go?

```
10 WIDTH 40:CLS
20 S=6
30 FOR T=1 TO 3
40 S=S-1
50 S=S-1
60 PRINT "TURN";T;" SUPPLIES";S
70 NEXT T
80 END
```

BASIC is quick enough to finish the flight before you can read the first result. Use a **breakpoint** to ask the editor to wait for you at a particular instruction. That lets you investigate while the clues are still in RAM.

Stop before the subtraction

1. In the editor, click the breakpoint circle in the margin beside **50**. To set it from the keyboard, put the insertion point on that line and activate **Toggle BASIC breakpoint at insertion point**.
2. Open **Variables and Watches**. Select **Watches**, add `S`, then add `T`. The editor will fill in their values when BASIC stops.
3. Choose the **editor's Run** again. The highlight should land on line 50. Read both watches before doing anything else. **T is 1. S is already 5.** Line 40 has spent one pack. Line 60 has yet to print the first turn.
4. Choose **Step Into BASIC statement** once. The highlight moves to line 60. Read the watches again. **T is still 1. S is now 4.** One crew, one turn, two subtractions. You found the leak.

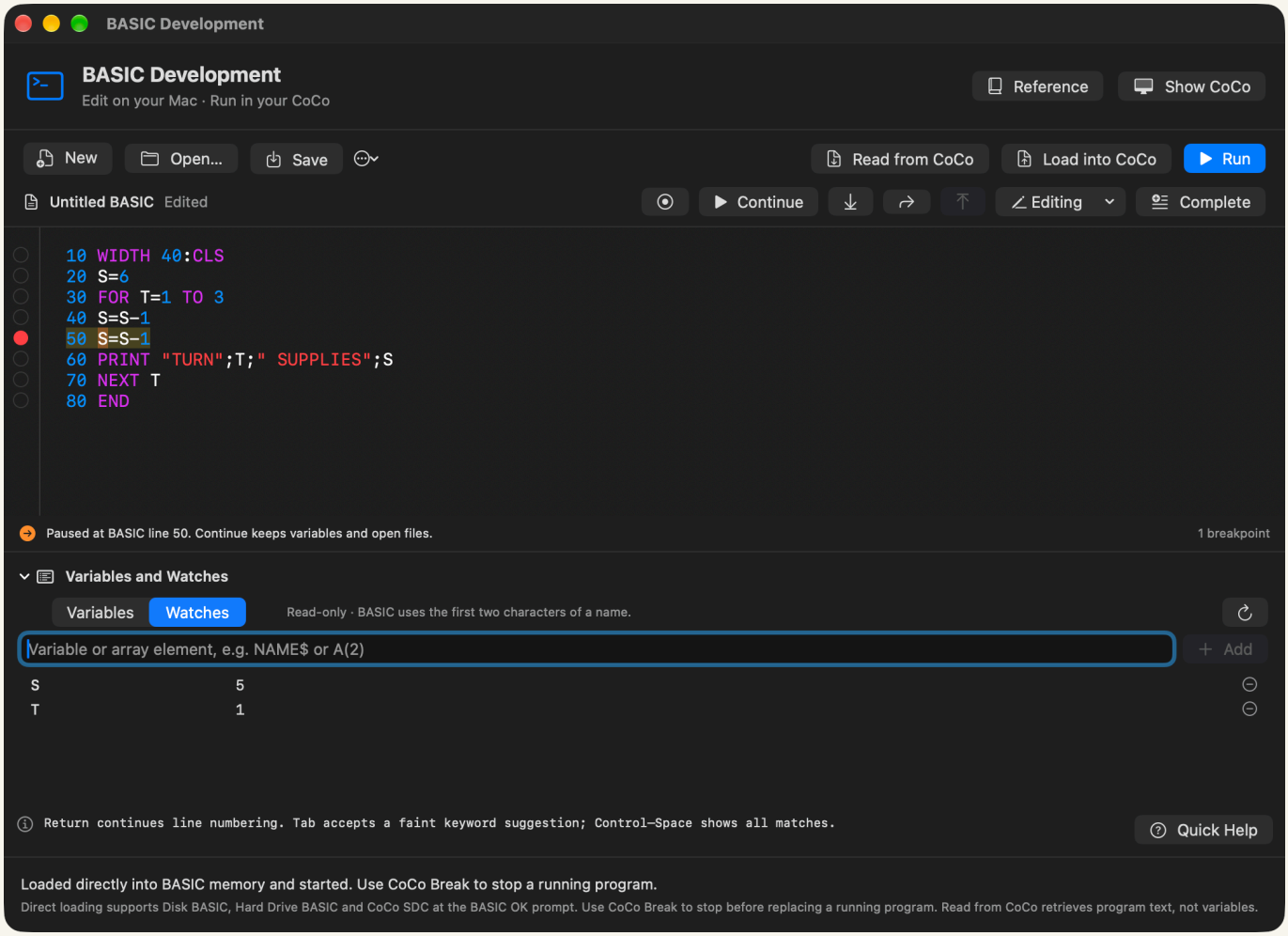


Figure 1: Before line 50 runs, the Watches list shows S at 5 and T at 1. The first turn is still in progress.

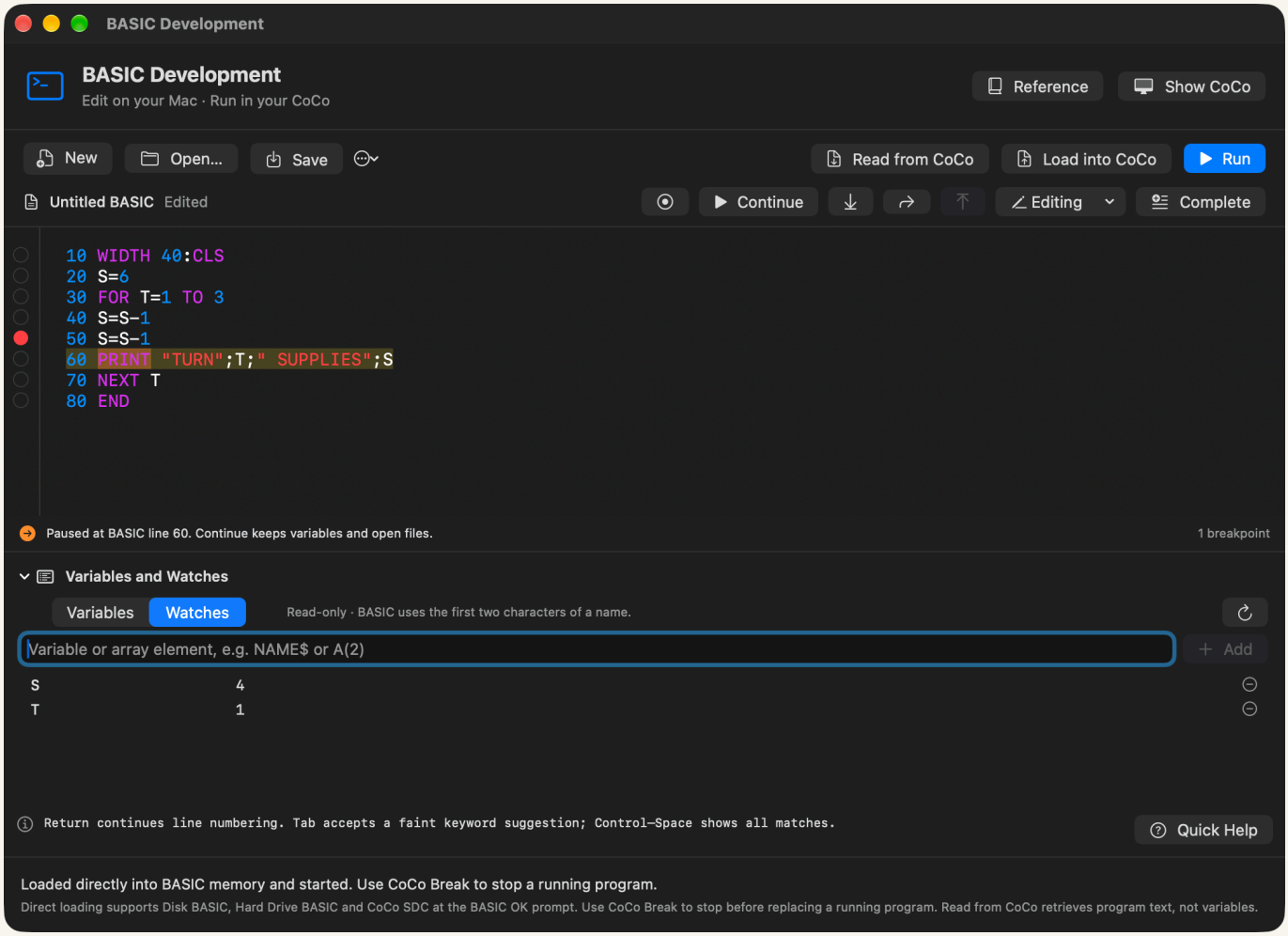


Figure 2: One step later, line 60 is highlighted. S has fallen to 4 while T is still 1.

The highlighted instruction is the next one BASIC will carry out. That small detail makes the screen and the watches agree: at the second stop S is 4, but the print instruction that will display it is still ahead.

Choose **Continue**. Line 60 prints the first result, NEXT moves T on, and the breakpoint catches the next trip through line 50. Choose Continue twice more to finish the three turns. You can use the **Variables** tab to inspect everything BASIC has created. A watch lets you pick a few names and keep your attention on them.

Repair the listing

Finish the run before editing. Click the breakpoint circle again to remove it. In your editable example, delete the entire line `50 S=S-1`. Keep line 40. Run the edited copy. You should see 5, 4 and 3. RATIONS.bas is the supplied repaired version if you want to compare. The gap between 40 and 60 costs the CoCo nothing.

Three kinds of stop. A breakpoint belongs to this editor document. **Step Into** carries out one BASIC statement. **Continue** runs until another stop or the end. BASIC also has a STOP statement and a CONT command of its own. A new run or a changed program can make guest continuation impossible. When BASIC reports a short error code and line number, use **Show Line** in the editor to inspect that statement and the error guide to look up the code. TRON and TROFF are the older guest tools for printing a trace of line numbers as BASIC runs.

When a step waits for you

A BASIC statement that asks for input can still be running while you look at the editor. **Pause BASIC step** appears during a pending step. A mid-statement pause can have no highlighted source statement because the

interpreter has yet to reach the next one. Choose **Continue** and focus the CoCo display to answer the waiting program. The debugger can step from a statement boundary again when it reaches a BASIC breakpoint.

A runtime error leads somewhere different. In the error banner, **Show Line** takes you to the numbered statement. The question-mark button, **Explain this BASIC error**, opens its entry in the local Quick Reference. The failed run has ended. Correct the source and run again when you are ready. To remove the document's stops together, use **Editing → Clear All Breakpoints**.

Move a line and keep its address

Line numbers are also destinations. In HELM, the bad-reply path jumps to 40 for another question. If you changed that label by hand and forgot the jump, the next bad reply would go looking for the old number.

For a small experiment, open HELM.bas again and run it. Enter **2** to reach the guest prompt. Type `RENUM 100,0,10`, then `LIST`. You asked BASIC to start the new numbers at 100, begin with the old lines from 0 onward, and leave a gap of ten. The old line 80 now reads **170 GOTO 130**. The question has moved to 130, and the jump followed it. Enter `RUN` and **1** to try the renumbered menu. Look for **BOUND FOR EARTH**.

That change happened in guest RAM. To bring it back to the Mac, use **Read from CoCo**. To renumber an *editor copy* instead, use **Editing → Renumber…**, choose the first number and increment, and apply the change. **⌘Z** undoes that edit before transfer. This distinction lets you try out new numbering on one side while keeping a saved copy on the other.

Give a chart its names

Chapter 5 of 12

What could go wrong if the computer drew Earth at the right place but printed “Mars” underneath? A chart needs more than a bag of names and numbers. It needs a way to keep the pieces of each planet together.

Make two cards for your chart

Open CHART.bas. Begin at the bottom of the listing, where its cartographer left this row of data.

```
150 DATA EARTH,64,80,MARS,176,100
```

Read three items at a time: a name, a distance across the picture, and a distance down. Make two little cards on paper.

Card number	Planet name	X, across	Y, down
1	EARTH	64	80
2	MARS	176	100

Which planet will appear farther to the right? Which will appear lower? In this graphics mode X grows as you move right, and Y grows as you move down. Mars has the larger number in both columns.

Deal the cards into RAM

```
20 DIM P$(2),X(2),Y(2)
30 FOR I=1 TO 2:READ P$(I),X(I),Y(I):NEXT I
```

`DIM` makes room for values you can look up by number. Each group is an **array**. `P$(1)` means the name for card 1, while `P$(2)` means the name for card 2. `X(2)` and `Y(2)` belong to that second card, too. You can think of the number in parentheses as a drawer label. BASIC calls it a *subscript*. A dimension of 2 makes positions 0, 1 and 2 available. This program uses positions 1 and 2 to match the numbers on the menu.

On line 30, READ asks for the next three items in DATA order. On the first pass through the loop, I is 1; on the second pass I is 2. READ remembers how far it got through the DATA list. So `X(2)` receives 176, even though the DATA line is more than a hundred line numbers away. BASIC only needs the list and a place to put each item.

Earth’s name is a string, so the dollar sign belongs on the name array. The X and Y arrays hold numbers. That lets the drawing code do arithmetic with the coordinates. For example, line 75 divides a pixel coordinate by eight and uses INT to find a whole text-cell position for a label.

Check the real chart against your cards

Select **RGB** under **Video Signal** for the colors used in this book. Choose the editor’s **Run**. Find the labels. Compare their positions with the paper cards. Press **A** on the CoCo keyboard. Now the text screen tells you the two marker values that the program read back from the picture.

At `OK`, enter `PRINT P$(2);X(2);Y(2)`. You should get **MARS 176 100**. You just asked BASIC for all three entries belonging to one card, after the drawing vanished. Try `PRINT LEN(P$(1));LEFT$(P$(2),1)` and predict both results. Earth has five letters, and Mars begins with M, so the CoCo prints **5 M**.

What if the next crew wants to read the original DATA items again? Type these three commands at the prompt, one per Return.

```
RESTORE
READ N$,E,V
PRINT N$;E;V
```

This time the result is **EARTH 64 80**. RESTORE moved the DATA reader back to the beginning. The new READ put the first three items into three new variables. The older arrays still have their own values. When you want to draw with those coordinates yourself, open Draw a planet you can find again.

Give a routine a job

Chapter 6 of 12

Think of the last thing a pilot ought to do after landing. The next crew needs a record of the trip. What if the flight program could simply ask someone to file that record and come back when the job was done?

Leave a place to come back to

Open VOYAGE.bas. First read this little piece of the main flight plan.

```
135 GOSUB 300
140 PLAY "T180O3L8CEG"
150 GOSUB 200
160 PRINT "LOG SAVED FOR ";P$(D)
170 END
```

A **subroutine** is a group of instructions with a job of its own. `GOSUB 300` sends BASIC to the landing routine. BASIC remembers *where to come back*: when the landing routine reaches RETURN, the next instruction after the call is the PLAY at line 140. The next call, `GOSUB 200`, sends BASIC to the filing routine. Its RETURN brings us back to line 160.

Turn the two calls into words and you can read the whole flight plan: land; play a tune; file the log; tell the pilot it was saved. The filing routine happens to be earlier than the landing routine in the listing. You can read them in the order that the pilot uses them. Line 170 ends the main plan, so BASIC leaves the code below it alone until a GOSUB asks for a job.

```
200 OPEN "O",#1,"VOYAGE.DAT:0"
210 WRITE #1,P$(D),S
220 CLOSE #1
230 RETURN
```

The file number `#1` connects the open, write and close instructions. Both the main program and the routine know the destination number D, the name array P$, and the supplies S. A GOSUB in this BASIC shares its caller's variables. That makes a short routine easy to call. It also means changing S inside the routine would change what the crew sees afterward.

Step into the file room

Use **Disk BASIC** with a disposable initialized floppy mounted in **FD0**. In **Media → Floppy Disks…**, check the name of that disk and look for **Saves automatically**. VOYAGE writes a file called **VOYAGE.DAT** and replaces an earlier flight under that name. If you don't have a practice disk yet, make one before running this step.

1. In the VOYAGE editor, mark line **150** as the only breakpoint. Add **D** and **S** to Watches. Choose **Run**.
2. Choose **2** and press Return at the helm. When the Mars chart asks you to land, press **A** on the CoCo keyboard. Look at the editor when it stops on line 150. D is **2**, and S is **3**.
3. Guess which line **Step Into BASIC statement** will highlight. Try it. The highlight moves to **200**, the first instruction of the file job. The crew hasn't announced a save yet.
4. Choose **Step Out of BASIC subroutine**. The CoCo finishes the job, including CLOSE. It stops on line **160**, where the main flight plan resumes. Choose **Continue** and look for **LOG SAVED FOR MARS**.

Choose how much of a job to watch

Leave the breakpoint at 150. Fly again, this time entering **1** for Earth and pressing **A** to land. At the stop, choose **Step Over BASIC subroutine**. Predict the next highlighted line. You land on **160** again, this time without a

pause at 200. The filing job still ran. Choose **Continue** and you see **LOG SAVED FOR EARTH**. Remove the breakpoint when you finish this exercise.

Use **Step Into** to see how a job works, **Step Over** when you already trust the job, and **Step Out** when you've seen enough of the current routine. A breakpoint you deliberately set *inside* a routine can still bring Step Over to a stop there. A GOTO, unlike GOSUB, makes no promise to come back.

As your programs grow, **ON ... GOSUB** can use a number to select one of several routines. **DEF FN** gives a name to a calculation that returns a value; BASIC's own LEN and INT are familiar examples of how it feels to use a function. Look up these forms in the language reference. The flight program already has two useful jobs, and you can follow both of them all the way back to the cockpit.

Design a flight deck

Chapter 7 of 12

You have one screen and two jobs for it. At takeoff the pilot needs a short question and two big choices. After landing the crew wants to compare a name at the left with a number farther across the page. Should both screens have the same number of columns?

Walk across three screens

Open CONSOLE.bas in BASIC Development and click **Run**. Click the CoCo display. Start with your hands off the keys. Count how many times the *same* Mac window changes before you get back to the helm.

1. The first screen says **32 COLUMNS**. Tap and release A.
2. The second says **40 COLUMNS**. Look for a star almost at the right edge. Tap A again.
3. The third says **80 COLUMNS**. Find **SUPPLIES LEFT** and the number below it. Tap A once more. **BACK AT THE HELM** puts you back in 40 columns.

The 32-column screen will look familiar to someone who used an early CoCo. The CoCo 3 adds 40- and 80-column text screens. BASIC changes how many character cells fit across the picture. Dragging the corner of the Mac window scales the picture you already have. Only one of those actions gives your program a column 48 to write in.

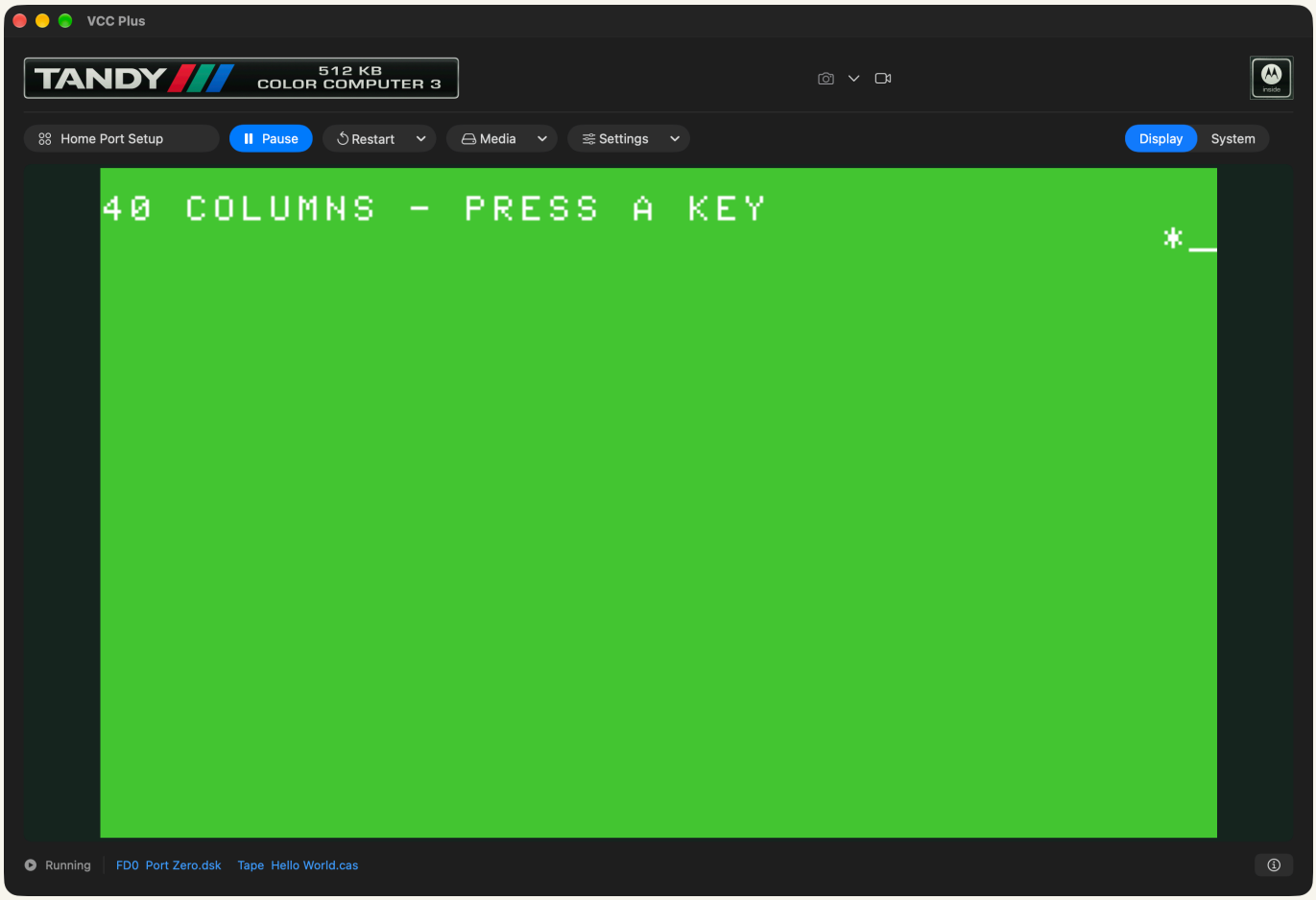


Figure 3: CONSOLE's actual 40-column screen places a star one cell in from the right edge. The Mac window also shows the camera, shared capture options and video controls.

Put one star where you want it

```
30 WIDTH 40:CLS:ATTR 3,0
40 LOCATE 0,1:PRINT "40 COLUMNS — PRESS A KEY";
50 LOCATE 38,2:PRINT "*";
60 IF INKEY$="" THEN 60
```

Think of a sheet of graph paper. CoCo 3 text coordinates count columns from the left and rows from the top. The first cell is column zero, row zero. In a 40-column row, the last column is 39. That puts the star at column 38 **one cell in from the right**. Row 2 is the third row down.

Read line 30 left to right. Select 40 columns, clear the screen, then choose the foreground and background attributes for the text to come. Line 40 moves the cursor and prints the instruction. The semicolon after the quoted text leaves the cursor on the same row. The colon between two instructions lets BASIC do them in order on a single numbered line.

Why does the star stay put? At line 60 the program checks for a keystroke over and over. It can spend a long time on one line without drawing another star. You decide when it moves on.

Make the cargo count easy to find

```
70 WIDTH 80:CLS
80 LOCATE 0,1:PRINT "80 COLUMNS";
90 LOCATE 48,1:PRINT "SUPPLIES LEFT";
100 LOCATE 0,2:PRINT "PRESS A KEY";
110 LOCATE 48,2:PRINT USING "##";6
120 IF INKEY$="" THEN 120
130 ATTR 0,1:WIDTH 40:CLS:PRINT "BACK AT THE HELM"
```

Follow the two instructions that start at column 48. The heading lands there on row 1. The number lands underneath on row 2. `PRINT USING` supplies a little printing pattern: the two number signs set aside two character places for the number six. The unused place at the left gives the digit some breathing room. Labels and numbers can line up even when a name on the left has a different length.

The actual flight reader, LOGBOOK.bas, uses the same two columns. It gets its values from a file. We will make that file in Keep a log past bedtime. For now CONSOLE gives you a rehearsal with a number already aboard.

Design decision. Would you put a two-choice landing question in 80 columns? You could, but the pilot would be reading smaller letters to make the same choice. Use the 40-column deck for the question and save the extra room for the report.

Draw a planet you can find again

Chapter 8 of 12

Somewhere on the screen is Earth. You can point to it. How could a program find it again after it has drawn it? Give Earth an address on a grid and a color number you can read back.

Mark two landing sites

Choose **RGB** under **Settings → Video Signal**. Run CHART.bas. Leave it waiting for a key. Look for a blue ring labeled EARTH and a red ring labeled MARS. Each ring has a tiny dot in the center. Press A after you have found both dots.

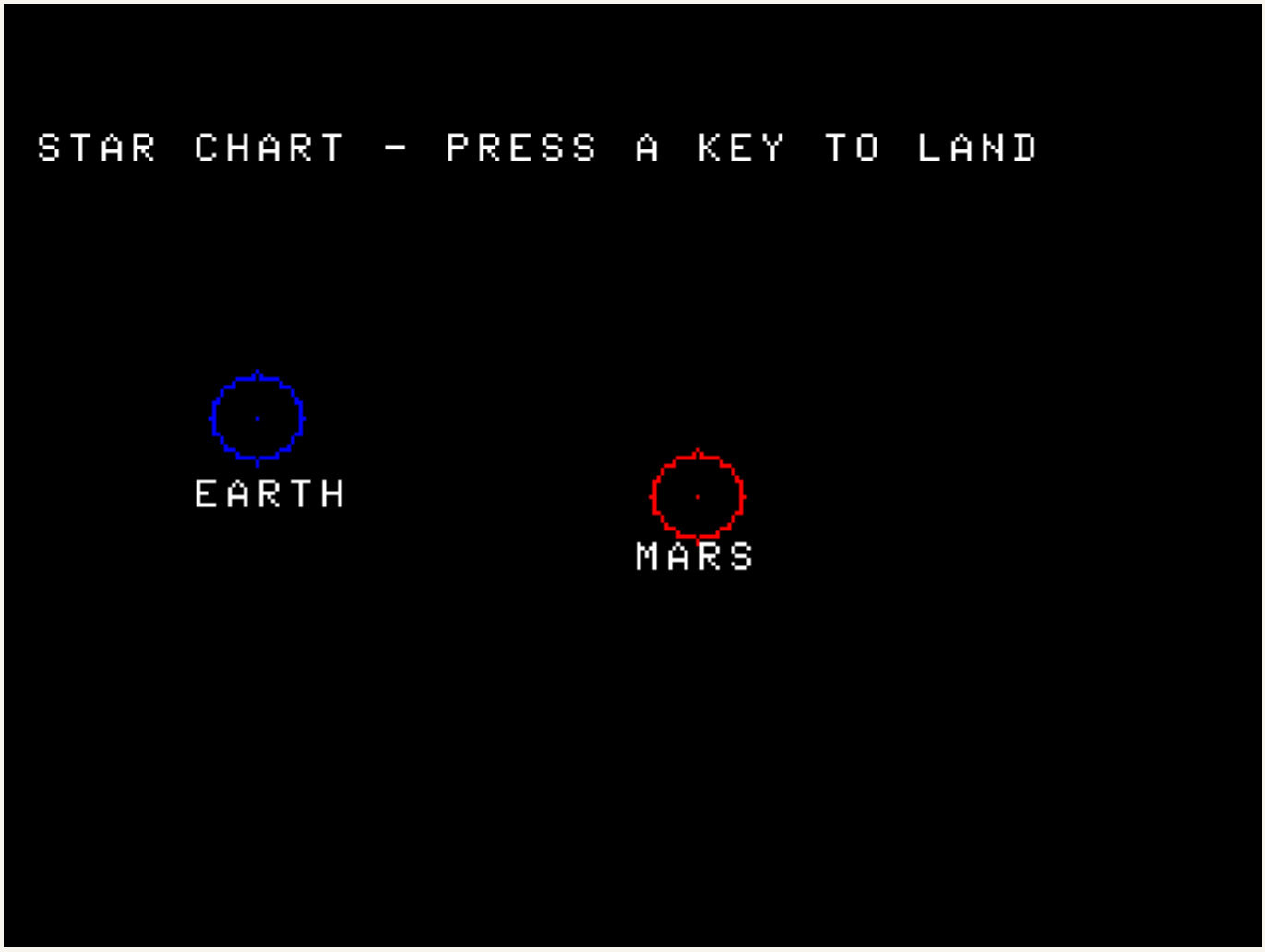


Figure 4: The guest star chart labels a blue circle Earth and a red circle Mars. Each circle has a center marker.

The program reports **EARTH MARKER 1** and **MARS MARKER 2**. Those numbers came from the dots, not the names. You could tell the planets apart from the labels even when the two colors were hard to distinguish.

```
40 HSCREEN 2:HCLS 0
42 PALETTE 0,0:PALETTE 1,9:PALETTE 2,36:PALETTE 3,63
45 HCOLOR 3:HPRINT(1,1),"STAR CHART - PRESS A KEY TO LAND"
50 FOR I=1 TO 2
60 HCIRCLE(X(I),Y(I)),12,I
```

```
70 HSET(X(I),Y(I),I)
75 HPRINT(INT(X(I)/8)-2,INT(Y(I)/8)+2),P$(I)
80 NEXT I
90 A=HPOINT(X(1),Y(1)):B=HPOINT(X(2),Y(2))
```

`HSCREEN 2` opens a CoCo 3 graphics screen. `HCLS 0` fills it with color slot zero. Think of a palette as numbered pots of paint. Line 42 puts black in pot 0, blue in 1, red in 2 and white in 3 for our RGB display. The number stored at Earth's center is **1**, even though the RGB color chosen for pot 1 is **9**. This distinction explains the report at the end.

`HCIRCLE` draws the rim. `HSET` colors the pixel at the center. `HPOINT` asks the screen which pot that pixel uses. The coordinate pairs came from the chart data. `HPRINT` places whole letters on its own grid: line 75 divides each pixel coordinate by eight, drops the fraction, and shifts the label underneath the ring. A letter covers a little block of pixels.

Keep a second picture ready

Earlier BASIC programs could draw on a coarse **SET** grid or use **PMODE** graphics. A PMODE program can keep more than one picture in RAM. Book 2's stripe experiment shows what one of those pictures looks like through composite and RGB. Here we will change which picture the monitor sees.

Run PAGES.bas in **Disk BASIC**. It first selects the 32-column screen so the following legacy graphics commands start in the intended display mode. Wait for the square at the left. Tap A once and find a smaller square on the right. Tap it again and the first square returns. A third tap brings you back to 40-column text.

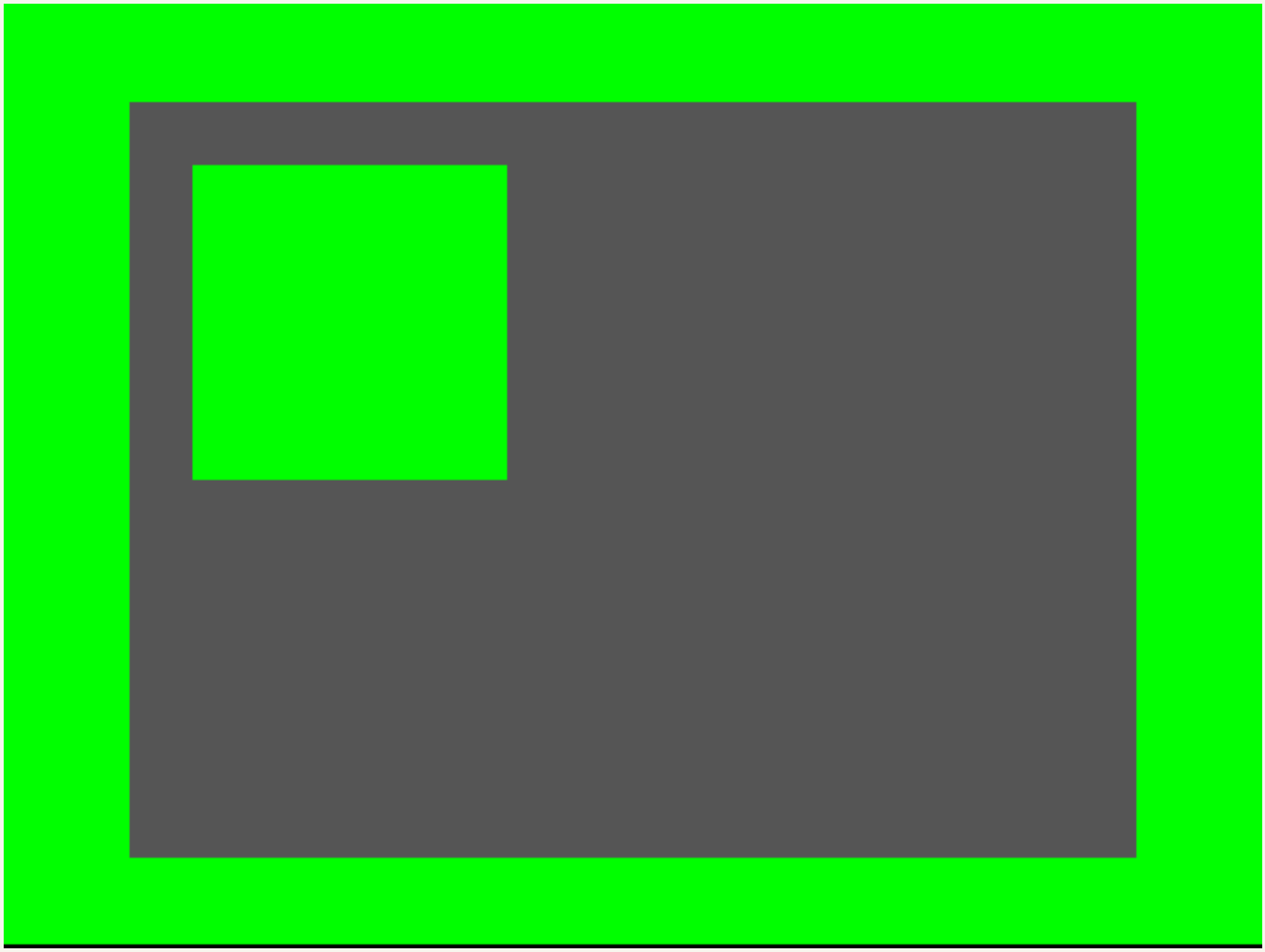


Figure 5: The first PMODE picture shows a filled square at the left. The other picture has already been drawn in memory.

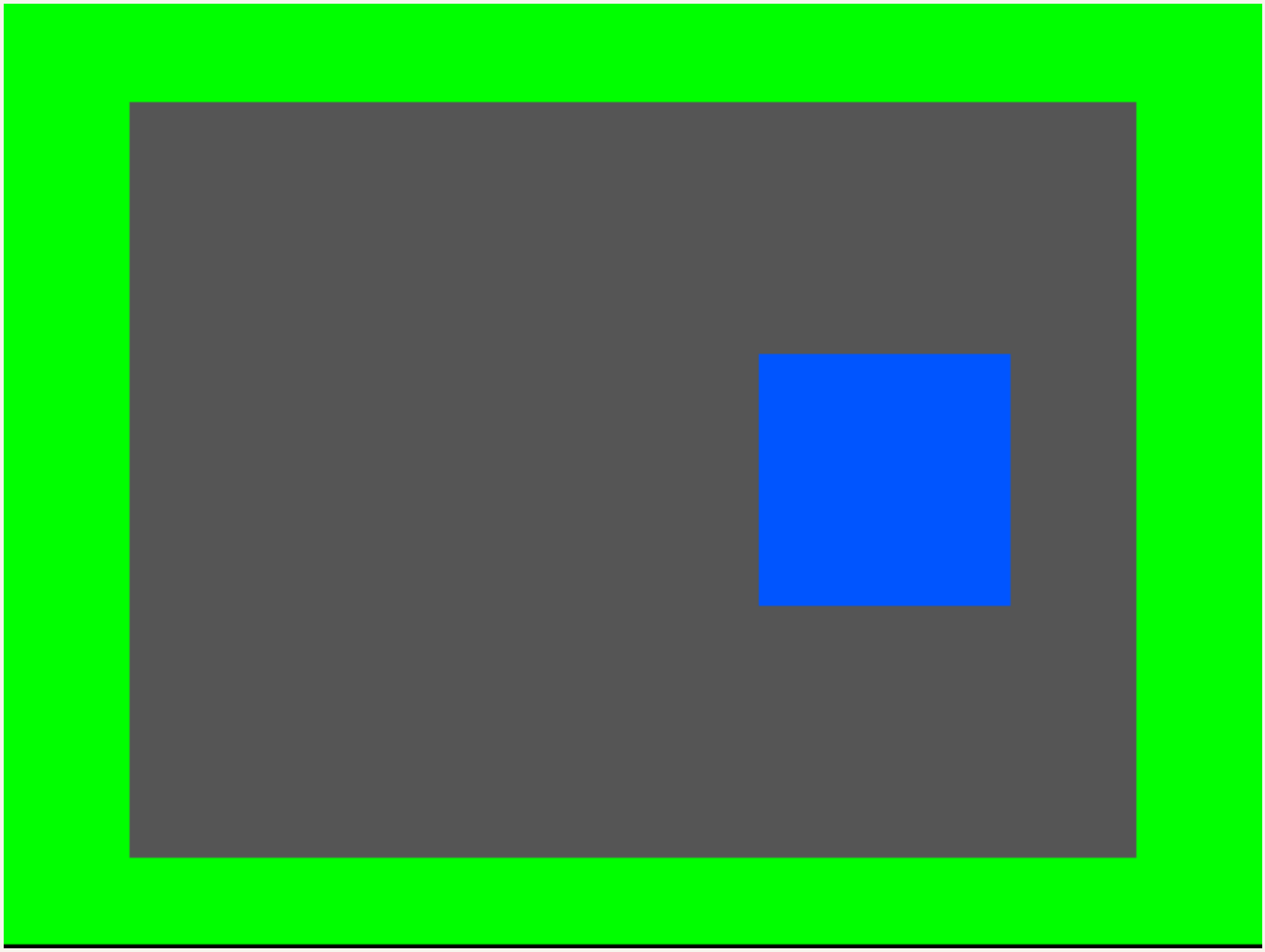


Figure 6: After the first keypress a smaller square appears on the right. The previous square is still on the other page.

```
5 WIDTH 32:CLS
10 PMODE 1,1:PCLS 0:SCREEN 1,0
20 COLOR 1,0:LINE(16,16)-(95,95),PSET,BF
30 PMODE 1,3:PCLS 0:COLOR 3,0
40 LINE(160,64)-(223,127),PSET,BF
50 IF INKEY$="" THEN 50
60 SCREEN 1,0
70 IF INKEY$="" THEN 70
80 PMODE 1,1:SCREEN 1,0
90 IF INKEY$="" THEN 90
```

Look closely at lines 30 and 40. BASIC has switched its **drawing** to another part of RAM before that first keypress. The monitor keeps showing the first square while the second is being drawn. Line 60 is the handoff: `SCREEN` shows the page BASIC has selected. Line 80 selects and shows the first one again, without drawing its square a second time. This is one way an animation can have its next picture ready in time.

Look behind the visible picture

For a closer look, put a BASIC breakpoint on line **50** of PAGES and run it again. At the stop, leave the CoCo paused. Open the **Debugger** and choose **Graphics Preview**.

1. Choose **Current Video Page**. The left square is on display.

2. Choose **At Address**, **Mapped CPU**, enter **1A00** in the hexadecimal address field and choose **Go**. Select **PMODE 1**, **Set 0**. The square on the right is already there.
3. In **Pixel Inspector**, look at X **81**, Y **32**. The byte is **$AA** at **$1E14**. You can leave its value alone. This is one byte helping hold the second picture before anyone can see it on the CoCo monitor.

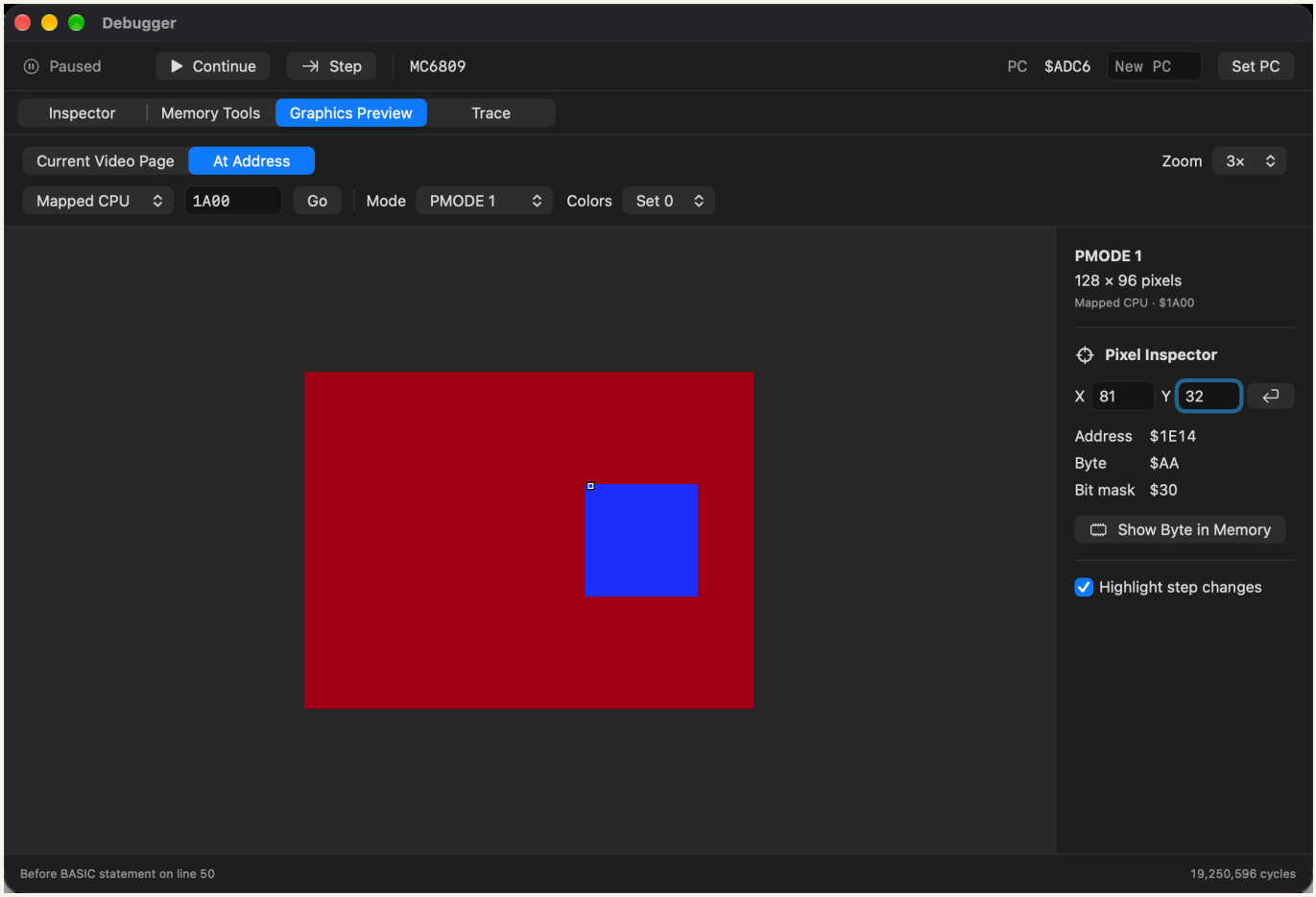


Figure 7: The native Graphics Preview inspects the right-hand square at mapped address 1A00. Pixel 81,32 uses two bits of the byte at 1E14.

Why page starts 1 and 3? One PMODE 1 picture needs two of BASIC's graphics allocation pages. The first uses pages 1 and 2; the second starts at 3. The address above applies to the Disk BASIC setup in this exercise. The preview shows its 128 by 96 stored pixels. BASIC's drawing coordinates cover a finer grid, so the preview pixel numbers differ from the coordinates in the LINE statement.

Remove the line 50 breakpoint in BASIC Development, choose **Continue**, and use the three keypresses to finish the trip. You have seen two jobs that a single block of memory can do in succession: hold a drawing under construction, then supply the image on the monitor.

Listen for a signal

Chapter 9 of 12

A joystick can report more than “left” or “right.” It can tell your program how far the stick has moved. Give the ship three messages to send, and see how one number decides which one you receive.

Point the right-port scanner

Open **Input → Keyboard and Joysticks…**. Make a note of your current **right CoCo port** assignment so you can put it back. Either select your controller for that port, or select **Keyboard** and the arrow-key layout. If the options are folded, expand the right port’s options. With the keyboard layout, Left and Right move the virtual stick and releasing the keys centers it.

Open RADAR.bas and click **Run**. Focus the CoCo display. Hold the stick toward the left, or hold Left on your keyboard, and tap **A** on the CoCo keyboard. You should get **WEST SIGNAL**. Start another run, release the direction and tap A for **CENTER SIGNAL**. On a third run, hold right and tap A for **EAST SIGNAL**. A is an ordinary typing key in the arrow layout. Choosing the W/A/S/D layout would give that key a joystick job of its own.

```
10 WIDTH 40:CLS
20 PRINT "RIGHT STICK SCANNER"
30 PRINT "MOVE THE STICK. PRESS A KEY TO SCAN."
40 IF INKEY$="" THEN 40
50 X=JOYSTK(0)
60 IF X<24 THEN PRINT "WEST SIGNAL":END
70 IF X>39 THEN PRINT "EAST SIGNAL":END
80 PRINT "CENTER SIGNAL"
```

The pause on line 40 gives you time to aim. `INKEY$` returns an empty string while there is no character waiting. When it receives a character, execution reaches line 50. `JOYSTK(0)` then samples the right port’s horizontal axis. The 0 inside the parentheses selects the axis. The number assigned to X describes its position.

Decide where west begins

The stick reports a value from 0 through 63. The middle is around 31. The program gives the center a generous landing strip.

X reading	Line that decides	Message
0 through 23	60	WEST SIGNAL
24 through 39	80	CENTER SIGNAL
40 through 63	70	EAST SIGNAL

Prediction. Suppose X is exactly 39. Line 60 asks whether 39 is smaller than 24. Line 70 asks whether it is larger than 39. Both answers are false, so BASIC reaches line 80. At 40, line 70 prints its message and executes END. The center message never gets its turn.

If all three runs return CENTER, check the right-port assignment and the selected stick. If the prompt keeps asking for a key, click the CoCo display and tap A again. Restore your previous port assignment when you finish.

Give the pilot an arrival sound

The flight in VOYAGE.bas plays three notes after the landing key and before filing the log.

```
140 PLAY "T180O3L8CEG"
150 GOSUB 200
160 PRINT "LOG SAVED FOR ";P$(D)
```

Read the short score inside the quotes. T sets the tempo to 180; O picks octave 3; L8 makes the following notes eighth notes. C, E and G are the three pitches in order. PLAY follows the score just as BASIC follows a sequence of program lines. SOUND, another command in the reference, takes a pitch number and a duration for a single tone.

During the final flight, listen when you press the landing key. The words **LOG SAVED FOR MARS** or **LOG SAVED FOR EARTH** also mark the arrival. If the Mac is muted, those words remain your check that the ship landed and its log was closed.

Keep a log past bedtime

Chapter 10 of 12

The pilot knows where the ship landed. So does the variable in RAM. Tomorrow's pilot gets a freshly started computer. How will that pilot find out where the ship went?

Give the ship its own disk

Use the **Disk BASIC** profile for this lesson. Save anything you want to keep before changing profiles or replacing the program in RAM. Open **Media → Floppy Disks…** and put a disposable, initialized floppy image in **FD0**. You can use **New Disk** in the drive actions or duplicate an initialized practice image in Finder. Keep the original somewhere else. Name the Mac working copy so that you will recognize it tomorrow.

Check the FD0 row. You want your working filename, write protection off, and **Saves automatically**. If the disk exists only in memory, use **Save As…** to give it a Mac file. If the machine is paused after the change, resume it. Back at the CoCo shows how to tell a Mac image from the CoCo files inside it.

What you hold	Where it lives	What it contains
VOYAGE.bas	A Mac source file	Instructions you can read and edit
Your working .dsk	A file in Finder, mounted in FD0	A whole emulated floppy, with room for many files
VOYAGE.DAT	A CoCo file inside that floppy	The destination and remaining supplies from the latest flight

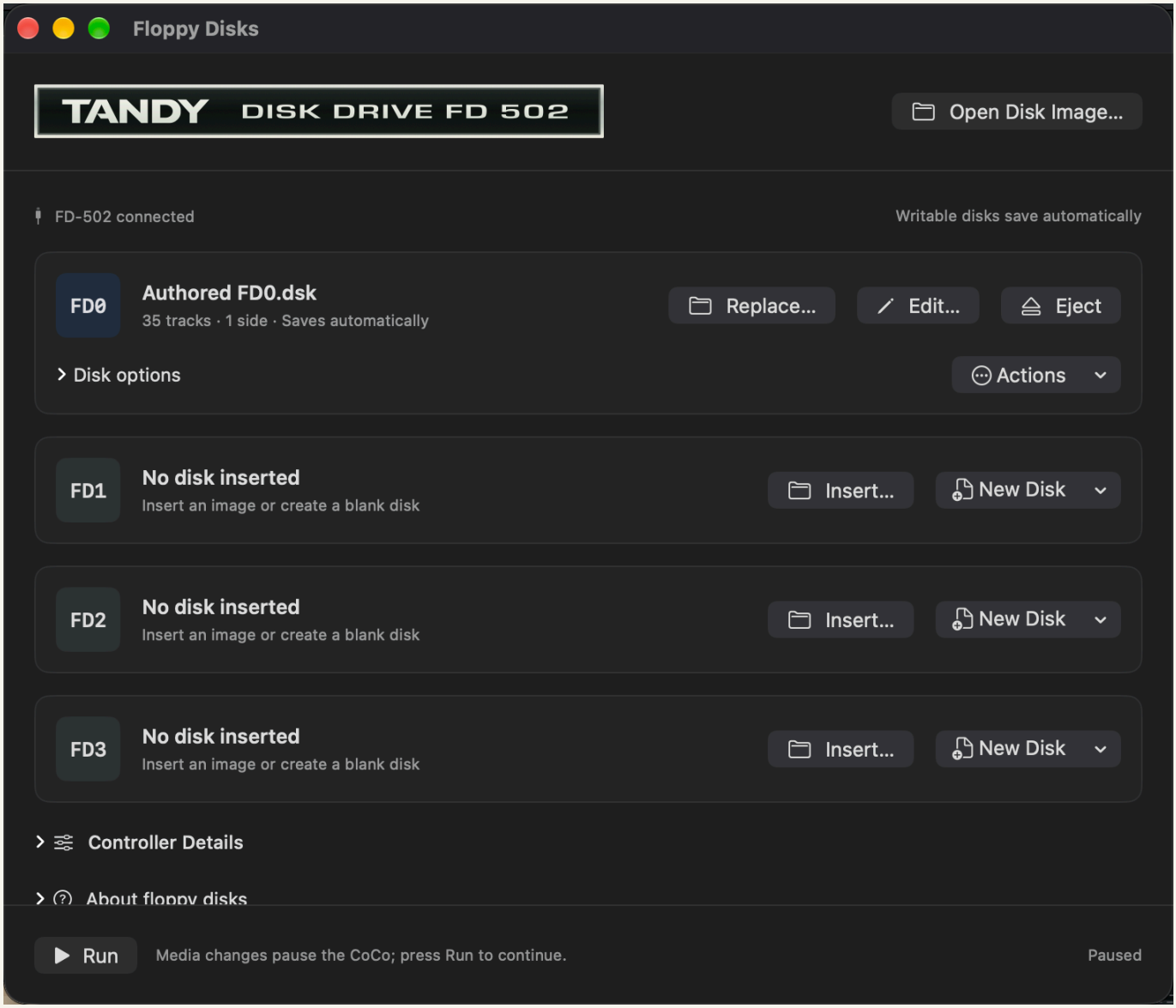


Figure 8: The current FD0 row identifies the mounted image and its save mode. This representative authored disk says Saves automatically; your own window should name the working voyage disk you chose.

Write a voyage

Open VOYAGE.bas in BASIC Development and run it. Click the CoCo display. Enter **1** and press Return for Earth. When the chart says **PRESS A KEY TO LAND**, tap A. Wait for **LOG SAVED FOR EARTH** and the BASIC prompt. Here is what happened after the landing jingle.

```
200 OPEN "O",#1,"VOYAGE.DAT:0"
210 WRITE #1,P$(D),S
220 CLOSE #1
230 RETURN
```

Line 200 opens a door between BASIC and a file. The O stands for output. **VOYAGE.DAT** is the name on the CoCo floppy, and **:0** selects drive zero. **#1** is a file number that this program can use while that door is open. The file number and drive number have different jobs.

Line 210 writes two items in order: a destination string and the number of supplies. WRITE includes the separators that another BASIC program will need to read the values. Line 220 closes the file and lets Disk BASIC finish its bookkeeping. Line 230 returns to the caller. Only after all that does the ship print its success message. Opening this file for output again replaces the previous voyage; keep this experiment on its own disk.

Hand the log to a second program

Leave FD0 in place. Open LOGBOOK.bas in BASIC Development and run it. Where did the word EARTH come from? Look at the entire reader. You will find no planet names in its source.

```
10 WIDTH 80:CLS
20 OPEN "I",#1,"VOYAGE.DAT:0"
30 INPUT #1,P$,S
40 CLOSE #1
50 LOCATE 0,1:PRINT "FLIGHT LOG";
60 LOCATE 48,1:PRINT "SUPPLIES LEFT";
70 LOCATE 0,2:PRINT P$;
80 LOCATE 48,2:PRINT S
90 END
```

The I on line 20 stands for input. Line 30 asks for the same two items in the same order in which the writer stored them. The dollar sign on P$ tells BASIC to receive a string. S receives a number. Then the reader closes the file and lays out its report. On this run the result is **EARTH** with **4** supplies left.

Try a missing log. On a different, empty initialized disk, LOGBOOK reports **?NE ERROR IN 20**. It reached the command to open the input file, but that disk has no file by the requested name. Put your voyage disk back in FD0 before running the reader again. If the BASIC workspace stopped at the error, choose **Continue** to return to the prompt first. Keep the disk with the good log; you do not need to create a blank VOYAGE.DAT to fix this.

Open locker two before locker one

A flight log can be read from the beginning. What if your ship has numbered lockers and you need the contents of locker two right now? Try LOCKERS.bas on the same disposable disk. It creates or updates a separate file called **LOCKERS.DAT**. The output names Mars first, then Earth.

```
10 WIDTH 40:CLS
20 OPEN "D",#1,"LOCKERS.DAT:0",8
30 FIELD #1,8 AS N$
40 LSET N$="EARTH":PUT #1,1
50 LSET N$="MARS":PUT #1,2
60 GET #1,2:PRINT "LOCKER TWO: ";N$
70 GET #1,1:PRINT "LOCKER ONE: ";N$
80 CLOSE #1
```

D opens a **direct-access** file. Each record has eight character places. FIELD gives those places a name in the working buffer. LSET puts a name at the left of that space. PUT sends the buffer to a particular numbered record on disk. Line 50 writes Mars to record 2. Line 60 fetches that record immediately, and line 70 goes back to fetch record 1. Disk BASIC can do those reads in the order you request.

Imagine needing 80 lockers. The fixed record length would let the system find the start of locker 80 without reading the text in the first 79. Compare that with the tiny shared log: its two items always travel together, so sequential input is enough for the job.

Read a magazine’s address book program

Bob Devries used random-access records for names and addresses in *Mini Data Base*. Open **Help → Archive Library…**, browse **Magazines → Coco Nutz! Newsletter → 2005**, and open the January issue. The article and listing begin on PDF page 10. Enter 13 in the reader’s page field and press **Return**. This page ends *Mini Data Base* and starts Richard Kelly’s *Pill*.

Above the scan, choose **Examples on this page → Mini Data Base → Open in BASIC Editor**. You now have an editable copy of the complete numbered program from the publisher’s companion ZIP. Find line 1070. Its `FIELD` statement divides a 100-character record among the parts of an address, where LOCKERS gave all eight characters to a single name. Opening the example leaves the program in the CoCo alone. **Copy Example** in the same menu puts the listing on the Mac clipboard.

Take the voyage to another machine

Chapter 11 of 12

A friend at a BASIC prompt cannot tell the CoCo to open a Mac editor window. Give the friend a disk with a real BASIC program inside, and an instruction they can type. This chapter also makes a tape for the trip. Start in **Disk BASIC** with your work saved.

Put VOYAGE on a disk

1. In **Develop → BASIC Development...**, open VOYAGE.bas. Beside Save, open **More file actions** (the button with three dots) and choose **Export Disk...**.
2. Use **VOYAGE** for the CoCo program name. For the first disk, leave **Install DOS bootloader** clear. Save the Mac image as **VOYAGE.dsk** in a practice folder. This operation makes a disk carrying your program. It does not insert the disk for you.
3. In **Media → Floppy Disks...**, insert that new image in FD0, or replace your practice disk with it. This is a fresh *working* export. Enable writing if you want to land and keep a log on this disk. Check for **Saves automatically**. If paused, resume the machine.
4. At the CoCo prompt type each of the following on its own line. Press Return after each command.

```
DIR
LOAD"VOYAGE"
RUN
```

The directory shows VOYAGE as a BASIC program. LOAD copies it into RAM. RUN reaches **FLIGHT DECK**. Choose 2, press Return, then tap A on the landing chart. Wait for **LOG SAVED FOR MARS**. Now the exported disk has a program *and* the data file that the trip just wrote.

A shorter invitation. Export a second disk from the same source and use the same CoCo name, **VOYAGE**. This time check **Install DOS bootloader** and give the Mac image a distinct name such as **VOYAGE-BOOT.dsk**. Insert it in FD0 in Disk BASIC. At the prompt type `DOS` and press Return. The loader on this particular disk starts VOYAGE and takes you to the same choice of planets. The full route, `LOAD"VOYAGE"` followed by `RUN`, also names the program you want. A plain Disk BASIC data disk needs no startup loader.

Send the program by cassette

Return to the VOYAGE source in the editor. Choose **More file actions → Export Cassette...**, set the CoCo name to **VOYAGE**, and save the Mac file as **VOYAGE.cas**. Keep a writable **practice floppy** in FD0: the program travels on tape, but its own OPEN command still sends the flight log to a disk.

Open the new cassette image in **Media → Cassette...**, rewind to the beginning, and choose **Play**. At a running BASIC prompt, type the first command below and press Return. Wait for the program to load and BASIC to return to its prompt before typing the second.

```
CLOAD"VOYAGE"
RUN
```

The recorder advances while BASIC runs its motor. The game reaches **FLIGHT DECK** from tape. Choose Mars and land to write the log on your practice floppy. If you try again, rewind the tape first. You can open LOGBOOK from the editor to read the result from FD0. A cassette holds its entries in sequence, so this time there is no floppy-style directory step.

Know which drive answered

The first two recipes used Disk BASIC and an ordinary floppy. Keep that setup if you are following the book straight through. The other BASIC profiles can run VOYAGE, too. Use disposable copies for these optional trips and keep your current Mac source saved before switching machines.

Hard Drive BASIC. RGB-DOS normally directs a request for drive zero to a hard-disk volume. Attach a writable copy of the exported floppy in **FD0**. At the prompt enter the following lines separately.

```
DRIVE OFF
DIR 0
LOAD"VOYAGE:0"
RUN
```

DRIVE OFF exposes the floppy for this route. Make sure DIR lists VOYAGE before running it. Leave floppy routing in effect through the landing **and** any LOGBOOK read, since their data file also ends in :0. When finished, use `DRIVE ON` and then `DIR 0` to see your hard-disk volume again. Do this switch at the prompt after the file has closed.

CoCo SDC. Its SD card is a Mac folder containing disk images. Put a disposable **VOYAGE.DSK** in the root of a working card folder. In the SDC cartridge details you can show the folder in Finder or choose a different one. To land and write the flight log, the **working** folder and image need to be writable and **Protect SD card files (read-only)** must be off. Preserve the original export. At the SDC-DOS prompt enter each line separately.

```
DRIVE 0,"/VOYAGE.DSK"
DIR 0
LOAD"VOYAGE:0"
RUN
```

The slash begins at the top of the card folder. The first line mounts a whole image on SDC0. The third line asks BASIC for one program *inside* that image. Leave it mounted through the landing and LOGBOOK read. Restore your usual card, mount and protection afterward. Which drive zero? shows the controls and the starter mounts. Pack a note with the exported disk. A friend needs the intended BASIC setup, the filename *inside* the disk, the command to start it, and permission to use a working copy for the new log. Those four details tell them much more than a Finder filename alone.

Fly, land, power down, remember

Chapter 12 of 12

Today's crew gets six supplies and a question. Tomorrow's crew gets a disk. Your job is to leave enough information on that disk for someone who never saw your screen to say where the ship went.

Make the handoff

1. Start in **Disk BASIC** with a disposable, initialized writable disk in FD0. Use the Mac filename and the **Saves automatically** status to identify the disk. It is fine to use your own working VOYAGE export. Keep an original copy elsewhere.
2. Open VOYAGE.bas and run it. Click the display. Try entering X first. The crew should ask again. Choose **1**, press Return, and watch the turn counts before the Earth chart appears. Tap **A** to land. Wait for **LOG SAVED FOR EARTH**.
3. Open LOGBOOK.bas and run it. **EARTH** with **4** supplies makes a useful first receipt. On paper, predict the destination **and** count the next crew will read if you fly to Mars instead.
4. Run VOYAGE again. Enter **2** for Mars and press Return. When the landing chart appears, tap A. Wait for **LOG SAVED FOR MARS** and for the Mac disk save to finish.
5. Restart the CoCo from the emulator controls. Check FD0 afterward. If you set up a fresh machine for the second crew, mount that same working disk in FD0. The reader only needs to read this time; a protected copy works.
6. From BASIC Development, open LOGBOOK and choose its **Run** button to load that reader into the newly started machine. Compare the 80-column report with your prediction.

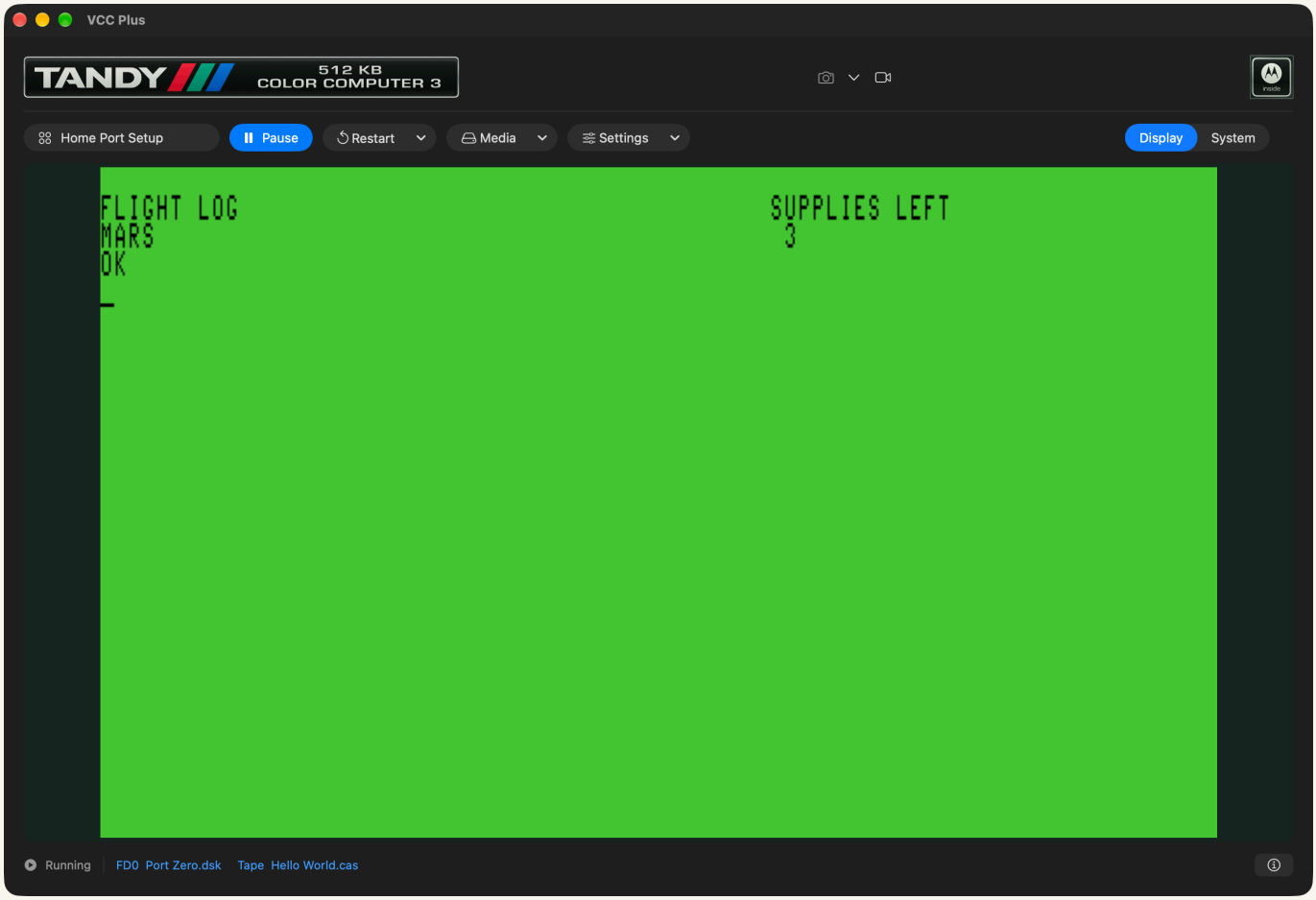


Figure 9: The flight log on the wide CoCo 3 text screen reads MARS with three supplies left.

Why three? In the flight listing, D is 1 for Earth and 2 for Mars. The loop runs through D+1 turns. Earth takes two supplies out of six. Mars takes three. The report has a destination name as well as a number, because a plain 3 could mean almost anything to a crew that missed the trip.

Reconstruct the last trip

Give the second crew the report and this set of questions. Let them try answering before opening the source.

- **Where did the pilot's keystroke go?** Line 40 of VOYAGE puts the reply in C$. Lines 50 and 60 choose a destination index. The pilot can make a mistake and get another turn.
- **What drew the landing site?** The data table connects that index to a name and two coordinates. The routine at line 300 uses them to draw and label the selected planet. Pressing a landing key releases the wait and the routine returns.
- **What survived the restart?** The machine cleared its working values. The disk file still held the name and supply count because the writer closed the file and VCC Plus retained the working image. LOGBOOK opened that file and read two values into its own variables.
- **Why is Earth missing?** The next flight opened a file of the same name for output and replaced its contents. The program promises a *last* voyage. A history of many voyages would need a different filing decision.

Recover a lost flight

What you see	First thing to check
An old planet appears on the report.	Compare the FD0 **Mac filename** with the copy used for the last landing. Check the route to drive zero if you changed BASIC profiles.
The program stops at the OPEN line with ?NE.	The reader looked for VOYAGE.DAT on the selected disk. Reattach the disk the writer used, or complete one flight on your working disk first.
The planet screen stays up.	Click the CoCo display and tap an ordinary character key. The program is waiting for permission to land.
The file never reaches the next crew's Mac image.	Check write protection, save mode and any pending save error. Let the guest close its file, then finish saving the working image before ejecting it.

For a short next project, show another person the left square in the hidden-page experiment. Ask them to guess whether the right square exists yet. You can now take them from a keypress to an instruction, from an instruction to RAM, and from RAM to the evidence on the screen. Start with whichever step makes them curious.

BASIC at your elbow

Appendix A

When you remember that `GET` can pick something up but have forgotten what goes in the parentheses, start here. The PMODE graphics form picks up a rectangle. The disk form reads a record through an open file. A program can use both.

Find a command

Browse by first letter below, or search this volume. In the app, **Read in BASIC Manual** in BASIC Quick Reference brings you straight to the entry for the selected command or error. Each name here also has its own link, so you can go straight to PRINT USING, SDC-DOS DRIVE or JOYSTK. For a message that begins with a question mark, look up its error code.

A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · R · S · T · U · V · W

How to read a form. Uppercase words are BASIC keywords. Replace names such as `filename` or `number` with your own value. Square brackets enclose an optional part. An ellipsis means the preceding item can repeat. The forms describe how to write a statement or expression. Files, devices and variables named in a form may need setting up first. When you want a complete exercise, follow its lesson or use the named example in **Develop → BASIC Quick Reference...**.

Which machine? "CoCo 3 BASIC" includes commands provided by the extended interpreter in this machine. "Disk BASIC" needs the disk interpreter. The Hard Drive BASIC and CoCo SDC profiles add their own forms of some familiar commands. Compare the three ways to reach a disk before using a command that changes its contents.

Work out an expression

BASIC can do several jobs inside one expression. In ORDER.bas, guess both answers before running lines 20 and 30. Both use a two, a three and a four. The parentheses change which operation happens first.

```
10 WIDTH 40:CLS
20 PRINT 2+3*4
30 PRINT (2+3)*4
40 PRINT 2^3
50 PRINT 6/2-1
60 PRINT 4>3;4=3;4<>3
70 PRINT "RED"+" DUST"
80 PRINT (4>3) AND (2=2)
90 PRINT NOT (4>3)
100 END
```

The first two answers are 14 and 20. BASIC normally multiplies before adding. Parentheses let you ask for the sum first. The caret raises a number to a power. Two cubed is 8. The slash divides. Six divided by two, minus one, is 2.

The comparison signs ask questions. = asks "the same?", <> asks "different?", and <, >, <= and >= compare order. Line 60 prints **−1, 0, −1** because the answers are yes, no and yes. In an IF test, zero is false and a nonzero result is true. With text on both sides, the plus sign joins the two strings. The space at the start of the second string puts a gap in **RED DUST**.

In line 80, the two comparisons each produce a true result. AND combines them and prints −1. In line 90, NOT reverses the result of the comparison and prints 0. AND, OR and NOT can also work on a number's individual bits. If you mean to join two questions, put parentheses around each one as line 80 does.

A

ABS · AND · AS · ASC · ATN · ATTR · AUDIO

ABS

Return the magnitude of a number.

```
ABS(number)
```

CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *Numbers and rounding*.

AND

Combine integer bit patterns; often used to join conditions.

```
expression AND expression
```

CoCo 3 BASIC · Choose a destination
See also OR, NOT, IF.

AS

Give a string variable a field within a direct-access file buffer.

```
FIELD #file,length AS stringVariable
```

CoCo 3 BASIC · Keep a log past bedtime
See also FIELD.

ASC

Return the character code of the first character.

```
ASC(string)
```

The string must not be empty.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

ATN

Return the arctangent in radians.

```
ATN(number)
```

Multiply by 180/PI to convert the result to degrees.
CoCo 3 BASIC · Count the supplies

ATTR

Choose text colors and optional character attributes.

```
ATTR foreground,background[,B][,U]
```

B enables blinking; U enables underlining.
CoCo 3 high-resolution text modes.

CoCo 3 BASIC · Design a flight deck
See also WIDTH, LOCATE, PALETTE.

AUDIO

Connect or disconnect cassette playback sound from the CoCo audio output.

```
AUDIO ON
AUDIO OFF
```

Use the cassette transport and audio settings in VCC Plus to supply and hear tape playback.
CoCo 3 BASIC · Listen for a signal
See also MOTOR, CLOAD, SOUND.

B

BACKUP · BRK · BUTTON

BACKUP

Duplicate a BASIC disk onto another disk.

```
BACKUP source TO destination
BACKUP drive
```

Single-drive form requires disk exchanges.
Overwrites the destination disk. Use a new disposable image, never the only copy of your data.
Disk BASIC · Take the voyage to another machine
See also COPY.

BRK

Introduce a Break-key trap as part of ON BRK GOTO.

```
ON BRK GOTO line
```

CoCo 3 BASIC · Find the missing supply
See also ON BRK.

BUTTON

Read a joystick button on the CoCo 3.

```
BUTTON(number)
```

0/1: right first/second button; 2/3: left first/second button.
Returns 1 when pressed, otherwise 0.
CoCo 3 BASIC · Listen for a signal
See also JOYSTK.

C

CHR$ · CIRCLE · CLEAR · CLOAD · CLOADM · CLOSE · CLS · CMP · COLOR · CONT · COPY · COPY — RGB-DOS · COS · CSAVE · CSAVEM · CVN

CHR$

Return the character with the specified code.

```
CHR$(code)
```

code: 0-255.
Control codes may move the cursor or change display behavior.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

CIRCLE

Draw a circle, ellipse or arc on the PMODE screen.

```
CIRCLE(x,y),radius[,color[,ratio[,start,end]]]
```

ratio changes height relative to width.
start and end select fractions of a full turn.
CoCo 3 BASIC · Draw a planet you can find again
See also LINE, PAINT, HCIRCLE.

CLEAR

Clear variables and adjust string space or the BASIC memory ceiling.

```
CLEAR [stringBytes [, topAddress]]
```

stringBytes reserves string workspace.
topAddress leaves higher memory for machine-language code.
This discards current variable values; do not use it merely to clear the display.
CoCo 3 BASIC · Give a chart its names
See also MEM, OS — Out of string space.

CLOAD

Load a BASIC program from cassette.

```
CLOAD ["filename"]
```

Omitting the name selects the next BASIC program.
Cassette names have up to eight characters.
Loading replaces the program in CoCo memory. It does not open a Mac source document.
CoCo 3 BASIC · Take the voyage to another machine
See also CLOADM, CSAVE.

CLOADM

Load a machine-language program from cassette.

```
CLOADM ["filename"[,offset]]
```

offset relocates loaded bytes; it does not rewrite addresses inside the program.
CoCo 3 BASIC · Take the voyage to another machine
See also EXEC, CSAVEM, CLOAD.

CLOSE

Finish access to one or more open files.

```
CLOSE [#file[,#file ...]]
```

With no file number, close all open files.
Closing output files completes buffered writes.
Disk BASIC · Keep a log past bedtime
See also OPEN, UNLOAD.

CLS

Clear the text screen.

```
CLS [color]
```

color chooses a text-screen background when supplied.
CoCo 3 BASIC · Make the CoCo answer
In BASIC Quick Reference, open *Hello, CoCo*.
See also PRINT, WIDTH, HCLS, PCLS.

CMP

Restore the standard composite palette on a CoCo 3.

```
PALETTE CMP
```

CoCo 3 BASIC · Draw a planet you can find again
See also PALETTE, RGB.

COLOR

Choose foreground and background colors for PMODE graphics.

```
COLOR foreground,background
```

Available colors depend on the current graphics mode.
CoCo 3 BASIC · Draw a planet you can find again
See also PMODE, PCLS, PSET.

CONT

Continue after an ordinary BASIC stop when the interpreter still has a valid continuation.

```
CONT
```

The editor Continue button resumes its debugger stop. It does not type this command.
Changing the program can invalidate BASIC's saved continuation.
CoCo 3 BASIC · Find the missing supply
See also STOP, CN — Cannot continue.

COPY

Copy one BASIC disk file to another filename or drive.

```
COPY "source/ext:drive" TO "destination/ext:drive"
```

Include the filename extensions.
Writes to the destination disk. RGB-DOS changes overwrite behavior; see its variant entry.
Disk BASIC · Take the voyage to another machine
See also COPY — RGB-DOS, BACKUP.

COPY — RGB-DOS

Copy a file, retaining its name when the destination is only a drive number.

```
COPY "source/ext:drive" TO drive
COPY "source/ext:drive" TO "destination/ext:drive"
```

Writes the destination. Direct mode asks before replacing an existing file; inside a BASIC program, replacement occurs without that prompt.
Hard Drive BASIC (RGB-DOS) · Take the voyage to another machine
See also COPY, DRIVE — RGB-DOS.

COS

Return the cosine of an angle in radians.

```
COS(angle)
```

Angles are radians, not degrees.
CoCo 3 BASIC · Count the supplies

CSAVE

Record the current BASIC program on cassette.

```
CSAVE "filename"[,A]
```

A selects ASCII rather than tokenized program storage.
Prepare the cassette recording transport first. Export the recording in VCC Plus to retain it as a Mac file.
CoCo 3 BASIC · Take the voyage to another machine
See also CLOAD, CSAVEM.

CSAVEM

Record a memory range as a machine-language cassette file.

```
CSAVEM "filename",first,last,entry
```

first and last are inclusive byte addresses; entry is the EXEC address.
Prepare cassette recording before issuing the command.
CoCo 3 BASIC · Take the voyage to another machine
See also CLOADM, EXEC.

CVN

Decode a five-byte BASIC number stored in a string.

```
CVN(string)
```

The string must contain the expected numeric encoding.
Disk BASIC · Keep a log past bedtime
See also MKN$.

D

DATA · DEF · DEF FN · DEF USR · DEL · DIM · DIR · DIR — SDC-DOS · DLOAD · DOS · DOS — RGB-DOS · DRAW · DRIVE · DRIVE NEW · DRIVE — RGB-DOS · DRIVE — SDC-DOS · DSKI$ · DSKINI · DSKO$

DATA

Place constant values in a program for READ.

```
DATA constant [, constant ...]
```

Use quotes around strings containing commas.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Read a small data table*.
See also READ, RESTORE.

DEF

Introduce a user-defined BASIC or machine-language function.

```
DEF FNname(argument)=expression
DEF USRn=address
```

CoCo 3 BASIC · Give a routine a job
See also DEF FN, DEF USR.

DEF FN

Define a numeric function to evaluate later.

```
DEF FNname(parameter) = expression
```

FNname is invoked with a numeric argument.
CoCo 3 BASIC · Give a routine a job
In BASIC Quick Reference, open *Numbers and rounding*.
See also FN.

DEF USR

Set the entry address of a machine-language function.

```
DEF USRn = address
```

n: 0-9; the function must follow BASIC's machine-language calling convention.
CoCo 3 BASIC · Where BASIC meets the machine
See also USR, EXEC.

DEL

Delete one or more lines from the CoCo's in-memory program.

```
DEL first[–last]
DEL –last
DEL first–
```

Use care: the interpreter does not provide Mac-style Undo for this command.
CoCo 3 BASIC · Make the CoCo answer
See also LIST, NEW.

DIM

Reserve numeric or string arrays with inclusive upper bounds.

```
DIM name(upperBound [, upperBound ...]) [, ...]
```

Indices start at zero: DIM A(5) provides six elements.
Dimension an array before using it; do not dimension it twice.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Array bounds and elements*.
See also BS — Bad subscript, DD — Duplicate dimension.

DIR

List files stored on a BASIC disk.

```
DIR [drive]
```

Omit drive to list the default drive.
The listing shows name, extension, type, storage form and granule count.
In SDC-DOS, plain DIR still lists the mounted disk; DIR - browses the SD folder.
Disk BASIC · Take the voyage to another machine
In BASIC Quick Reference, open *List the mounted disk*.
See also DRIVE, DRIVE — RGB-DOS, DIR — SDC-DOS.

DIR — SDC-DOS

Browse the card folder instead of the files inside a mounted disk.

```
DIR –
DIR "path"
DIR = "path"
```

DIR = "/" selects the card root; DIR = ".." selects its parent folder.
Paths are relative to the SDC card folder selected in VCC Plus, not arbitrary Mac paths.
Plain DIR 0 still lists files inside the image mounted in drive 0.
CoCo SDC (SDC-DOS) · Take the voyage to another machine
In BASIC Quick Reference, open *Browse and mount Hello Disk*.
See also DIR, DRIVE — SDC-DOS.

DLOAD

Historical serial program-loading command.

```
DLOAD ["filename"]
```

Requires a compatible serial host and protocol.

The bundled CoCo 3 environment does not provide the historical DLOAD host service. Use native source opening or media loading instead.
CoCo 3 BASIC · Where BASIC meets the machine
See also CLOAD, LOAD.

DOS

Invoke a boot loader from drive 0.

```
DOS
```

Drive 0 must contain a compatible boot loader.
A boot loader can start another operating system or a program; DOS does not always mean OS-9.
Disk BASIC · Take the voyage to another machine
See also DOS — RGB-DOS.

DOS — RGB-DOS

Boot a selected BASIC volume, or try its AUTOEXEC program.

```
DOS [drive]
```

The NitrOS-9 preset boots automatically. Its running OS-9 shell is not a BASIC editor target.
Hard Drive BASIC (RGB-DOS) · Take the voyage to another machine
See also DOS, DRIVE — RGB-DOS.

DRAW

Draw PMODE graphics using a compact movement string.

```
DRAW string
```

U,D,L,R move in the four main directions; E,F,G,H move diagonally.
BM moves without drawing. C selects color, S scale and A angle.
X executes a string variable embedded in the drawing sequence.
CoCo 3 BASIC · Draw a planet you can find again
See also HDRAW, LINE.

DRIVE

Choose the default drive for later disk commands.

```
DRIVE number
```

Disk BASIC uses drive numbers 0-3.
RGB-DOS and SDC-DOS extend this command in different ways.
Disk BASIC · Take the voyage to another machine
See also DIR, DRIVE — RGB-DOS, DRIVE — SDC-DOS.

DRIVE NEW

Create and mount a new disk image in the selected card folder.

```
DRIVE drive,"filename.DSK",NEW
```

Writes a new file; it does not replace an existing one.

Hardware documentation also covers SDF, flash, RTC and DriveWire features. These are not supported SDC features in VCC Plus.
CoCo SDC (SDC-DOS) · Take the voyage to another machine
See also DRIVE — SDC-DOS, DIR — SDC-DOS, DSKINI.

DRIVE — RGB-DOS

Switch between virtual hard-disk volumes and floppy drives.

```
DRIVE volume
DRIVE ON
DRIVE OFF [lastFloppy]
DRIVE #unit
```

volume: 0-255 selects a default virtual BASIC disk. With DRIVE OFF, drives 0-3 refer to floppy drives instead.
DRIVE OFF exposes floppy drives; DRIVE ON restores hard-disk routing.
#unit selects the hard-disk unit. VCC Plus has HD0 and HD1.
The Hard Drive BASIC preset uses RGB-DOS. Do not confuse a virtual BASIC volume number with an HD device number.
Historical DRIVE STOP and DRIVE RESTORE hardware operations are not needed for VCC Plus image files.
Hard Drive BASIC (RGB-DOS) · Take the voyage to another machine
In BASIC Quick Reference, open *Floppy and hard-disk directories*.
See also DRIVE, DIR, DOS — RGB-DOS.

DRIVE — SDC-DOS

Inspect, mount or unload images on the SDC card.

```
DRIVE
DRIVE drive,"path"[,volume]
DRIVE drive,UNLOAD
```

drive is 0 or 1; volume selects a BASIC volume inside a compatible hard-disk image.
Use short card filenames. A mounted image is a disk containing BASIC files, not itself a BASIC program.
The same image cannot be mounted in both drives.
CoCo SDC (SDC-DOS) · Take the voyage to another machine
In BASIC Quick Reference, open *Browse and mount Hello Disk*.
See also DRIVE, DIR — SDC-DOS, DRIVE NEW.

DSKI$

Read a disk sector into two string variables.

```
DSKI$ drive,track,sector,string1,string2
```

The strings receive the two halves of a 256-byte sector.
Raw-sector access bypasses normal file interpretation.
Disk BASIC · Take the voyage to another machine
See also DSKO$.

DSKINI

Initialize a disk for BASIC file storage.

```
DSKINI [drive]
```

Select the intended drive explicitly.
Erases the selected disk and disturbs BASIC workspace. Use only a blank disposable image.
Disk BASIC · Take the voyage to another machine
See also FREE, DIR.

DSKO$

Write two strings directly to a disk sector.

```
DSKO$ drive,track,sector,string1,string2
```

Use properly sized data for the selected sector.
Raw writes can corrupt the filesystem. This is not a safe way to save a BASIC program.
Disk BASIC · Take the voyage to another machine
See also DSKI$.

E

EDIT · ELSE · END · EOF · ERLIN · ERNO · ERR · EXEC · EXP

EDIT

Use the original BASIC line editor for one program line.

```
EDIT line
```

The native BASIC editor is a separate way to edit source.
CoCo 3 BASIC · Make the CoCo answer
See also LIST, DEL, RENUM.

ELSE

Introduce the false branch of an IF statement.

```
IF condition THEN statements ELSE statements
```

CoCo 3 BASIC · Choose a destination
See also IF, THEN.

END

Finish the program.

```
END
```

CoCo 3 BASIC · Make the CoCo answer
In BASIC Quick Reference, open *Hello, CoCo*.
See also STOP, RUN.

EOF

Check whether an open input stream is at end of file.

```
EOF(file)
```

0 means more input remains; −1 indicates end of file.

Also applies to cassette input with device −1.
Disk BASIC · Keep a log past bedtime
See also OPEN, INPUT — disk file.

ERLIN

Read the BASIC line number that caused the trapped error.

```
ERLIN
```

Use inside an ON ERR handler.
CoCo 3 BASIC · Find the missing supply
In BASIC Quick Reference, open *Handle an error*.
See also ERNO, ON ERR.

ERNO

Read the numeric code for the last trapped BASIC error.

```
ERNO
```

Use inside an ON ERR handler.
CoCo 3 BASIC · Find the missing supply
In BASIC Quick Reference, open *Handle an error*.
See also ERLIN, ON ERR.

ERR

Introduce an error trap as part of ON ERR GOTO.

```
ON ERR GOTO line
```

CoCo 3 BASIC · Find the missing supply
See also ON ERR, ERNO, ERLIN.

EXEC

Transfer execution to a machine-language entry point.

```
EXEC [address]
```

Without an address, use the entry point from the last applicable machine-language load.
Running arbitrary memory as code can crash or alter the emulated machine.
CoCo 3 BASIC · Where BASIC meets the machine
See also LOADM, CLOADM, DEF USR.

EXP

Raise e to a numeric power.

```
EXP(number)
```

Large arguments can overflow.
CoCo 3 BASIC · Count the supplies

F

FIELD · FILES · FIX · FN · FOR · FREE

FIELD

Divide a direct-access buffer into string fields.

```
FIELD #file,length AS stringVariable[, ...]
```

Field lengths must fit in the opened record.
Use LSET/RSET rather than ordinary string assignment when filling fields.
Disk BASIC · Keep a log past bedtime
See also OPEN, LSET, RSET, GET — disk record, PUT — disk record.

FILES

Allocate file buffers for disk access.

```
FILES count[,size]
```

count reserves file buffers; size supplies the total buffer size.
Changes BASIC workspace allocation. Arrange file buffers before storing working data.
Disk BASIC · Keep a log past bedtime
See also OPEN, FIELD.

FIX

Remove the fractional part toward zero.

```
FIX(number)
```

FIX(-2.7) is −2.
CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *Numbers and rounding*.
See also INT.

FN

Invoke a function defined with DEF FN.

```
FNname(argument)
```

CoCo 3 BASIC · Give a routine a job
See also DEF FN.

FOR

Start a counted loop.

```
FOR variable = start TO finish [STEP increment]
```

increment defaults to 1; a negative value counts downward.
NEXT closes the loop.
CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *A counting loop*.

See also NEXT, TO, STEP.

FREE

Count free granules on a BASIC disk.

```
FREE(drive)
```

Granules are the BASIC filesystem's allocation units.
Disk BASIC · Take the voyage to another machine
See also DIR.

G

GET · GET — disk record · GO · GOSUB · GOTO

GET

Copy a PMODE rectangle into a numeric array.

```
GET(x1,y1)-(x2,y2),array[,G]
```

Dimension a two-dimensional array large enough for the rectangle.
G stores full graphics detail for PUT operations.
GET #file,record is a different form for disk files.
CoCo 3 BASIC · Draw a planet you can find again
See also PUT, GET — disk record.

GET — disk record

Read a direct-access record into its file buffer.

```
GET #file[,record]
```

Use OPEN "D" and FIELD to interpret the record.
An omitted record advances through the file.
This form is separate from graphics GET.
Disk BASIC · Keep a log past bedtime
See also FIELD, PUT — disk record, GET.

GO

Prefix of GO TO and GO SUB; the joined forms are also accepted.

```
GO TO line
GO SUB line
```

CoCo 3 BASIC · Choose a destination
See also GOTO, GOSUB.

GOSUB

Call a numbered subroutine and remember where to return.

```
GOSUB line
```

End the subroutine with RETURN.
Use the editor's Step Into, Step Over or Step Out to follow calls.
CoCo 3 BASIC · Give a routine a job
In BASIC Quick Reference, open *A returning subroutine*.
See also RETURN, ON.

GOTO

Continue execution at another numbered line.

```
GOTO line
```

line must exist in the loaded program.
CoCo 3 BASIC · Choose a destination
In BASIC Quick Reference, open *Choose a branch*.
See also GOSUB, ON, UL — Undefined line.

H

HBUFF · HCIRCLE · HCLS · HCOLOR · HDRAW · HEX$ · HGET · HLINE · HPAINT · HPOINT · HPRINT · HPUT · HRESET · HSCREEN · HSET · HSTAT

HBUFF

Reserve a buffer for high-resolution GET/PUT graphics.

```
HBUFF buffer,size
```

size is the number of bytes to reserve; maximum 7931.
CoCo 3 BASIC · Draw a planet you can find again
See also HGET, HPUT.

HCIRCLE

Draw a high-resolution circle, ellipse or arc.

```
HCIRCLE(x,y),radius[,color[,ratio[,start,end]]]
```

Optional ratio changes the shape; start/end restrict the arc.
CoCo 3 BASIC · Draw a planet you can find again
See also HLINE, HPAINT, CIRCLE.

HCLS

Fill the CoCo 3 high-resolution graphics screen.

```
HCLS [color]
```

Omitting color uses the current background.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *CoCo 3 graphics*.
See also HSCREEN, HCOLOR.

HCOLOR

Choose foreground and background palette entries for high-resolution graphics.

```
HCOLOR foreground,background
```

Use palette entries available in the selected HSCREEN mode.
CoCo 3 BASIC · Draw a planet you can find again
See also HSCREEN, PALETTE.

HDRAW

Use a movement string to draw high-resolution graphics.

```
HDRAW string
```

Uses the movement commands described under DRAW.
CoCo 3 BASIC · Draw a planet you can find again
See also DRAW, HLINE.

HEX$

Convert a number to hexadecimal text.

```
HEX$(number)
```

The returned string has no &H prefix.
CoCo 3 BASIC · Give a chart its names

HGET

Save a high-resolution rectangle in an HBUFF buffer.

```
HGET(x1,y1)-(x2,y2),buffer
```

Reserve sufficient space with HBUFF before copying.
CoCo 3 BASIC · Draw a planet you can find again
See also HBUFF, HPUT, GET.

HLINE

Draw a high-resolution line or rectangle.

```
HLINE [(x1,y1)]-(x2,y2),color[,B or BF]
```

B draws a box; BF fills it.
color may be a palette entry, PSET or PRESET.
CoCo 3 BASIC · Draw a planet you can find again
See also HSET, HCIRCLE, LINE.

HPAINT

Fill a bounded high-resolution region.

```
HPAINT(x,y)[,color[,border]]
```

Select a point inside a closed border.
CoCo 3 BASIC · Draw a planet you can find again
See also HCIRCLE, HLINE, PAINT.

HPOINT

Read the color of a high-resolution graphics point.

```
HPOINT(x,y)
```

Coordinates follow the selected HSCREEN mode.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *CoCo 3 graphics*.
See also HSET.

HPRINT

Draw text in a CoCo 3 high-resolution graphics display.

```
HPRINT(column,row),string
```

Coordinates select character cells, not individual pixels.
CoCo 3 BASIC · Draw a planet you can find again
See also HSCREEN, HCOLOR, PRINT.

HPUT

Restore a high-resolution rectangle from a buffer.

```
HPUT(x1,y1)-(x2,y2),buffer,action
```

action: PSET, PRESET, AND, OR or NOT.
CoCo 3 BASIC · Draw a planet you can find again
See also HBUFF, HGET, PUT.

HRESET

Reset a high-resolution point to the background.

```
HRESET(x,y)
```

CoCo 3 BASIC · Draw a planet you can find again
See also HSET, HPOINT.

HSCREEN

Select a CoCo 3 high-resolution graphics display.

```
HSCREEN mode
```

0: return to low-resolution display.
1: 320x192, 4 colors; 2: 320x192, 16 colors.
3: 640x192, 2 colors; 4: 640x192, 4 colors.
Selecting modes 1-4 also clears that graphics display.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *CoCo 3 graphics*.
See also HCLS, HCOLOR, PALETTE.

HSET

Set a point in CoCo 3 high-resolution graphics.

```
HSET(x,y[,color])
```

Omitting color uses the foreground.
Coordinates depend on the HSCREEN resolution.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *CoCo 3 graphics*.
See also HRESET, HPOINT.

HSTAT

Read the high-resolution text character, attributes and cursor coordinates.

```
HSTAT character,attribute,column,row
```

Supply numeric variables to receive the four results.
CoCo 3 BASIC · Design a flight deck
See also LOCATE, ATTR, WIDTH.

I

IF · INKEY$ · INPUT · INPUT — disk file · INSTR · INT

IF

Choose statements or a destination line according to a test.

```
IF condition THEN statements [ELSE statements]
IF condition THEN line [ELSE line]
```

Zero is false; a nonzero value is true.
A false condition skips the remainder of its branch on that BASIC line.
CoCo 3 BASIC · Choose a destination
In BASIC Quick Reference, open *Choose a branch*.
See also THEN, ELSE, AND, OR.

INKEY$

Read a waiting key without stopping for input.

```
INKEY$
```

The result is an empty string when no key is waiting.
CoCo 3 BASIC · Listen for a signal

INPUT

Read keyboard values into numeric or string variables.

```
INPUT ["prompt";] variable [, variable ...]
```

Separate multiple responses with commas.
Use LINE INPUT when a whole line, including commas, belongs in one string.
CoCo 3 BASIC · Choose a destination
See also LINE INPUT, INKEY$.

INPUT — disk file

Read delimited values from an open input file.

```
INPUT #file,variable[,variable ...]
```

Variable types must match the stored fields.
Disk BASIC · Keep a log past bedtime
See also OPEN, WRITE, EOF.

INSTR

Find the first occurrence of a substring.

```
INSTR([start,]string,target)
```

Positions start at 1; a zero result means no match.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

INT

Round down to an integer, including for negative values.

```
INT(number)
```

INT(-2.7) is −3.
CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *Numbers and rounding*.
See also FIX.

J

JOYSTK

JOYSTK

Read an emulated joystick axis.

```
JOYSTK(axis)
```

0: right X; 1: right Y; 2: left X; 3: left Y.
Values range from 0 to 63; center is about 31.
CoCo 3 BASIC · Listen for a signal
See also BUTTON.

K

KILL

KILL

Remove a file from a BASIC disk directory.

```
KILL "filename/ext[:drive]"
```

Include the extension.
Deletes the file from the selected disk; the command has no Undo.
Disk BASIC · Take the voyage to another machine
See also DIR, RENAME.

L

LEFT$ · LEN · LET · LINE · LINE INPUT · LINE INPUT — disk file · LIST · LLIST · LOAD · LOADM · LOC · LOCATE · LOF · LOG · LPEEK · LPOKE · LSET

LEFT$

Take characters from the beginning of a string.

```
LEFT$(string,count)
```

count is the maximum number of characters returned.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

LEN

Count the characters in a string.

```
LEN(string)
```

CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

LET

Assign a value to a variable; LET is optional.

```
[LET] variable = expression
```

Use a name ending in $ for a string.
Only the first two characters of a variable name distinguish it in this BASIC.
Changing source in the editor does not change variables in a running CoCo program.
CoCo 3 BASIC · Choose a destination
In BASIC Quick Reference, open *Numbers and rounding*.
See also DIM.

LINE

Draw a PMODE line, rectangle or filled rectangle.

```
LINE [(x1,y1)]-(x2,y2),action[,B or BF]
```

action is PSET or PRESET; B draws a rectangle and BF fills it.
If the first point is omitted, drawing starts at the previous endpoint.
B or BF can be used with either PSET or PRESET.
CoCo 3 BASIC · Draw a planet you can find again
See also PSET, CIRCLE, HLINE.

LINE INPUT

Read an entire line into a string variable.

```
LINE INPUT ["prompt";] stringVariable
```

Commas in the entered line remain part of the string.
CoCo 3 BASIC · Choose a destination
See also INPUT, LINE INPUT — disk file.

LINE INPUT — disk file

Read an entire file line into one string.

```
LINE INPUT #file,stringVariable
```

Commas remain part of the string.
Disk BASIC · Keep a log past bedtime
See also OPEN, EOF, LINE INPUT.

LIST

Display a program or a range of BASIC lines.

```
LIST [first[-last]]
LIST -last
LIST first-
```

A single line number lists only that line.
Read from CoCo brings program text into the native editor without typing LIST.
CoCo 3 BASIC · Make the CoCo answer
See also LLIST, EDIT.

LLIST

Send a program listing to the printer device.

```
LLIST [first[-last]]
```

Enable VCC Plus Printer Output capture to collect printer text.
CoCo 3 BASIC · Design a flight deck
See also LIST, PRINT.

LOAD

Load a BASIC program from disk.

```
LOAD "filename"[,R]
```

R runs the program after loading.
A :drive suffix selects a drive instead of the default.
Loading replaces the BASIC program in CoCo memory.
Disk BASIC · Take the voyage to another machine
See also SAVE, RUN — disk file.

LOADM

Load a machine-language file from disk.

```
LOADM "filename"[,offset]
```

offset changes where bytes are loaded; internal machine-code addresses are not relocated.
Use EXEC after loading unless the file implements its own autostart.
Disk BASIC · Take the voyage to another machine
See also EXEC, SAVEM.

LOC

Read the current record number for an open disk file.

```
LOC(file)
```

file refers to an open disk buffer.
Disk BASIC · Keep a log past bedtime
See also OPEN, LOF.

LOCATE

Position the cursor in the CoCo 3 high-resolution text screen.

```
LOCATE column,row
```

Coordinates start at zero.
Use WIDTH 40 or WIDTH 80 first.
CoCo 3 BASIC · Design a flight deck
See also WIDTH, ATTR, HSTAT.

LOF

Read the highest record number in an open disk file.

```
LOF(file)
```

This is a record count, not a Mac file-size query.
Disk BASIC · Keep a log past bedtime
See also OPEN, LOC.

LOG

Return the natural logarithm.

```
LOG(number)
```

number must be positive.
CoCo 3 BASIC · Count the supplies

LPEEK

Read a byte using the CoCo 3 long-memory interface.

```
LPEEK(address)
```

address: 0-524287. This BASIC interface is limited to 512 KB.
CoCo 3 BASIC · Where BASIC meets the machine
See also LPOKE.

LPOKE

Write a byte using the CoCo 3 BASIC long-memory interface.

```
LPOKE address,value
```

address: 0-524287; value: 0-255.
This command's address range does not expand simply because the emulator has more RAM.
CoCo 3 BASIC · Where BASIC meets the machine
See also LPEEK, POKE.

LSET

Left-align text in a direct-access field.

```
LSET stringField = string
```

Short strings are padded; long strings are limited to the field length.
Disk BASIC · Keep a log past bedtime
See also FIELD, RSET.

M

MEM · MERGE · MID$ · MKN$ · MOTOR

MEM

Report currently free BASIC memory.

```
MEM
```

Available BASIC workspace is not the same as the selected total machine RAM.
CoCo 3 BASIC · Where BASIC meets the machine

MERGE

Combine a disk ASCII listing with the program in memory.

```
MERGE "filename"[,R]
```

Save the source file with SAVE ...,A first.
R starts the merged program.
Duplicate line numbers are replaced by the incoming lines.
Disk BASIC · Take the voyage to another machine
See also LOAD, SAVE.

MID$

Read a substring, or replace characters in a string variable.

```
MID$(string,start[,count])
MID$(stringVariable,start[,count]) = replacement
```

Positions start at 1.
Without count, the function reads to the end.
Assignment changes existing characters; it does not insert new space.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

MKN$

Encode a BASIC number into its five-byte representation.

```
MKN$(number)
```

Use with fixed-length binary fields; this is not decimal text.
Disk BASIC · Keep a log past bedtime
See also CVN, FIELD.

MOTOR

Control the cassette motor signal.

```
MOTOR ON
MOTOR OFF
```

CoCo 3 BASIC · Take the voyage to another machine
See also AUDIO, CLOAD, CSAVE.

N

NEW · NEXT · NOT

NEW

Erase the program currently held by BASIC.

```
NEW
```

This changes CoCo memory, not a saved Mac source file.
CoCo 3 BASIC · Make the CoCo answer
See also CLEAR, RUN.

NEXT

Advance a FOR loop and test whether it should repeat.

```
NEXT [variable [, variable ...]]
```

For nested loops, close the innermost loop first.
CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *A counting loop*.
See also FOR, NF — NEXT without FOR.

NOT

Invert an integer bit pattern or a BASIC truth value.

```
NOT expression
```

CoCo 3 BASIC · Choose a destination
See also AND, OR, IF.

O

OFF · ON · ON BRK · ON ERR · OPEN · OR

OFF

Disable a feature in commands that accept an ON/OFF switch.

```
AUDIO OFF
MOTOR OFF
VERIFY OFF
```

CoCo 3 BASIC · Listen for a signal
See also AUDIO, MOTOR, VERIFY.

ON

Select a GOTO or GOSUB destination by its position in a list.

```
ON expression GOTO line [, line ...]
ON expression GOSUB line [, line ...]
```

A value of 1 selects the first destination.
Zero or a value beyond the list falls through.
CoCo 3 BASIC · Give a routine a job
In BASIC Quick Reference, open *Choose a branch*.
See also GOTO, GOSUB, ON ERR, ON BRK.

ON BRK

Branch to a handler when the CoCo Break key is pressed.

```
ON BRK GOTO line
```

CoCo 3 extension.
ON BRK GOTO 0 restores ordinary Break handling.
CoCo 3 BASIC · Find the missing supply
See also STOP, ON ERR.

ON ERR

Send runtime errors to a BASIC handler on the CoCo 3.

```
ON ERR GOTO line
```

Use ERNO and ERLIN in the handler to identify the error.
ON ERR GOTO 0 disables the handler.

Handled errors remain under BASIC control; VCC Plus does not replace them with its unhandled-error pause.
CoCo 3 BASIC · Find the missing supply
In BASIC Quick Reference, open *Handle an error*.
See also ERNO, ERLIN, SN — Syntax error.

OPEN

Open a data stream to a file or device.

```
OPEN "mode",#file,"filename"[,recordLength]
```

I reads sequential data; O writes sequential data; D selects direct-access disk records.
Disk file numbers: 1-15, subject to FILES allocation.
Device −1 is cassette; −2 is printer; 0 is console.
O can replace an existing file; use a disposable output filename while learning.
Disk BASIC · Keep a log past bedtime
See also CLOSE, FILES, FIELD.

OR

Combine conditions or integer bit patterns when either side is set.

```
expression OR expression
```

CoCo 3 BASIC · Choose a destination
See also AND, NOT, IF.

P

PAINT · PALETTE · PCLEAR · PCLS · PCOPY · PEEK · PLAY · PMODE · POINT · POKE · POS · PPOINT · PRESET · PRINT · PRINT USING · PRINT — file or printer · PRINT@ · PSET · PUT · PUT — disk record

PAINT

Fill a bounded PMODE area from a starting point.

```
PAINT(x,y)[,color[,border]]
```

The starting point must be inside a closed border for a bounded fill.
CoCo 3 BASIC · Draw a planet you can find again
See also CIRCLE, LINE, HPAINT.

PALETTE

Assign a CoCo 3 palette entry or restore a standard palette.

```
PALETTE register,color
PALETTE CMP
PALETTE RGB
```

register: 0-15; color: 0-63.
CMP and RGB choose standard tables for the corresponding monitor signal.
The BASIC palette and VCC Plus Video Signal setting are related but separate controls.
CoCo 3 BASIC · Draw a planet you can find again
See also HSCREEN, HCOLOR, CMP, RGB.

PCLEAR

Reserve 1.5 KB pages for PMODE graphics.

```
PCLEAR pages
```

Reserve enough pages for the chosen PMODE and starting page.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PMODE, PCOPY, MEM.

PCLS

Fill the current PMODE graphics screen.

```
PCLS [color]
```

Omitting color uses the current background.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PMODE, COLOR, CLS.

PCOPY

Copy one reserved PMODE graphics page to another.

```
PCOPY source TO destination
```

Pages must have been reserved with PCLEAR.
CoCo 3 BASIC · Draw a planet you can find again
See also PCLEAR, PMODE.

PEEK

Read a byte through the current CPU memory map.

```
PEEK(address)
```

address: 0-65535.
Reading device registers can affect the device.
CoCo 3 BASIC · Where BASIC meets the machine
See also POKE.

PLAY

Play notes described by a music string.

```
PLAY string
```

A-G name notes; O selects octave, L note length and T tempo.
or + sharpens a note; - flattens it; P inserts a pause.
CoCo 3 BASIC · Listen for a signal
See also SOUND.

PMODE

Choose the resolution and starting page for traditional CoCo graphics.

```
PMODE mode,page
```

0: 128x96, 2 colors; 1: 128x96, 4 colors.
2: 128x192, 2 colors; 3: 128x192, 4 colors.
4: 256x192, 2 colors.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PCLEAR, SCREEN, PCLS.

POINT

Read a semigraphics block on the text screen.

```
POINT(x,y)
```

-1 indicates part of an ordinary text character; otherwise the result describes the block color.
CoCo 3 BASIC · Draw a planet you can find again
See also SET.

POKE

Write one byte through the current CPU memory map.

```
POKE address,value
```

address: 0-65535; value: 0-255.
Writes can change programs, device registers or system state. Use only addresses you understand.
CoCo 3 BASIC · Where BASIC meets the machine
See also PEEK, LPOKE.

POS

Read the current print-column position for a device.

```
POS(device)
```

device 0 is the screen; −2 is the printer.
CoCo 3 BASIC · Design a flight deck

PPOINT

Read the color of a PMODE graphics point.

```
PPOINT(x,y)
```

CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PSET.

PRESET

Reset a PMODE point to the background color.

```
PRESET(x,y)
```

CoCo 3 BASIC · Draw a planet you can find again
See also PSET, PPOINT.

PRINT

Display values on the text screen.

```
PRINT [expression [; or , expression] ...]
```

Semicolon joins output without a print-zone advance.
Comma advances to the next print zone.
A trailing separator suppresses the usual newline.
? is a shorthand for PRINT.
CoCo 3 BASIC · Make the CoCo answer
In BASIC Quick Reference, open *Hello, CoCo*.
See also PRINT@, USING, TAB.

PRINT USING

Print values with a reusable layout.

```
PRINT USING "format";expression[;expression ...]
```

Each # reserves a digit position; a decimal point places the fractional part.
For strings, ! prints one character and % followed by spaces and % defines a fixed-width field.
CoCo 3 BASIC · Design a flight deck
In BASIC Quick Reference, open *Formatted output*.
See also PRINT, USING, TAB.

PRINT — file or printer

Write formatted values to an open file or output device.

```
PRINT #file,expression[; or , expression ...]
```

-2 selects the printer; −1 selects cassette.
Positive file numbers refer to previously opened disk files.
Disk/cassette output writes media. Printer text is captured only when the VCC Plus printer is enabled.
Disk BASIC · Keep a log past bedtime
See also OPEN, CLOSE, WRITE, USING.

PRINT@

Place text at a numbered position on the 32-column screen.

```
PRINT @position, expression
```

position: 0-511, counted across rows from the top left.
CoCo 3 BASIC · Design a flight deck
See also PRINT, LOCATE, WIDTH.

PSET

Set a point in the PMODE drawing area.

```
PSET(x,y[,color])
```

Drawing coordinates use x: 0-255 and y: 0-191.
Omit color to use the current foreground.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PRESET, PPOINT.

PUT

Draw a PMODE rectangle from a saved numeric array.

```
PUT(x1,y1)-(x2,y2),array[,action]
```

action: PSET, PRESET, AND, OR or NOT.
Use a matching rectangle and storage prepared by GET.
CoCo 3 BASIC · Draw a planet you can find again
See also GET, PUT — disk record.

PUT — disk record

Write the file buffer as a direct-access record.

```
PUT #file[,record]
```

Use LSET/RSET to fill the buffer before writing.
This changes the opened disk file. It is separate from graphics PUT.
Disk BASIC · Keep a log past bedtime
See also FIELD, GET — disk record, PUT.

R

READ · REM · RENAME · RENUM · RESET · RESTORE · RETURN · RGB · RIGHT$ · RND · RSET · RUN · RUN — disk file · RUNM

READ

Take the next values from the program's DATA statements.

```
READ variable [, variable ...]
```

Variable type must match the corresponding value.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Read a small data table*.
See also DATA, RESTORE, OD — Out of DATA.

REM

Add a comment that BASIC does not execute.

```
REM comment
```

An apostrophe also starts a comment. Quoted strings and comments retain their entered case.
CoCo 3 BASIC · Make the CoCo answer

See also DATA.

RENAME

Change a filename within the same BASIC disk.

```
RENAME "old/ext[:drive]" TO "new/ext[:drive]"
```

Include both extensions.
Changes the disk directory.
Disk BASIC · Take the voyage to another machine
See also DIR, COPY.

RENUM

Renumber program lines and update recognized line references.

```
RENUM [newStart [, oldStart [, increment]]]
```

newStart is the first assigned number.
oldStart selects the first old line to change.
increment controls spacing.
The editor Renumber action supports Undo and updates editor source rather than the current CoCo program.
CoCo 3 BASIC · Find the missing supply
See also LIST.

RESET

Clear a semigraphics block on the text screen.

```
RESET(x,y)
```

x: 0-63; y: 0-31.
CoCo 3 BASIC · Draw a planet you can find again
See also SET, POINT.

RESTORE

Move the DATA pointer back to the first item.

```
RESTORE
```

CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Read a small data table*.
See also READ, DATA.

RETURN

Resume immediately after the matching GOSUB call.

```
RETURN
```

RETURN needs an active GOSUB call.
CoCo 3 BASIC · Give a routine a job
In BASIC Quick Reference, open *A returning subroutine*.
See also GOSUB, RG — RETURN without GOSUB.

RGB

Restore the standard RGB palette on a CoCo 3.

```
PALETTE RGB
```

CoCo 3 BASIC · Draw a planet you can find again
See also PALETTE, CMP.

RIGHT$

Take characters from the end of a string.

```
RIGHT$(string,count)
```

count is the maximum number of characters returned.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

RND

Generate a pseudo-random value.

```
RND(limit)
```

RND(0) returns a fractional value.
A positive integer limit returns an integer from 1 through limit.
CoCo 3 BASIC · Count the supplies

RSET

Right-align text in a direct-access field.

```
RSET stringField = string
```

Short strings are padded on the left.
Disk BASIC · Keep a log past bedtime
See also FIELD, LSET.

RUN

Start the program in memory, normally clearing its variables.

```
RUN [line]
```

Omit line to start at the beginning.
Use the editor's Run button to transfer edited source and start it.
CoCo 3 BASIC · Make the CoCo answer
In BASIC Quick Reference, open *Hello, CoCo*.
See also END, RUN — disk file.

RUN — disk file

Load and run a BASIC program from disk.

```
RUN "filename"[,R]
```

R preserves open files while chaining to the new program.
Replaces the in-memory BASIC program.
Disk BASIC · Take the voyage to another machine
See also LOAD, RUN.

RUNM

Load and execute a machine-language program in one command.

```
RUNM "filename[:drive]"
```

For binaries that start themselves during loading, use LOADM instead; RUNM can execute their entry point again.
Hard Drive BASIC (RGB-DOS) · Take the voyage to another machine
See also LOADM, EXEC.

S

SAVE · SAVEM · SCREEN · SET · SGN · SIN · SKIPF · SOUND · SQR · STEP · STOP · STR$ · STRING$ · SUB

SAVE

Write the in-memory BASIC program to disk.

```
SAVE "filename"[,A]
```

A selects an ASCII listing instead of tokenized storage.
Writes to the selected image and may replace an existing file. The native editor Save command instead saves a Mac source file.
Disk BASIC · Take the voyage to another machine
See also LOAD, CSAVE.

SAVEM

Write a memory range as a machine-language disk file.

```
SAVEM "filename",first,last,entry
```

first/last are inclusive addresses; entry is the default EXEC address.
Writes to the selected disk image.
Disk BASIC · Take the voyage to another machine
See also LOADM, EXEC.

SCREEN

Switch between the traditional text and graphics displays.

```
SCREEN type,colorSet
```

type: 0 for text, 1 for PMODE graphics.
colorSet: 0 or 1.
CoCo 3 BASIC · Draw a planet you can find again
In BASIC Quick Reference, open *Draw and inspect a point*.
See also PMODE, HSCREEN.

SET

Color one semigraphics block on the 32-column text screen.

```
SET(x,y,color)
```

x: 0-63; y: 0-31; color: 0-8.
CoCo 3 BASIC · Draw a planet you can find again
See also RESET, POINT, PSET.

SGN

Return −1, 0 or 1 for a negative, zero or positive number.

```
SGN(number)
```

CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *Numbers and rounding*.

SIN

Return the sine of an angle in radians.

```
SIN(angle)
```

A full turn is 2*PI radians; use a numeric value for PI.
CoCo 3 BASIC · Count the supplies

SKIPF

Advance through a cassette file without loading it.

```
SKIPF ["filename"]
```

CoCo 3 BASIC · Take the voyage to another machine
See also CLOAD, CLOADM.

SOUND

Play a tone for a specified duration.

```
SOUND pitch,duration
```

pitch: 1-255; duration: 1-255.
CoCo 3 BASIC · Listen for a signal
See also PLAY, AUDIO.

SQR

Return the nonnegative square root.

```
SQR(number)
```

number must be zero or positive.
CoCo 3 BASIC · Count the supplies
In BASIC Quick Reference, open *Numbers and rounding*.

STEP

Set the increment of a FOR loop.

```
FOR variable=start TO finish STEP increment
```

CoCo 3 BASIC · Count the supplies
See also FOR, NEXT.

STOP

Stop execution and report the BASIC line.

```
STOP
```

A BASIC STOP is distinct from an editor breakpoint.
CoCo 3 BASIC · Find the missing supply
See also CONT, END.

STR$

Convert a number into a printable string.

```
STR$(number)
```

Positive numbers normally include a leading space.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

STRING$

Create a string containing a repeated character.

```
STRING$(count,codeOrString)
```

A string argument supplies its first character.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Formatted output*.

SUB

Complete the split spelling GO SUB.

```
GO SUB line
```

CoCo 3 BASIC · Give a routine a job
See also GOSUB, RETURN.

T

TAB · TAN · THEN · TIMER · TO · TROFF · TRON

TAB

Advance within a PRINT line to a column.

```
PRINT TAB(column);expression
```

CoCo 3 BASIC · Design a flight deck
See also PRINT, POS.

TAN

Return the tangent of an angle in radians.

```
TAN(angle)
```

Angles near odd multiples of PI/2 produce very large results.
CoCo 3 BASIC · Count the supplies

THEN

Introduce the true branch of an IF statement.

```
IF condition THEN statements
```

CoCo 3 BASIC · Choose a destination
See also IF, ELSE.

TIMER

Read or set the interpreter's ticking counter.

```
TIMER
TIMER = value
```

The counter wraps at 65536.
It advances with the emulated machine, not a guaranteed wall-clock duration.
CoCo 3 BASIC · Listen for a signal

TO

Separate start/end values or source/destination operands.

```
FOR variable=start TO finish
```

CoCo 3 BASIC · Count the supplies
See also FOR, COPY, BACKUP.

TROFF

Turn off the BASIC line-number trace.

```
TROFF
```

CoCo 3 BASIC · Find the missing supply
See also TRON.

TRON

Show BASIC line numbers as the program executes.

```
TRON
```

This is the interpreter's text trace; it is independent of editor breakpoints.
CoCo 3 BASIC · Find the missing supply
See also TROFF.

U

UNLOAD · USING · USR

UNLOAD

Close open files associated with a disk.

```
UNLOAD [drive]
```

Omitting drive selects the default drive.
This BASIC command does not eject the Mac disk-image file from VCC Plus.
Disk BASIC · Take the voyage to another machine
See also CLOSE, DRIVE — SDC-DOS.

USING

Apply a numeric or string layout to printed values.

```
PRINT USING "format";expression
```

CoCo 3 BASIC · Design a flight deck
In BASIC Quick Reference, open *Formatted output*.
See also PRINT, PRINT — file or printer.

USR

Call a machine-language function defined with DEF USR.

```
USRn(argument)
```

n: 0-9. Define a valid entry address and calling convention first.
CoCo 3 BASIC · Where BASIC meets the machine
See also DEF USR.

V

VAL · VARPTR · VERIFY

VAL

Convert a numeric prefix of a string into a number.

```
VAL(string)
```

Parsing follows BASIC numeric notation.
CoCo 3 BASIC · Give a chart its names
In BASIC Quick Reference, open *Working with strings*.

VARPTR

Find the address of a variable's internal storage.

```
VARPTR(variable)
```

BASIC may move storage later; do not assume the address remains valid.
CoCo 3 BASIC · Where BASIC meets the machine

VERIFY

Choose whether disk writes are followed by a verification read.

```
VERIFY ON
VERIFY OFF
```

Verification checks the data written to the emulated disk.
This is an interpreter setting, not the image's write-protection setting.
Disk BASIC · Take the voyage to another machine
See also SAVE.

W

WIDTH · WRITE

WIDTH

Select the CoCo 3 text-screen width.

```
WIDTH columns
```

columns: 32, 40 or 80.
40- and 80-column modes have 24 rows.
CoCo 3 extension.
CoCo 3 BASIC · Design a flight deck
See also LOCATE, ATTR, PRINT.

WRITE

Write comma-delimited data suitable for INPUT.

```
WRITE #file,expression[,expression ...]
```

String values are quoted automatically.
Writes to the opened output file.
Disk BASIC · Keep a log past bedtime
See also OPEN, INPUT, INPUT — disk file.

Where BASIC meets the machine

Appendix B

When you ask BASIC how many supplies are left, you use the name S. Somewhere inside the CoCo, the interpreter keeps bytes for that number. In this little tour you can look at one of those bytes without changing it.

Find a value and its address

Open ADDRESS.bas. Line 20 puts the number 42 in A. The next three lines ask different questions about the same variable. Which line do you expect to print 42?

```
10 WIDTH 40:CLS
20 A=42
30 PRINT "VALUE";A
40 PRINT "ADDRESS ";HEX$(VARPTR(A))
50 PRINT "FIRST BYTE";PEEK(VARPTR(A))
60 END
```

The output labeled **VALUE** is 42. The **ADDRESS** is where BASIC put the variable's data this time. HEX$ displays the address in hexadecimal, a number system with sixteen digits per place. After 9 its digits are A, B, C, D, E and F. Your address can include letters.

The last line with PRINT asks for **one byte**, starting at that address. In the supplied exercise we saw address 2667 in hexadecimal and first byte 134 in decimal. BASIC stores this numeric variable as several bytes. Reading just the first byte gives a piece of the stored number. To learn the value of A while debugging, ask for A in Watches. The whole value is already there.

Use the address reported in **your** session when inspecting memory. Adding program lines or variables can move the data, even if A still holds 42. The VARPTR, HEX$ and PEEK entries describe the three operations in the listing.

Follow the workspace

Think of BASIC's workspace as part of RAM that the interpreter manages on your behalf. It keeps the numbered program, variables and space for strings there. MEM reports the space BASIC still has available. The RAM badge at the top of the emulator reports the capacity of the emulated machine. A program, a screen, and the interpreter can occupy different pieces of that machine.

The processor uses 16-bit addresses: it can name 65,536 byte positions in its current view. The CoCo 3 can contain much more RAM than this. Its memory management unit, or MMU, maps blocks of physical memory into the processor's view. In the debugger this is the distinction between a CPU address and a physical address. It explains why a number from VARPTR only describes the current arrangement.

PEEK reads from the current address space. POKE writes a byte there. A write can change a variable, an instruction, or a hardware setting, depending on the address. Use a separate practice session for an experiment that writes to memory. The CoCo 3's LPEEK and LPOKE accept long addresses up through 524287. That BASIC interface covers the first 512 KB, including on a machine configured with more RAM.

Hand control to machine code

BASIC lines are stored in a form that the interpreter reads. A machine-language program contains instructions that the processor executes. Each has its own loading path. A disk BASIC program such as the one in the export lesson can be loaded by name and started with RUN. LOADM brings machine-code blocks from disk into memory. EXEC transfers control to an entry address. The assembler and the program's loading instructions tell you the right address.

A machine routine can also do one job for a BASIC program. DEF USR assigns a routine's entry address to a USR slot. USR calls the routine with an argument. The caller and routine have to agree on how data goes in and how the result comes back. For an ordinary BASIC subroutine like the landing routine, GOSUB and RETURN already handle the trip.

To explore CPU instructions on the Mac, open **Develop → Assembly Quick Reference…**. For loading old programs, remember that DLOAD once expected a compatible host on a serial connection. VCC Plus provides native source opening and emulated media loading; select the one appropriate to the file you have.

Historical sources and credits

- Tandy Corporation. *Tandy Color Computer 3 BASIC Quick Reference Guide*. 1986. Commands, pp. 2–17; functions, pp. 18–22; operators, p. 23; error codes, p. 27. The short format of the lookup entries follows the same practical idea: a reader at the keyboard should be able to find one operation quickly.
- InterTAN Canada Ltd. *Color Computer Disk BASIC Quick Reference Guide*. 1987. Disk commands, pp. 3–8; disk errors, p. 9. Useful for comparing sequential files with direct-access records.
- Krupski, Roger, Joe Scinta, Greg Miller, and Colin J. Stearman. *RGB-DOS*. Ken-Ton Electronics. COPY, DRIVE, DOS and RUNM, PDF pp. 6–8. Page references here use the pages of the scan.
- Lindner, Tim. *CoCo SDC User Guide*. 4th ed., February 2023. Based on the original manual by Brian Blake; CoCo SDC hardware and software by Darren Atkinson. Drive status and mounting, pp. 9–10; directory and startup operations, pp. 11–12.

The Color Computer Archive preserves historical documentation that informed this book. The lessons and programs here were written for VCC Plus. The printed sources describe period equipment or DOS extensions; the flight exercises, screen captures and Mac controls show how to work with this emulator.

What did that error mean?

Appendix C

A short message can tell you exactly where to start looking. On our empty practice disk, the flight-log reader stopped with **?NE ERROR IN 20**. The two letters identify the problem. The number after IN is the line BASIC was trying to execute.

Look up the code

/0 · AE · AO · BR · BS · CN · DD · DF · DN · DS · ER · FC · FD · FM · FN · FO · FS · HP · HR · ID · IE · IO · LS · NE · NF · NO · OB · OD · OM · OS · OV · RG · SE · SN · ST · TM · UL · VF · WP

For the flight-log example, NE means “file not found.” Open LOGBOOK.bas and look at line 20. It asks drive zero to open a file called VOYAGE.DAT for input. Is the practice disk in FD0? Has the crew already landed and saved a voyage? Either answer is more helpful than changing line 20 at random.

For a program of your own, note the code and line number before starting over. Choose that line in the editor and read it together with the line above it. A bad value can be assigned much earlier than the line that first tries to use it. A breakpoint and a watch let you catch the change while the program is still running.

Check the value

Give each operation the sort of value it expects.

/0 · FC · LS · OV · OS · ST · TM

/0 — Division by zero

A denominator evaluated to zero. Check the values at the reported line.

See also IF, Choose a destination.

FC — Illegal function call

An argument is outside the range accepted by this command or function. Check its parameter descriptions.

See also SQR, ASC, Choose a destination.

LS — String too long

The result exceeds BASIC's maximum string length of 255 characters.

See also LEN, LEFT$, MID$, Choose a destination.

OV — Overflow

A numeric result is too large for BASIC to represent.

See also EXP, Choose a destination.

OS — Out of string space

The reserved string workspace is exhausted. Consider reserving more with CLEAR before creating data.

See also CLEAR, LEN, Choose a destination.

ST — String expression too complex

Simplify the expression by saving intermediate string results in variables.

See also LET, MID$, Choose a destination.

TM — Type mismatch

An expression uses a string where a number is required, or the reverse.

See also VAL, STR$, INPUT, Choose a destination.

Trace the program

Start at the reported line. Check how control reached it.
CN · DS · ID · NF · RG · SN · UL

CN — Cannot continue

BASIC no longer has a valid continuation after a stop. Restart with RUN if appropriate.
See also CONT, RUN, STOP, Find the missing supply.

DS — Direct statement in file

A program being loaded contains a statement without a BASIC line number.
See also LOAD, MERGE, Find the missing supply.

ID — Illegal direct statement

This statement requires execution from a numbered program rather than the immediate prompt.
See also RUN, Find the missing supply.

NF — NEXT without FOR

Execution reached NEXT without a matching active loop. Check loop entry paths.
See also FOR, NEXT, Find the missing supply.

RG — RETURN without GOSUB

There is no active subroutine call to return from. Avoid falling into a subroutine from the main program.
See also GOSUB, RETURN, END, Find the missing supply.

SN — Syntax error

BASIC could not interpret a statement. Check spelling, punctuation, quotes and the command form.
See also PRINT, Find the missing supply.

UL — Undefined line

A branch or run command refers to a line that is not in the program.
See also GOTO, GOSUB, RUN, Find the missing supply.

Count the items

Compare the number of items you set aside with the number you used.
BS · DD · OD · OM

BS — Bad subscript

An array index is outside its declared bounds or has the wrong number of dimensions.
See also DIM, Give a chart its names.

DD — Duplicate dimension

An array has already been dimensioned or implicitly created. Dimension it before its first use.
See also DIM, Give a chart its names.

OD — Out of DATA

READ requested more values than the DATA statements provide.
See also DATA, READ, RESTORE, Give a chart its names.

OM — Out of memory

BASIC has insufficient workspace for the operation. Check arrays, strings, graphics pages and program size.
See also MEM, CLEAR, DIM, PCLEAR, Give a chart its names.

Check the screen mode

Match the graphics mode, coordinates and buffers to the drawing command.
HP · HR

HP — High-resolution print error

The requested high-resolution text operation is invalid. Check the display mode and coordinates.
See also HPRINT, HSCREEN, Draw a planet you can find again.

HR — High-resolution graphics error

A CoCo 3 graphics argument or buffer is invalid. Check coordinates, mode and allocated buffer size.
See also HSCREEN, HBUFF, HGET, HPUT, Draw a planet you can find again.

Check the open file

Look for OPEN, its file number and its mode, then follow each read or write.
AO · BR · ER · FD · FM · FO · IE · NO · OB · SE

AO — Already open

The file is already open. Close it before reopening; on SDC, an image cannot be mounted twice.
See also CLOSE, OPEN, DRIVE — SDC-DOS, Keep a log past bedtime.

BR — Bad record

The direct-access record number is invalid.
See also GET — disk record, PUT — disk record, LOC, LOF, Keep a log past bedtime.

ER — End of record

A direct-access operation went beyond the current record boundary. Check record size and fields.
See also OPEN, FIELD, Keep a log past bedtime.

FD — Bad file data

The next input value cannot be read into the requested variable type.
See also INPUT, INPUT — disk file, WRITE, Keep a log past bedtime.

FM — File mode

The operation does not match the mode used to open the file.
See also OPEN, Keep a log past bedtime.

FO — Field overflow

FIELD definitions exceed the file buffer's record size.
See also FIELD, OPEN, Keep a log past bedtime.

IE — Input past end

The program tried to read beyond the end of a file. Test EOF before requesting another item.
See also EOF, Keep a log past bedtime.

NO — File not open

Open the file before reading, writing or requesting its records.
See also OPEN, CLOSE, Keep a log past bedtime.

OB — Out of buffers

No suitable disk buffer is available. Allocate the required file buffers before opening files.
See also FILES, OPEN, Keep a log past bedtime.

SE — Not a field string

LSET or RSET was given a string that is not currently mapped by FIELD.
See also FIELD, LSET, RSET, Keep a log past bedtime.

Check the disk or device

Check the filename and the device that answered the request.
AE · DF · DN · FN · FS · IO · NE · VF · WP

AE — File already exists

The destination name is already in use. Choose another name or use the documented replacement behavior.
See also COPY, COPY — RGB-DOS, RENAME, Take the voyage to another machine.

DF — Disk full

There is insufficient space for the write. Use another working image or remove unneeded files carefully.
See also FREE, DIR, Take the voyage to another machine.

DN — Device number

The selected device or drive number is not valid for this operation.
See also OPEN, DRIVE, Take the voyage to another machine.

FN — Bad filename

The filename is not valid for the selected filesystem. Check its length, extension and drive syntax.
See also LOAD, SAVE, Take the voyage to another machine.

FS — Bad file structure

The disk file or filesystem structure is inconsistent with the requested operation.
See also DIR, Take the voyage to another machine.

IO — Input/output error

The requested media operation failed. Check mounting, format, filename and the selected drive.
See also DIR, DRIVE, CLOAD, Take the voyage to another machine.

NE — File not found

The named disk file was not found on the selected drive. Check DIR, extension and drive suffix.
See also DIR, LOAD, DRIVE, Take the voyage to another machine.

VF — Verification error

Data read back after a write did not verify. Preserve another copy before continuing to modify the image.
See also VERIFY, Take the voyage to another machine.

WP — Write protected

The selected media is read-only. Change protection only if you intend to modify that working copy.

See also SAVE, DRIVE — SDC-DOS, Take the voyage to another machine.

Publication and artwork notices

Original VCC Plus prose and instructional images © 2026 Brian K. Holman. Original component and example notices remain with their respective material.

Except where otherwise identified, original artwork owned by Brian K. Holman is © 2026 Brian K. Holman. All rights reserved.

Permission is granted to use and redistribute this artwork unchanged with VCC Plus, including modified versions of its GPL-licensed software. Other reproduction, modification or redistribution requires prior written permission, except as permitted by applicable law. Third-party materials remain subject to their respective licenses and rights.

The CoCo 3 and CM-8 monitor presentation in VCC Plus's skeuomorphic System view was inspired by SockMaster's "pic of CoCo 3 and CM-8 monitor," published on CoCopedia. VCC Plus's artwork was designed and rendered independently. The remaining hardware illustrations were independently designed using historical photographs and renderings as references.

Read this book online

CMOC license notice · LWTools COPYING notice